IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 16 October 2022, accepted 31 October 2022, date of publication 3 November 2022, date of current version 9 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3219068

RESEARCH ARTICLE

A Smart Home Energy Management System Utilizing Neurocomputing-Based Time-Series Load Modeling and Forecasting Facilitated by Energy Decomposition for Smart Home Automation

YU-HSIU LIN[1], HUEI-SHENG TANG[2], TING-YU SHEN[1], AND CHIH-HSIEN HSIA[2], (Member, IEEE)

[1]Graduate Institute of Automation Technology, National Taipei University of Technology, Taipei 106344, Taiwan
[2]Department of Computer Science and Information Engineering, National Ilan University, Ilan 260007, Taiwan

Corresponding author: Chih-Hsien Hsia (chhsia625@gmail.com)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 109-2221-E-027-121-MY2, Grant MOST 111-3116-F-006-005, Grant MOST 111-2221-E-027-050, and Grant MOST 111-3116-F-027-001.

ABSTRACT The key advantage of using power-utility-owned smart meters is the ability to transmit electrical energy consumption data to power utilities' remote data centers for various purposes, such as billing. Several useful consumer-centric use cases can also be identified for the collection and further analysis of consumers' electrical energy consumption data from smart meters. One of the use cases is home automation. Recent related solutions for home automation involving home security and healthcare depend on the installation of sensors and/or other devices such as video cameras, which have high costs for installation and annual maintenance. Because the electrical energy consumption patterns mined from smart meter data are indicative of residents' daily life, it is possible to develop a new home automation approach based on *energy decomposition* for smart home automation. Accordingly, in this work, a smart home energy management system (SHEMS) utilizing a parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism is proposed for smart home automation. Energy decomposition is used to facilitate the time-series load modeling and forecasting mechanism, which tracks appliance-level electrical energy consumption to be quantitatively modeled from circuit-level consumption, with no intrusive deployment of networked plug-level power meters for individual electrical home appliances. For the neurocomputing approach applied in this mechanism, an autoregressive multilayer perceptron methodology is compared against a stacked long short-term memory methodology. The presented neurocomputing-based time-series load modeling and forecasting mechanism facilitated by energy decomposition is capable of predicting residents' daily behavioral patterns by nonintrusively analyzing and modeling relevant electrical home appliances based on their past trends for smart home automation.

INDEX TERMS Artificial intelligence, energy decomposition, neurocomputing, smart home automation, smart grid.

NOMENCLATURE

A. LIST OF SYMBOLS

$x_i(t)$ on/off status of the i-th monitored electrical home appliance in the t-th time slot

$x_{t,w}^{i,d}$ on/off status of the i-th relevant electrical home appliance as determined through energy decomposition on the d-th weekday in the t-th time slot of the w-th week

τ_w chronological weight of $x_{t,w}^{i,d}$ for past $x_{t,w}^{i,d}$ in the w-th week

The associate editor coordinating the review of this manuscript and approving it for publication was Amjad Anvari-Moghaddam.

$UR_t^{i,d}$ quantified usage rate of the i-th relevant electrical home appliance on the d-th weekday in the t-th time slot, statistically computed in consideration of $\tau_w \cdot x_{t,w}{}^{i,d}$ from the past W weeks

P_i^{rated} rated power of the i-th relevant electrical home appliance

B. ABBREVIATIONS

AIoT	Artificial Intelligence of Things
HEMS	home energy management system
ADLs	activities of daily living
GPU	graphics processing unit
RFE	recursive feature elimination
DNN	deep neural network
UR	usage rate
MLP	multilayer perceptron
LSTM	long short-term memory
MAPE	mean absolute percentage error

I. INTRODUCTION

Smart cities can enhance our daily lives by providing comprehensive smart services such as smart transportation and energy. Electricity is one of the most popular forms of energy. As the electricity demands of consumers from downstream sectors of an electric power grid that is to be upgraded to a smart grid continuously increase, energy management schemes that encourage consumers to consume less electrical energy than they used to by raising their awareness of their consumption levels [1] are vital. With the implementation of such energy management, these ever-increasing electricity demands can be satisfied.

Smart grids can make existing cities ready for future needs, such as those of smart homes. In brief, a smart grid can be defined as an electric power grid that uses smart metering, automatic monitoring, and intelligent control techniques in conjunction with information and communication technologies and Artificial Intelligence of Things (AIoT) technology to operate more efficiently, enabling energy conservation and carbon emission reduction [2], [3], [4], [5], [6], [7]. In a smart grid with widespread deployment of power-utility-owned smart meters instead of traditional rotating-disc meters, it has become possible to automatically collect electrical energy consumption data at much more fine-grained spatial and temporal resolutions and to analyze these data at a more granular level for several useful consumer-centric use cases [8], [9], [10].

Fig. 1 depicts the location of a smart meter installed for a residential consumer in a smart electric power grid. As depicted in Fig. 1, the smart meter is located at the entry point of the residential building's electric grid connection. In a smart grid, the key advantage of using smart meters is the ability to transmit metered electrical energy consumption data to a power utility's remote data center [10]. Through the automated collection of consumers' electrical energy consumption data and the analysis of these data via artificial intelligence (AI), several novel use cases can be realized. From the grid-level monitoring perspective, the smart meter in Fig. 1 is distinct from a consumer-side monitoring system such as a building-level home energy management system (HEMS). Such an HEMS can communicate with monitoring devices such as wirelessly networked plug-level power meters to keep track of domestic devices such as electrical home appliances in a home environment. As a revolutionary paradigm for a connected home environment, a smart home can be defined as a residence developed with AIoT technologies to enable the provision of useful user-centric services to its occupants. Instead of addressing the building-level or grid-level monitoring perspective, this work considers time-series load modeling and forecasting for smart home automation as one such service relevant to appliance-level monitoring. The smart meter in Fig. 1, which is located at the entry point of the building's electric grid connection, can acquire circuit-level electrical energy consumption data for building-level and grid-level monitoring purposes, but it cannot decompose them into appliance-level electrical energy consumption data for appliance-level monitoring purposes. To achieve appliance-level monitoring, a traditional HEMS identifies electrical energy consumption of relevant individual electrical home appliances by means of plug-level power meters, i.e., smart plugs, attached directly to these appliances. This load monitoring approach, an intrusive load monitoring approach, requires field deployment of these HEMS instruments and incurs a high investment burden, including installation and annual maintenance costs. As an alternative to this intrusive approach, energy decomposition has been developed as a so-called nonintrusive load monitoring (NILM) technique to enhance the practical applicability of HEMSs for the detailed (appliance-level) monitoring of electrical energy consumption [11], [12], [13], [14], [15], [16], [17], [18]. A comparative summary of energy decomposition vs. intrusive load monitoring can be found in [14] and [19]. Being part of an (H)EMS, energy decomposition is the process of decomposing circuit-level electrical energy consumption data, i.e., smart meter data, into appliance-level electrical energy consumption data with no need for any additional plug-level power meters, thus keeping the investment costs, including installation and annual maintenance, of the (H)EMS to a minimum. Refs. [10], [20] present a comprehensive review of recent trends in energy decomposition.

Smart meter data gathered in, for instance, residential environments and analyzed through energy decomposition based on signal processing and AI techniques can support several useful user-centric use cases [10], [20], [21]. One of these use cases is home automation, including anomaly detection for home security [22] and the recognition of activities of daily living (ADLs) for healthcare applications [22], [23], [24], [25]. The current techniques adopted in HEMSs for such home automation applications depend on the installation and use of sensors and/or other devices such as video cameras, which also require additional installation and annual maintenance costs [20]. However, an HEMS that exploits

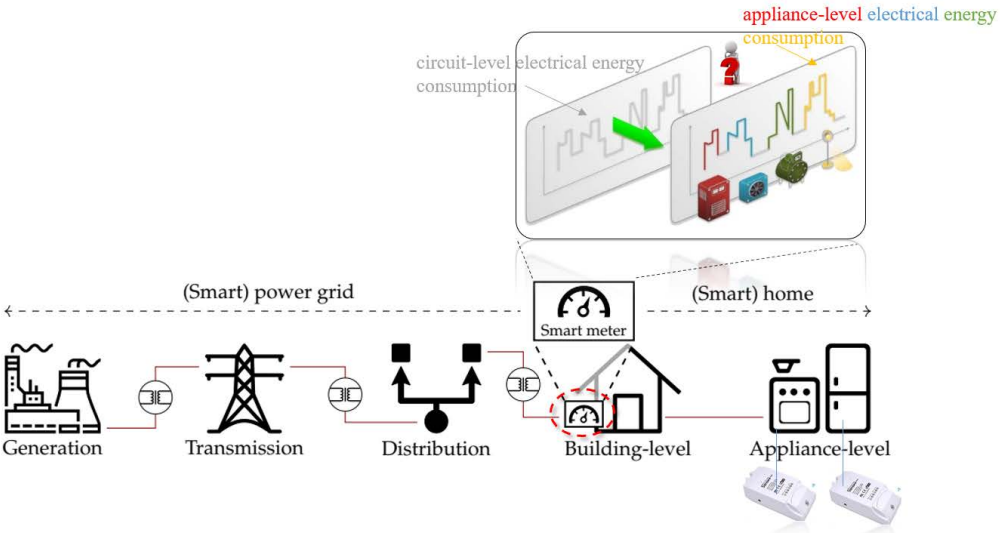


FIGURE 1. Location of a smart meter installed for a residential consumer using the public utility service for domestic purposes in a smart electric power grid. The smart meter capturing all power flows between the electric grid and the building can lead to potential benefits on both sides. For example, ideally, the balance between energy consumption and energy generation can be optimized.

energy decomposition for home automation purposes (based on the premise that electrical energy consumption patterns are indicative of residents' daily lives) does not require such additional costs. Hence, the complexity of implementing practical home automation for home security and healthcare can be reduced.

Fig. 2 shows a sample of acquired circuit-level power demand over 24 hours, from 00:00 to 24:00, in a realistic house environment, which shows the feasibility of the identification of ADLs from such data [23]. This work presents a smart HEMS (SHEMS) utilizing a parallel-processing-implemented, GPU-accelerated *neurocomputing*-based time-series load modeling and forecasting mechanism for smart home automation. In contrast to a conventional home automation system with multiple devices, sensors and/or actuators installed as home furnishings to unobtrusively collect behavioral data in the home environment, the load modeling and forecasting mechanism presented here is based on energy decomposition and is capable of predicting residents' daily behavioral patterns by nonintrusively analyzing and modeling relevant individual electrical home appliances based on the usage trends observed in circuit-level electrical energy consumption data, which can be obtained from a power-company-owned smart meter. Relevant electrical home appliances can reflect behavioral routines of occupants using electricity [26]. The work presented here is an extension of the previous work reported in [12], [13], and [23], which presents a new methodology for modeling relevant individual electrical home appliances by means of energy decomposition applied on circuit-level electrical energy consumption data, with a very low level of intrusiveness [27]. The present work is different from that described in [23] and [24], where the load modeling and forecasting mechanism was a heuristic-based mechanism.

This paper is organized as follows. First, the methodology proposed in this work is presented in Sec. II. Then, Sec. III describes the conducted experiments. Finally, the paper is concluded with a discussion of future work in Sec. IV.

II. METHODOLOGY

An overview of the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism (as a module) based on energy decomposition is depicted in Fig. 3. Extended from the work presented in the previous studies [12], [23], this work presents a neurocomputing-based time-series load modeling and forecasting mechanism based on energy decomposition for smart home automation, in which 1) a set of neurocomputing tools accelerated by a graphics processing unit (GPU) are processed in parallel for time-series load modeling and forecasting and 2) the energy decomposition method used to facilitate time-series load modeling and forecasting is improved. For the neurocomputing tools performed in the presented mechanism, an autoregressive multilayer perceptron (MLP) methodology is compared with a stacked long short-term memory (LSTM) methodology.

A. HARDWARE—THE HEMS

As depicted in Fig. 3, a house utilizing the HEMS in [13], [23] is connected via advanced metering infrastructure (AMI) to a smart grid. As a complement to the demand response (DR) and non-context-aware human life-pattern identification implementations presented in [13] and [23], this work

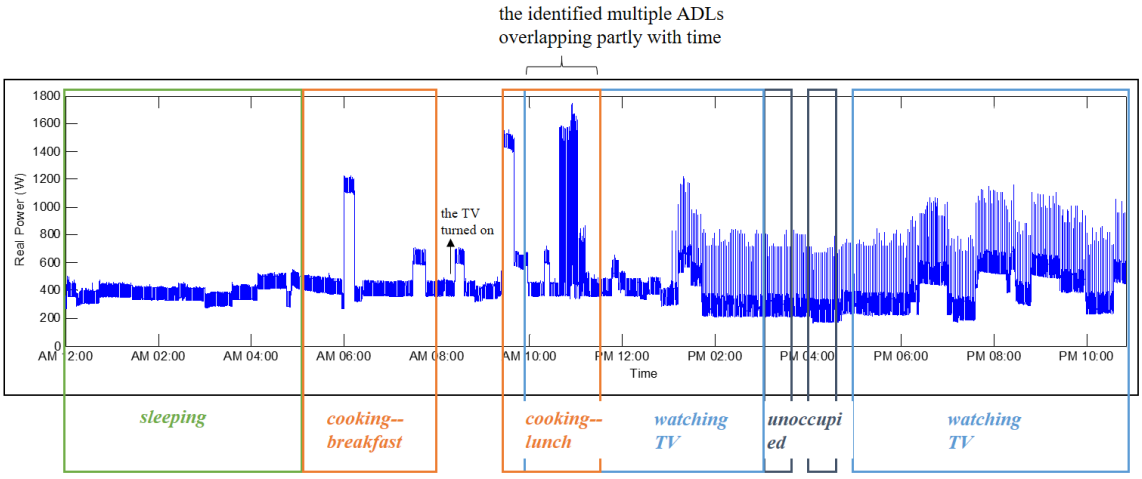


FIGURE 2. A sample of acquired circuit-level power demand data over 24 hours, from 00:00 to 24:00, in a realistic house environment [23].

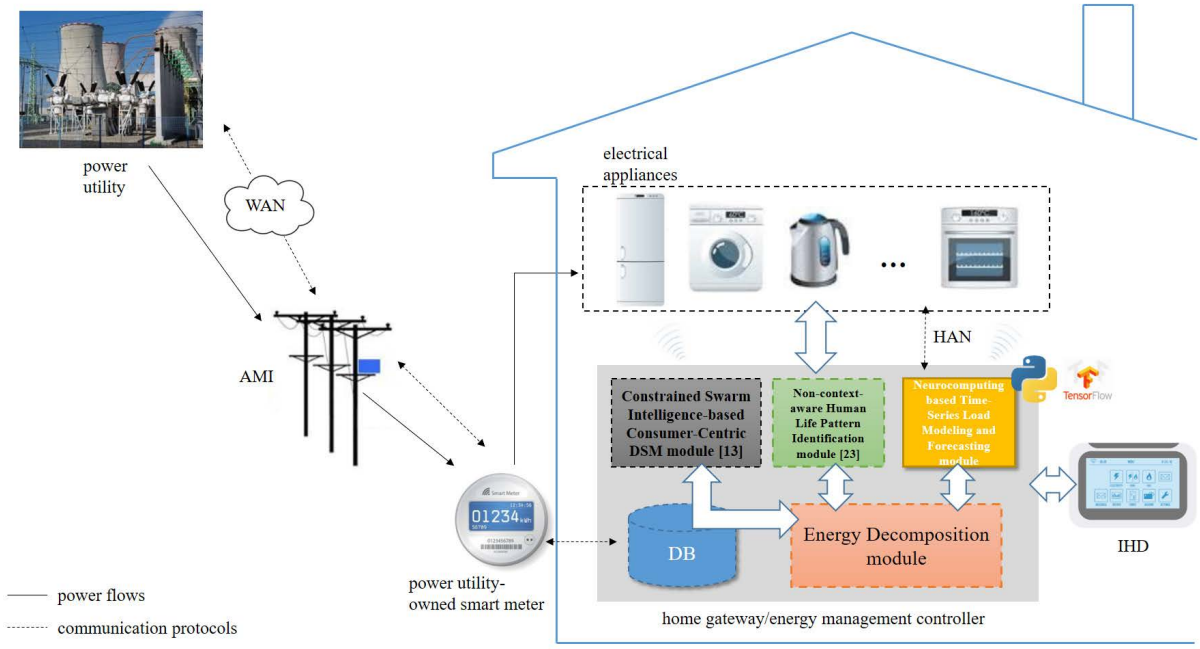


FIGURE 3. A schematic diagram of the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism, based on energy decomposition, implemented as a module of the HEMS in [13] and [23]. In [13] and [23], a metaheuristic-based residential-consumer-centric demand-side management (DSM) model and a heuristic-based non-context-aware human life-pattern identification method have previously been developed.

presents a time-series load modeling and forecasting implementation based on energy decomposition for smart home automation, which can also be implemented for residential load management. As seen in Fig. 3, the house with the HEMS (implementing the mechanism presented here) mainly includes 1) a smart meter used to receive DR signals from a power utility via AMI for the implementation of demand-side management (DSM) as described in [13]; 2) a home gateway, implemented on an embedded system as in [12] or a laptop as in this work, which acts as an energy management controller in the house; and 3) relevant electrical home appliances networked wirelessly via a home area network and monitored through energy decomposition for the identification of ADLs as described in [23] and for smart home automation/residential load management based on time-series load modeling and forecasting as described in this work. In the house, two different types of installed sensors work together in a sensor-fusion sense: 1) plug-level ZigBee-based control relays, acting as switching gadgets, installed for remote load control and used to label relevant electrical home appliances

with their load combination classes (a one-time intrusive setup period for energy decomposition is needed for the load combinations to be labeled and addressed through multi-class/multilabel load identification [28]. Alternatively, this one-time intrusive setup period can be achieved through a preliminary stage of energy decomposition as described in [29]), and 2) a minimal set of circuit-level current and voltage sensors used to acquire composite electrical energy consumption data, as smart meter data, to be decomposed through energy decomposition into appliance-level electrical energy consumption data. The energy decomposition results, such as the time durations of use for each individual monitored electrical home appliance, are very useful for further analysis for applications such as the time-series load modeling and forecasting mechanism proposed in this work. Regarding the home gateway, it can be implemented based on a BeagleBoard embedded system with an OMAP3530 720 MHz ARM® Cortex™-A8 processor, in which an Apache HTTP web server, MySQL relational database, and PHP sous Linux OS stack can be configured [12]. Here, the presented work is implemented based on scikit-learn [30] and TensorFlow™[31] in the Python programming language.

B. SOFTWARE–PROPOSED NEUROCOMPUTING-BASED TIME-SERIES LOAD MODELING AND FORECASTING MECHANISM BASED ON ENERGY DECOMPOSITION

Fig. 4 shows the workflow of the presented neurocomputing-based time-series load modeling and forecasting mechanism based on energy decomposition. The energy decomposition procedure involved in the mechanism is described in Sec. II-B-1). The mechanism by which the energy decomposition results are used for time-series load modeling and forecasting is described in Sec. II-B-2). In this mechanism, a set of deep learning (DL)-based neurocomputing models developed as load behavior learners for predicting the routine weekday behaviors of relevant individual electrical home appliances is parallelized, wherein the deep neural networks (DNNs) are accelerated by a GPU. Because considerable computational power is required to allow a DNN to learn from data, such a model can be trained faster by simply running all operations (matrix multiplications) simultaneously rather than one after the other. To achieve this, we use a GPU to train our DNNs in this work. Moreover, with such utilization of GPU-accelerated DNNs, it is necessary to speed them up by exploiting parallel processing, as they have very high computational requirements due to 1) the large number of input–output data pairs (note that the time-series load modeling is framed as supervised learning) to be learned and 2) the need to iteratively execute the algorithmic routines.

1) ENERGY DECOMPOSITION

The basic energy decomposition process can be formulated as shown in Eq. (1), which is referenced from [28]. In Eq. (1), 1) $P(t)$, the total power consumption ($= y(t)$), corresponds to the circuit-level composite electrical energy consumption recorded in the t-th time slot; 2) $P_i(t)$ represents the real power consumption of the i-th individual monitored electrical home appliance ($i = 1, 2, \ldots, N$) in the t-th time slot; 3) $x_i(t)$ indicates the on/off status of the i-th appliance in the t-th time slot, taking values of {0, 1} where 0 corresponds to the "off" status and 1 corresponds to the "on" status; and 4) $P_{base}(t)$ represents the base load, comprising permanent/phantom loads, which can be profiled in advance during a one-time intrusive setup period or a preliminary stage of energy decomposition. Given $P(t)$ ($= y(t)$) as acquired from the smart meter and knowledge of $P_{base}(t)$ and $P_i(t)$, which can also be profiled during the one-time intrusive setup period or the preliminary stage of energy decomposition, energy decomposition can deduce the unknown $x_i(t)$ for the N individual monitored appliances in the t-th time slot: $[x_1(t), x_2(t), \ldots, x_i(t), \ldots, x_N(t)] = F(y(t))$, where F is a well-trained AI model. This model can return the most appropriate N estimates of $x_i(t)$ for all N relevant appliances in the t-th time slot such that the objective metric given in Eq. (2) [28] is minimized. The optimization objective in Eq. (2) is expressed as a sum of superimposed absorptions $\Sigma x_i(t)P_i(t)$ to be deduced with respect to $P(t)$ and $P_{base}(t)$ so as to minimize the error $\varepsilon(t)$. In this work, we use an ensemble machine learning approach to deduce the $x_i(t)$ for energy decomposition of Eq. (2).

$$P(t) = \sum_{i=1}^{N} x_i(t)P_i(t) + P_{base}(t) \tag{1}$$

$$\varepsilon(t) = \left| P(t) - \sum_{i=1}^{N} \hat{x}_i(t)P_i(t) - P_{base}(t) \right| \tag{2}$$

In this work, the energy decomposition workflow shown in Fig. 4 consists of the following three steps: 1) *data acquisition*; 2) *feature extraction*, including data reduction and feature selection; and 3) *load identification*, for which a random forest ensemble is implemented. In the data acquisition step, both composite circuit-level current and voltage signals were continuously and simultaneously acquired [12]. In the feature extraction step, electrical features were extracted from the acquired circuit-level current and voltage signals, including the real power (P), reactive power (Q) and current harmonics of up to the 11th order [12], [23], and data reduction and feature selection were additionally performed [12], [23]. The energy decomposition method described in [12] and [23] involved taking a feature reading every minute and then processing it for the collection of the extracted features to be identified. In [12] and [23], feature selection was performed by sequential feature selection. In this work, recursive feature elimination (RFE) [32] is applied instead. RFE works by recursively removing features and then building a model based on those features that remain; for this purpose, the model accuracy is used to identify which combination of features contributes the most to predicting the target. More details on RFE can be found in [32]. In the load identification step, the status of relevant electrical home appliances monitored nonintrusively for their usage traces is deduced

by a random forest ensemble [18], [33] in this work, and this approach will be compared against the load identification approaches in [12] and [23]. In [12], a backpropagation artificial neural network (BP-ANN)-based ensemble was used in combination with sequential feature selection. In [23], a bagging-decision-tree-based ensemble, an ensemble in which decision trees are used as the basis for bootstrap aggregation [17], was applied in combination with the same feature selection method used in [12]. In this work, a random forest ensemble with feature selection based on RFE is employed as the load identifier for the individual monitored appliances, and the identified loads are further analyzed and modeled for time-series load forecasting. To this end, RFE and the random forest ensemble are implemented in Python on a laptop. In addition, scikit-learn [29] is suited on the laptop to run the presented parallel-processing-implemented, GPU-accelerated time-series load modeling and forecasting mechanism based on energy decomposition.

2) NEUROCOMPUTING-BASED TIME-SERIES LOAD MODELING AND FORECASTING

This work proposes the parallel-processing-implemented, GPU-accelerated time-series load modeling and forecasting mechanism shown in Fig. 4, which is based on energy decomposition. The presented mechanism involves the following three steps.

Step 1): Parse historical energy decomposition data (on/off statuses of nonintrusively monitored electrical home appliances) to obtain the usage rates (URs) of the relevant appliances through energy decomposition for time-series load modeling and forecasting.

Step 2): Frame the time-series data, i.e., the URs parsed in *Step 1)*, in a supervised learning manner and build DNNs as load behavior learners to predict load behaviors of the relevant appliances that are representative of their routines on weekdays for smart home automation (residential load management).

Step 3): Train the DNNs built in *Step 2)* on the past trends of the framed time-series data to forecast the future URs of the individual monitored appliances based on their on/off statuses. For this purpose, parallel processing is adopted for the parallel training of the GPU-accelerated DNNs.

The three steps above are detailed in the following subsections.

a: PARSING HISTORICAL ENERGY DECOMPOSITION DATA TO OBTAIN THE URS OF RELEVANT MONITORED ELECTRICAL HOME APPLIANCES FOR TIME-SERIES LOAD MODELING AND FORECASTING

The historical energy decomposition data are parsed for time-series load modeling and forecasting as described below. Suppose that there are T time slots in a day, where $\boldsymbol{T} = \{1, 2, \ldots, t, \ldots, T\}$ and $\boldsymbol{H} = \{1, 2, \ldots, h, \ldots, H = 24 \text{ hours}\}$. For instance, T may be equal to 1,440, meaning that the time resolution is 1 minute. Further suppose that the i-th relevant electrical home appliance is considered for time-series load modeling and forecasting, meaning that energy decomposition is used to identify its on/off routines on D weekdays, where $\boldsymbol{D} = \{1, 2, \ldots, d, \ldots, D = 5 \text{ days (Monday through Friday)}\}$, for W weeks, where $\boldsymbol{W} = \{1, 2, \ldots, w, \ldots, W\}$. Then, the quantified UR of the i-th appliance on the d-th weekday in the t-th time slot can be statistically computed in consideration of $\tau_w \cdot {x_{t,w}}^{i,d}$ from the past W weeks, as shown in Eq. (3).

$$UR_t^{i,d} = \frac{\sum_{w \in W} \tau_w \cdot x_{t,w}^{i,d}}{W}, \quad \forall t \tag{3}$$

In Eq. (3), ${x_{t,w}}^{i,d}$ represents the on/off status of the i-th appliance as determined through energy decomposition on the d-th weekday in the t-th time slot of the w-th week; ${x_{t,w}}^{i,d}$ takes values of $\{0, 1\}$ where 0 indicates that the appliance status is "off" and 1 indicates that the appliance status is "on". In Eq. (3), the ${x_{t,w}}^{i,d}$ values are weighted chronologically by weights τ_w, which should satisfy Eq. (4). In this work, we use a set of neurocomputing-based behavior learners to model the occupants' life patterns. The occupants' life patterns are quantified chronologically in a weighted form (based on their routines from the past several weeks) as expressed in Eq. (3) and are then adaptively learned for ongoing forecasting, as the routines to be modeled may change over time.

$$\sum\nolimits_{w \in \mathbf{W}} \tau_w = 1 \tag{4}$$

Note: the predicted $UR_{t,d}$ values can be transformed into an activation profile for the i-th electrical home appliance of interest. If the predicted value is greater than or equal to a prespecified probability threshold, then the i-th appliance is considered to be "on"; otherwise, it is considered to be "off".

b: BUILDING DNNS FOR TIME-SERIES LOAD MODELING AND FORECASTING

In this work, neurocomputing is used to perform time-series load modeling and forecasting for relevant electrical home appliances monitored through energy decomposition. Neurocomputing is a type of computing based on an artificial neural network (ANN) mimicking a biological neural network (NN), which takes the form of a system of processing units known as artificial neurons that can be linked together in various ways to learn from arbitrary nonlinear data examples [1]. As a type of neurocomputing, DL is usually performed based on an ANN architecture; however, a traditional ANN contains only two or three hidden layers, whereas a DNN can have many hidden layers, even hundreds. Fig. 5 illustrates the general structure of an ANN, specifically, a feedforward multilayer DNN serving as an autoregressive MLP. It is designed and implemented for time-series load modeling and forecasting based on energy decomposition for smart home applications like smart home automation here. This model is shown in order to motivate innovations in the application thereof. As illustrated in Fig. 5, a general ANN consists of an input layer, some numbers of hidden layers, and an output layer.

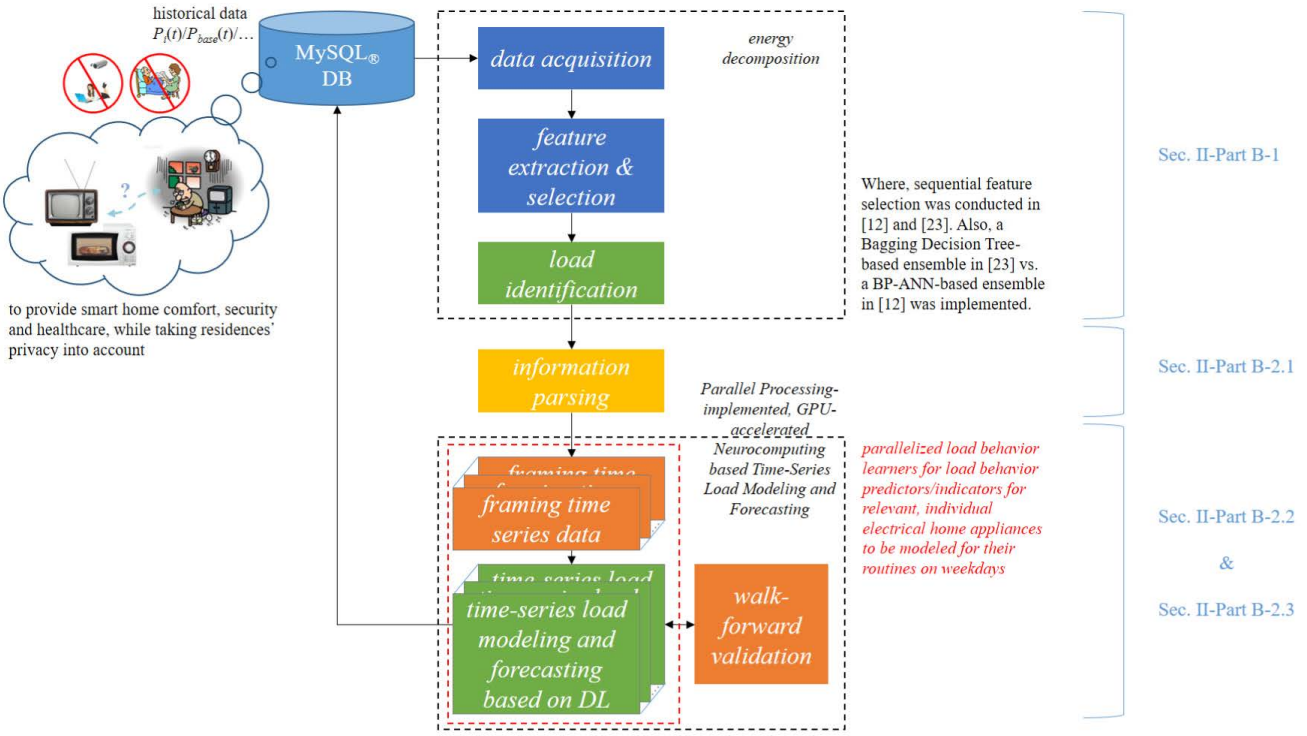


FIGURE 4. Workflow of the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism based on energy decomposition.

The size of the input layer depends on the number of independent input variables of the observed data to be learned. The number of hidden layers and the number of hidden neurons used in each hidden layer can be determined via hyperparameter tuning, where the hyperparameters affect how well the model can represent the observed data, as assessed through learning and testing. The hyperparameters include the hidden layer sizes, the training algorithm, the number of epochs of iterative training, and so on. The size of the output layer usually depends on the number of dependent output variables of the observed data. For a model such as the fully connected feedforward DNN illustrated in Fig. 5, the rectified linear unit (ReLU)-style activation function is commonly used for the neurons in the hidden layers, rather than an S-shaped function such as the sigmoid function, in order to overcome gradient vanishing [35], [36]; in the context of DNNs, the problem of vanishing gradients can often arise for a model configured with S-shaped activation functions for the hidden neurons to be trained by a stochastic gradient-based weight optimizer such as Adam, which was first proposed in [37], for their most appropriate weight coefficients including biases.

A simple form of time-series forecasting can be described as follows: upon the observation of a data point y_t in the t-th time slot, a one-step-ahead forecast $\hat{y}_{t+1}$ can be computed by n preceding data observations, namely, y_t, y_{t-1}, y_{t-2}, y_{t-3}, $\ldots$, y_{t-n-1}, from the t-th, (t-1)-th, (t-2)-th, (t-3)-th, $\ldots$, (t-n-1)-th time slots, respectively. This can be formulated as shown in Eq. (5).

$$\hat{y}_{t+1} = f(y_t, y_{t-1}, y_{t-2}, y_{t-3}, \ldots, y_{t-n-1}) \tag{5}$$

In Eq. (5), f can be realized by means of a well-trained feedforward multilayer ANN. In this work, however, a small difference with respect to Eq. (5) is considered. The problem addressed in this work is a multistep time-series forecasting problem, which is addressed by the model illustrated in Fig. 5. The model in Fig. 5 is developed as a multistep time-series forecasting model that predicts a vector output consisting of the URs, as defined in Eq. (3), of the i-th individual monitored electrical home appliance in terms of its on/off status in multiple time steps. That is, given an input vector consisting of data from the last *n_steps_in* time steps (the raw observed, trended time-series URs selected by a sliding window), the model illustrated in Fig. 5 can output a vector of predicted URs for the next *n_steps_out* time steps as a multistep time-series forecast.

The developed model can be applied directly to the raw observed, trended time-series URs, with no need to make them stationary via differencing prior to learning; thus, it is capable of automatically extracting and learning features from these UR data for time-series load modeling and forecasting. Dropout, an effective and commonly used technique for preventing overtraining of an AI model, can also be used in this model. Walk-forward validation is an approach in which a time-series forecasting model generates a forecast

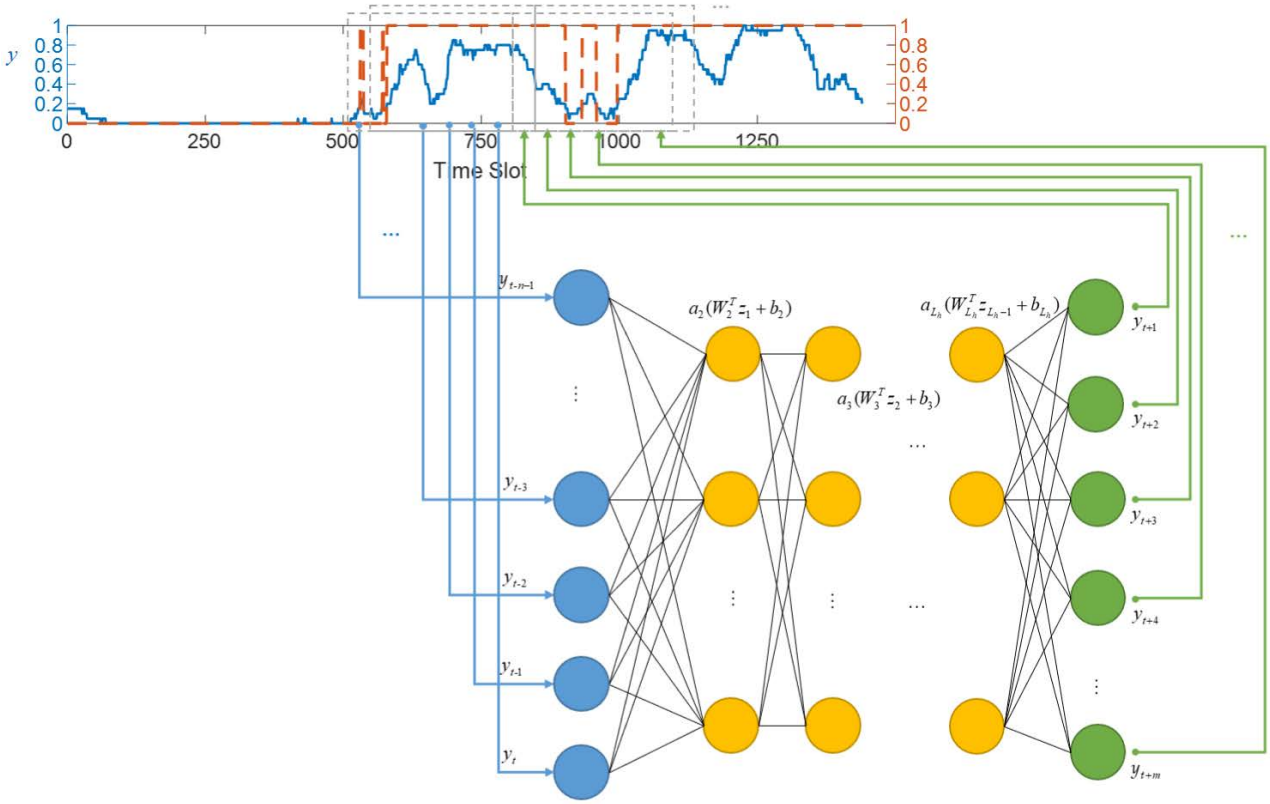


FIGURE 5. An autoregressive MLP, a fully connected feedforward DNN, applied to a trended time series of observed data for time-series forecasting. The network contains n input neurons for observations from the t-th, (t-1)-th, (t-2)-th, (t-3)-th,..., (t-n-1)-th time slots (the lagged observations are flattened and fed into the model) and m output neurons for forecasts for the (t+1)-th, (t+2)-th, (t+3)-th, (t+4)-th, ..., (t+m)-th time slots. The model has an arbitrary, complex topology that can be specified as desired for performing sophisticated (human-like) decision making. In the model, the L_h hidden layers can perform nonlinear data transformations $a_2(W_2^T z_1 + b_2)$, $a_3(W_3^T z_2 + b_3)$, ..., $a_{L_h}(W_{L_h}^T z_{L_h-1} + b_{L_h})$, where a_x denotes the activation function of the hidden neurons in the x-th hidden layer (various function types can be specified; for example, the typical activation functions used in feedforward DNNs include the sigmoid, tanh, and rectified linear unit (ReLU) functions), W_x represents the weight matrix, and b_x represents the bias vector. Dropout—an effective, commonly used technique for preventing overtraining [34] of an ANN—can also be applied in the form of a hyperparameter with a specified value to control the model training process; another technique used to prevent overtraining is early stopping.

for each observation one at a time. After a forecast has been generated for a time step, the true observation corresponding to that forecast is added and made available to the model. In this work, the presented model is evaluated through walk-forward validation.

Recurrent neural networks (RNNs) are a special type of ANNs that are adapted for applications to time-series data such as univariate time-series data. In practice, RNNs tend to suffer from the vanishing gradient problem. LSTM networks, which are a sequential model including specific forget, input and output gates, were invented to resolve the vanishing gradient problem faced by RNNs. LSTM networks can also be used to solve univariate time-series forecasting problems comprised of a single series of past observations, and an LSTM network is the most appropriate type of predictive model for modeling sequential data with long-term dependencies. Fig. 6 depicts the typical architecture of an LSTM cell, which can be trained through repetition over time to identify the long-term dependency between time steps and a time series of sequential data. The output of the LSTM cell depicted in Fig. 6 can be expressed as shown in Eq. (6). As seen in this equation, the output is dependent on both the input in the current time step ($\mathbf{x}_t$) and the previous hidden state ($\mathbf{h}_{t-1}$) (the output gate determines what the next hidden state should be). Moreover, the output is modulated by considering the current cell state ($\mathbf{c}_t$). Through a training process, the weight and bias parameters in Eq. (6), W_o, $\mathbf{b}_o$, W_c, $\mathbf{b}_c$, W_i, $\mathbf{b}_i$, W_f and $\mathbf{b}_f$, can be learned from the available data. A stacked LSTM network is a variant of a typical LSTM network. It consists of a series of stacked LSTM layers used to learn from time-series data for sophisticated decision making. Fig. 7 illustrates the architecture of the N-layer stacked LSTM model adopted for univariate time-series load modeling and forecasting in this work. In this work, a stacked LSTM model and a model based on autoregressive MLP are both used for multistep time-series load forecasting, and their performance is compared.

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \tag{6}$$

In Eq. (6), $\mathbf{o}_t = \sigma(W_o\mathbf{x}_t + W_o\mathbf{h}_{t-1} + \mathbf{b}_o)$, where 1) $\mathbf{x}_t$ is the input to the LSTM cell in the current time step, 2) $\mathbf{h}_{t-1}$ represents the output of the previous hidden state, and

3) W_o and $\mathbf{b}_o$ are the weight and bias parameters, respectively, of the output gate. σ denotes the sigmoid activation function. $\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}}_t$, where $\mathbf{c}_{t-1}$ represents the cell state that is previous to $\mathbf{c}_t$ and $\tilde{\mathbf{c}}_t$ denotes an intermediate cell state. $\mathbf{f}_t = \sigma(W_f \mathbf{x}_t + W_f \mathbf{h}_{t-1} + \mathbf{b}_f)$, where W_f and $\mathbf{b}_f$ are the weight and bias parameters, respectively, of the forget gate, which controls information removal and retention when calculating $\mathbf{c}_t$ based on the memory of the previous cell state ($\mathbf{c}_{t-1}$). $\mathbf{i}_t = \sigma(W_i \mathbf{x}_t + W_i \mathbf{h}_{t-1} + \mathbf{b}_i)$, where W_i and $\mathbf{b}_i$ are the weight and bias parameters, respectively, of the input gate, which updates the cell state ($\mathbf{c}_t$) based on the intermediate cell state ($\tilde{\mathbf{c}}_t$). $\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + W_c \mathbf{h}_{t-1} + \mathbf{b}_c)$, where W_c and $\mathbf{b}_c$ are the weight and bias parameters, respectively, for the intermediate cell state. Finally, denotes the Hadamard product (element wise multiplication) [38].

c: TRAINING DNN MODELS WITH PARALLEL COMPUTING AND A GPU

For a DNN, a complex ANN that mimics sophisticated (human-like) decision making, a Compute Unified Device Architecture (CUDA)-enabled GPU can be used to speed up the network for its training process. A GPU is capable of performing massive parallel computations simultaneously, and the computation routines that must be executed to update the weights of a network during training involve large amounts of matrix multiplications that can be parallelized. In this work, the training process for networks built based on the model illustrated in Fig. 5 is accelerated by a GPU to reduce the computation time needed for the networks to learn the trends of the URs of individual monitored electrical home appliances in order to predict their future on/off statuses. Parallel processing is a mode of operations in which tasks (processes) are executed simultaneously on multiple processors on one or multiple machine(s). In parallel processing, there are two types of execution: 1) synchronous execution, in which the processes are completed *in the same order* in which their execution was started (synchronous execution is achieved by locking the main program until the corresponding processes are done), and 2) asynchronous execution, in which the parallel processes are allowed to be executed *asynchronously* (in asynchronous execution, a process may start as soon as a previous one has finished, without regard for the starting order; this approach does not involve locking). In this work, in addition to the use of a GPU the speed up the model training process, asynchronous parallel processing is adopted for the parallel training of GPU-accelerated neurocomputing-based load behavior learners for individual relevant electrical home appliances to be predicted for their future on/off statuses by the established, well-trained learners.

3) USING F_1 SCORE TO EVALUATE TIME-SERIES LOAD FORECASTING MODELS

Accuracy [12] alone does not tell the full story of the performance achieved when a classifier/learner is used as a predictor on a class-imbalanced dataset. In this work, as shown in Eq. (7), F_1 *score* [15], [31], [39] is used to evaluate the prediction performance of the energy decomposition and time-series load forecasting approaches.

$$F_1 score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7}$$

In Eq. (7), precision is the ratio of the total number of correctly predicted positive samples to the total number of predicted positives, and recall (i.e., the sensitivity or hit rate) is the ratio of the total number of correctly predicted positive samples to the total number of actual positives.

As shown in Eq. (7), F_1 score is the harmonic mean of precision and recall. A classifier/learner that produces no false positives when used as a predictor has a precision of 1.0, and it has a recall of 1.0 if it produces no false negatives. Thus, as seen from Eq. (7), F_1 score has a maximum value of 1.0 (perfect precision and recall). The higher the score is, the better the prediction performance. Eqs. (8) and (9) show how to compute the true positive rate (TPR) (a synonym for recall) and the false positive rate (FPR), respectively. These metrics are typically used to construct the receiver operating characteristic (ROC) curve [40], [41], [42] as another tool for evaluating a classifier (a learner acting as a predictor). In addition to the area under the curve (AUC), where a larger AUC is usually better, the ''steepness'' of the ROC curve is also important. This is because it is ideal to maximize the TPR while minimizing the FPR; hence, the top left corner of the ROC curve plot is the ''ideal'' point for classification/prediction. Based on the ROC curve, the error of a trained and tested classifier/learner serving as a predictor can be computed based on the Euclidean distance from the ideal classification/prediction point (FPR=0, TPR=1) to the evaluated (FPR, TPR) point [16], [18]. Accordingly, in addition to using F_1 score, this work also uses ROC curves, which are plotted as FPR vs. TPR, to evaluate the prediction performance of the energy decomposition approach based on a random forest ensemble and of the time-series load forecasting approach based on parallel-processing-implemented, GPU-accelerated neurocomputing.

$$TPR = \frac{TP}{TP + FN} \tag{8}$$

$$FPR = \frac{FP}{FP + TN} \tag{9}$$

In Eqs. (8) and (9), the number of true positives (TP) is the number of data, i.e., examples/observations, that are predicted to be positives and are also positive examples in reality. True negatives (TN) similarly refer to data that are correctly predicted to be negative observations. False positives (FP) are data that are erroneously predicted to be positives, and false negatives (FN) are data that are incorrectly predicted to be negatives.

III. EXPERIMENTS

This section demonstrates the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism based on

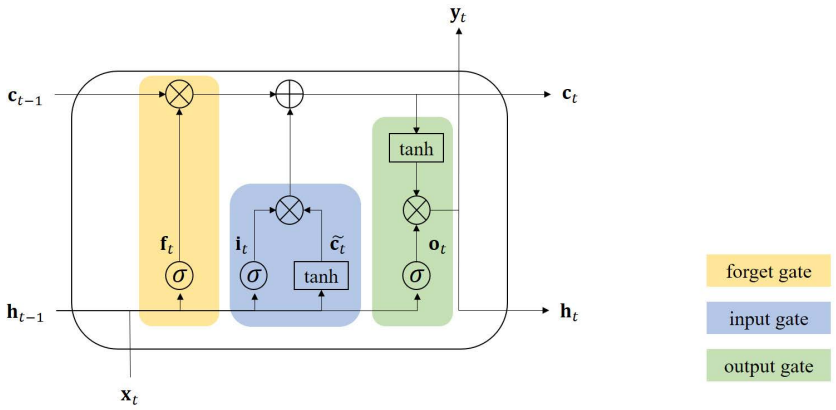


FIGURE 6. A typical LSTM cell architecture.

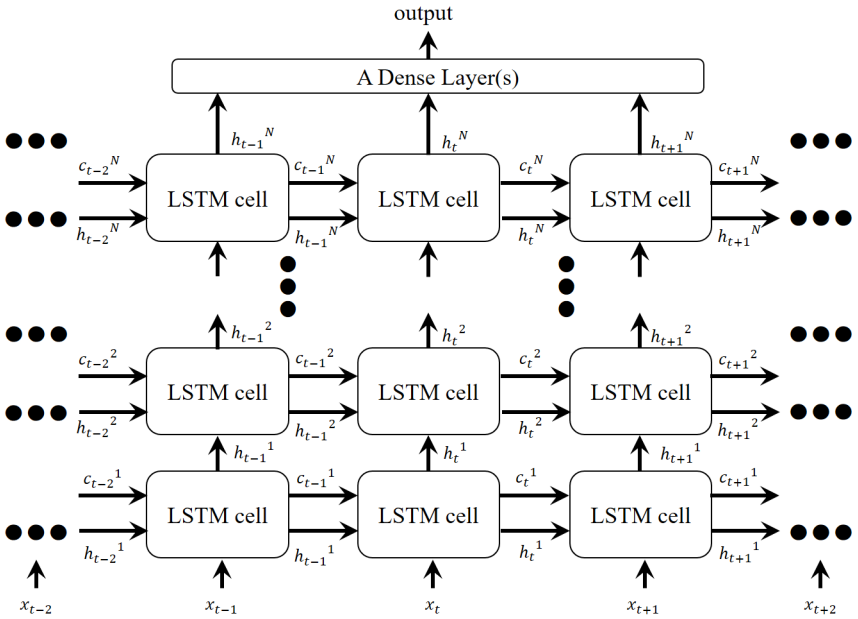


FIGURE 7. A stacked LSTM architecture that can learn a function mapping a given sequence of past observations serving as inputs to an output observation (or a sequence of output observations for multistep time-series forecasting). The number of hidden LSTM cells in each hidden LSTM layer is equal to the number of time steps. Each hidden LSTM cell is made up of hidden units to be prespecified for model training.

energy decomposition. The electrical home appliances monitored in the house environment in [12] and [23] are tabulated in Table 1. The rated power (wattage), denoted by P_i^{rated}, of each appliance listed in Table 1 was statistically computed (estimated) from historical circuit-level power consumption data recorded when only one appliance was in use at a time. The wattage of the base load in the house environment was similarly statistically computed to be 0.31 kW. In [12], the typical usage of the appliances was learned and modeled based on the onsite-collected feature data. Specifically, in addition to the electrical features P and Q, current harmonics of up to the 11^{th} order extracted via the fast Fourier transform (FFT) from the circuit-level current signals were considered as the potential electrical features for further selection and processed. In [12], the feature subset $\{P, Q, I_{Mag}{}^{2\text{nd}}, I_{Mag}{}^{7\text{th}}\}$ was obtained (refer to [12] for more details about the data reduction and feature selection processes). The energy decomposition process, which is viewed as load identification, in [12] was based on a set of BP-ANNs that made up an ensemble designed to classify a total of 17 load combination scenarios (classes) in the house environment. As reported in [12], a total of 6 electrical home appliances were nonintrusively monitored in hot L1 of the residential electrical wiring. Theoretically, there are 64 ($=2^6$) possible load combination scenarios in total that should be considered when there are 6 appliances of interest. However, in practice,

47 load combination scenarios were excluded since 1) some of these possible scenarios involved power-intensive appliances that would overload the conductor and 2) some of the appliances to the remaining scenarios were not used and labeled during the one-time intrusive setup period for the NILM. As a result, only 17 possible load combination scenarios were considered in total, and these same scenarios are addressed for energy decomposition in this work. In [23], a bagging-decision-tree-based ensemble was used for energy decomposition/load identification instead of the BP-ANN-based ensemble used in [12]; however, this ensemble was applied based on the same feature subset.

In this work, a different energy decomposition methodology is applied for the same house environment and compared with the energy decomposition/load identification approaches used in combination with the sequential feature selection method in [12] and [23]: specifically, a random forest ensemble is used to classify feature data selected based on RFE rather than the sequential feature selection method used in the previous studies. The basic energy decomposition process can be formulated as shown in Eqs. (1) and (2). Notably, electrical home appliances may be identical in terms of P (and Q). Therefore, in practice, additional electrical features need to be considered as feature candidates for energy decomposition to obtain $x_i(t)$ in Eq. (1) such that Eq. (2) is optimized/minimized. In this work, energy decomposition is performed every minute to identify whether each monitored appliance was currently in use in the house environment. To this end, at each time point for load identification, 1-second composite current and voltage signals were acquired simultaneously and analyzed to extract their electrical features, including P, Q and the magnitude of the current harmonics of up to the 11^{th} order [12], as the feature candidates to be examined through RFE to resolve any ambiguities when identifying different appliances or load combinations thereof under similar P (and Q) conditions for energy decomposition. Notably, from the perspective of multilabel load identification for $x_i(t)$ [28], cyclic appliances such as washing machines, tumble dryers and dishwashers can be treated as appliances with varying P, meaning that the uncertainties arising due to identical values of P (and Q) can be addressed by means of additional extracted electrical features. The harmonic feature candidates considered and examined by means of RFE in this work were extracted by applying the FFT [12] to composite current signals acquired with a sampling frequency of 2 kHz and a signal duration of 1 second. The *RFE()* function [43] in scikit-learn was applied to implement RFE [32] for feature selection. In this experiment, the top 7 features $\{P, Q, I_{Mag}{}^{1\text{st}}, I_{Mag}{}^{3\text{rd}}, I_{Mag}{}^{5\text{th}}, I_{Mag}{}^{7\text{th}}, I_{Mag}{}^{9\text{th}}\}$ were chosen through RFE as the feature subset for consideration (the number of features to be selected was specified as 7). The *RandomForestClassifier()* function [43] in scikit-learn was applied to implement a random forest ensemble as a meta-estimator by fitting a number of base decision tree classifiers on various subsamples of the complete training dataset and then averaging to improve the predictive performance while controlling overfitting. The goal of using an ensemble as a meta-estimator is to introduce randomization and combine several base estimators to achieve improved generalizability/robustness compared with a single estimator. In ensemble averaging methods such as the random forest method used here (in which a forest of diverse randomized decision trees is aggregated for decision making), the driving principle is to first build several estimators independently and then average their predictions (for a classification problem, the predicted class of a given input is determined by voting among the trees in the forest; the predicted class is the one with the highest mean probability estimate across all trees). On average, such a combined estimator is usually better than any single base estimator because its variance is reduced. An advantage of a random forest ensemble is that it can be trained in parallel; hence, in this experiment, the random forest ensemble was parallelized. The number of trees (estimates) was specified as 55; this number was determined experimentally. To tune/optimize such a meta-estimator, a grid search can be performed by exhaustively generating all possible parameter combinations from considered parameter values and specifying these combinations through manual intervention. In this experiment, *train_test_split()* [43] was used to split the entire feature dataset into two disjoint subsets: a training dataset (70%) and a test dataset (30%), where all data examples were shuffled before they were split.

To fully evaluate the effectiveness of the random forest ensemble in load identification for energy decomposition, this work examines both precision and recall to compute the F_1 score. Precision and recall are often in tension (improving precision typically reduces recall, and vice versa [23]). Table 2 shows the load classification results of the random forest ensemble for comparison with the ensemble in [12]. As shown in Table 2, the accuracy achieved by the random forest ensemble is 93.05% (note that in [12], accuracy alone does not tell the full story). In Table 3, the random forest ensemble is compared against the ensemble in [12] and the ensemble in [23]. As reported in Table 3, the accuracy of the ensemble in this work is improved by 2.54% and 1.59%, which is relative to the accuracy of 90.51% achieved by the ensemble in [12] and the accuracy of 91.46% achieved by the ensemble in [23], respectively. Notably, the bagging-decision-tree-based classification approach used in [23], which improves load classification by combining individual diverse decision trees into a meta-estimator/ensemble, is also superior to the ensemble in [12].

Next, we will show the feasibility of using the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism based on the energy decomposition results obtained above to realize a smart home automation environment. In this experiment, the ${x_{t,w}}^{i,d}$ values in Eq. (3) for computing the URs of each monitored electrical home appliance considered for smart home automation/load forecasting were chronologically weighted with weights of

TABLE 1. Monitored electrical home appliances in the house environment in [12] and [23].

electrical appliance	rated power (kW)
electric rice cooker	1.10
electric water boiler *	0.90
steamer *	0.80
TV *	0.22
range hood *	0.14
PC *	0.35
hair dryer	1.20
washing machine	0.30

TABLE 2. Load classification results obtained using the random forest ensemble with RFE-based feature selection in this experiment.

load class	precision	recall	F_1 score
1	0.98	0.93	0.96
2	0.97	0.98	0.97
3	1.00	0.90	0.95
4	0.71	0.16	0.26
5	0.90	0.86	0.88
6	0.91	0.98	0.94
7	0.82	0.90	0.86
8	0.60	0.64	0.62
9	0.64	0.90	0.75
10	0.86	0.96	0.91
11	0.86	0.99	0.92
12	0.87	0.76	0.81
13	1.00	0.36	0.53
14	1.00	0.98	0.99
15	1.00	0.67	0.80
16	0.56	0.20	0.29
17	0.65	0.43	0.52
accuracy (%)		93.05	

$\tau_w = [0.1, 0.2, 0.3, 0.4]$ for $\boldsymbol{W} = \{1, 2, 3, 4(= W)\}$ (the routines were observed for 4 weeks), which satisfy Eq. (4), and the computed time-series UR data were normalized and modeled for time-series load forecasting.

The autoregressive MLP model depicted in Fig. 5, which was used to learn the URs of the individual monitored electrical home appliances, was configured through trial and error as specified below.

- Number of hidden layers (L_h in Fig. 5): 3.
- Numbers of artificial neurons in hidden layers: [100, 50, 15].
- Activation function: the ReLU function.
- Weight optimizer: the stochastic gradient-based optimizer proposed in [34] (Adam).
- Strategy for preventing overtraining: dropout. Within a layer, dropout consists of randomly ''dropping out'' a number of output features from that layer during training, where the ''dropout rate'' represents the proportion of the output features that are zeroed out. Here, the dropout rate was set to 0.1.
- Loss function: mean squared error (MSE).
- Learning rate: 0.002.
- Maximum number of epochs: 500.

A grid search that exhaustively considers all hyperparameter combinations can be conducted and used to perform hyperparameter tuning to tune/optimize the model.

In this experiment, the training process of the model was accelerated by a GPU. The model to be reduced for its computation time for learning was programmed in Keras [44]—a high-level application programming interface (API) for ANNs in TensorFlow™based on Python—and the programmed model was executed (i.e., trained) on an NVIDIA® Tesla® K80 GPU, specified and provided by Colab (Google Colaboratory). Colab allows us to program and execute our ANNs with zero configuration and free access to the GPU. The amount of GPU memory provided for use was 12 GB. Here, we mainly used *tf.keras.Sequential()*, which can stack network layers, for model building. In addition, the *model.compile()* function of Keras was used to configure and compile the model to be trained. Finally, the *model.fit()* function of Keras was used to adjust the model's trainable parameters while minimizing the MSE loss given the prepared input–output training data pairs. Specifically, we took the computed and normalized time-series URs from a monitored electrical home appliance considered for time-series load modeling and forecasting and divided the observed data sequence into input–output data pairs, where 15 time steps were treated as the input to the model (*n_steps_in*, or *n* in Fig. 5, =15) and 5 time steps were treated as the output (corresponding to the associated input) of the model (*n_steps_out*, or *m* in Fig. 5, =5) to learn a function of the following form:

$$f(\cdot) : R^{n\_steps\_in} \rightarrow R^{n\_steps\_out}.$$

To *n_steps_in* and *n_steps_out* to be specified, according to DR programs, 5-minute settlement provides significant opportunities for consumers to participate in the market.

In this experiment, the multiprocessing package [45] was also used. This package is a standard Python library that supports spawning processes using one of the Pool class's parallelization methods, such as *starmap()* for synchronous execution or *starmap_async()* for asynchronous execution. With the multiprocessing package imported, a pool of parallel workers can be configured to run tasks in parallel for parallel processing and used to asynchronously parallelize the training processes for building a set of GPU-accelerated neurocomputing-based load behavior learners to predict the load behaviors of the individual monitored electrical home appliances considered for smart home automation/time-series load forecasting. As illustrative examples, Fig. 8 shows the training trajectories (i.e., the loss curves) for the models configured, trained, and used for time-series load forecasting for the monitored steamer and TV. Fig. 9 shows the results of the TV model as an illustrative example of a load behavior

TABLE 3. Comparison of load identification results between the random forest ensemble in this work and the ensemble methods in [12] and [23].

ensemble method	accuracy (%)	accuracy improvement (%)
[12]*	90.51	-
[23]	91.46	1.59
this work	93.05	2.54

* benchmark

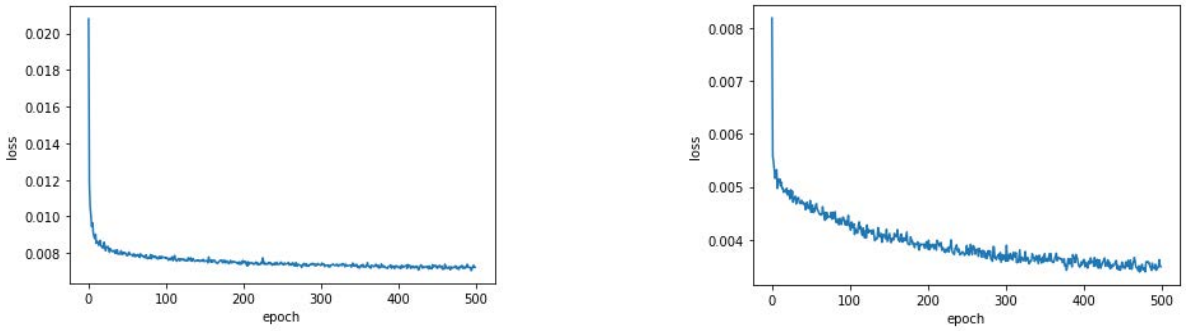


FIGURE 8. Illustrative examples of the training trajectories of models configured and trained as load behavior learners for two monitored electrical home appliances: (a) a steamer and (b) a TV.

learner trained on the computed, normalized and trended URs reflecting the on/off routines of an appliance over 5 weekdays (Monday through Friday, $T = 7200$). The illustrative example of the well-trained TV model as a load behavior predictor (for the on/off operation routines over 5 weekdays from Monday through Friday) is further detailed in Fig. 10. Specifically, Fig. 10(a) illustrates the comparison of the predicted URs against the actual trended data computed via Eq. (3) and normalized. Fig. 10(b) illustrates the corresponding on/off operation status, where if the predicted UR is greater than or equal to a certain threshold (0.5 here), then the operation status is considered to be ''on''; otherwise, it is considered to be ''off''. Based on such results, the total hourly power consumption in kilowatt-hours (kWh) of an electrical home appliance (one of the N (=5 here) monitored appliances) can be estimated/computed as shown in Eq. (10).

$$P_i^{rated} \times \sum_{t=1+60(h-1)}^{60+60(h-1)} \hat{x}_i(t) \Big/ 60 \tag{10}$$

In Eq. (10), $h \in \boldsymbol{H}$.

The ROC curves of the load behavior predictors were also drawn using the Python codes in [40] for further model evaluation. These curves are presented in Fig. 11, where the AUCs are also shown.

To fully evaluate the model performance for time-series load modeling and forecasting, accuracy alone is not sufficient for evaluating a model learning form a class-imbalanced dataset with a significant disparity between positives (appliance status ''on'' here) and negatives (appliance status ''off''). Therefore, in addition to the ROC curves based on the TPR and FPR as defined in Eqs. (8) and (9), we compute and report the precision, recall and F_1 score. Table 4 reports

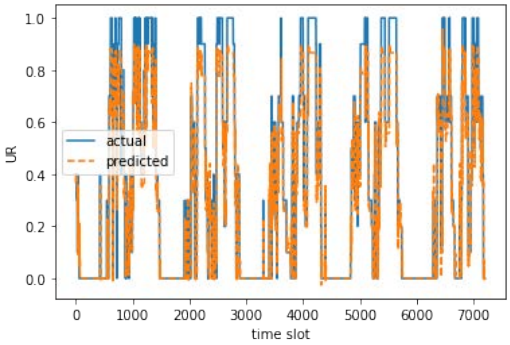
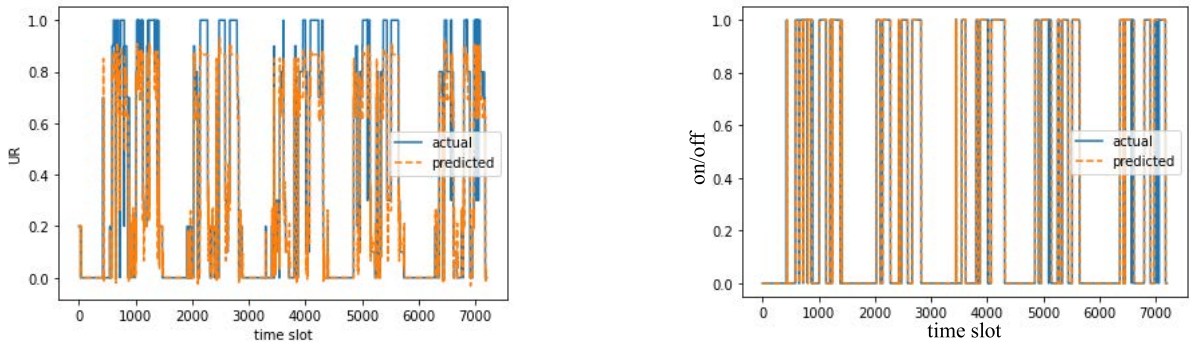


FIGURE 9. Results of the TV model as an illustrative example of a trained load behavior learner for a monitored appliance.

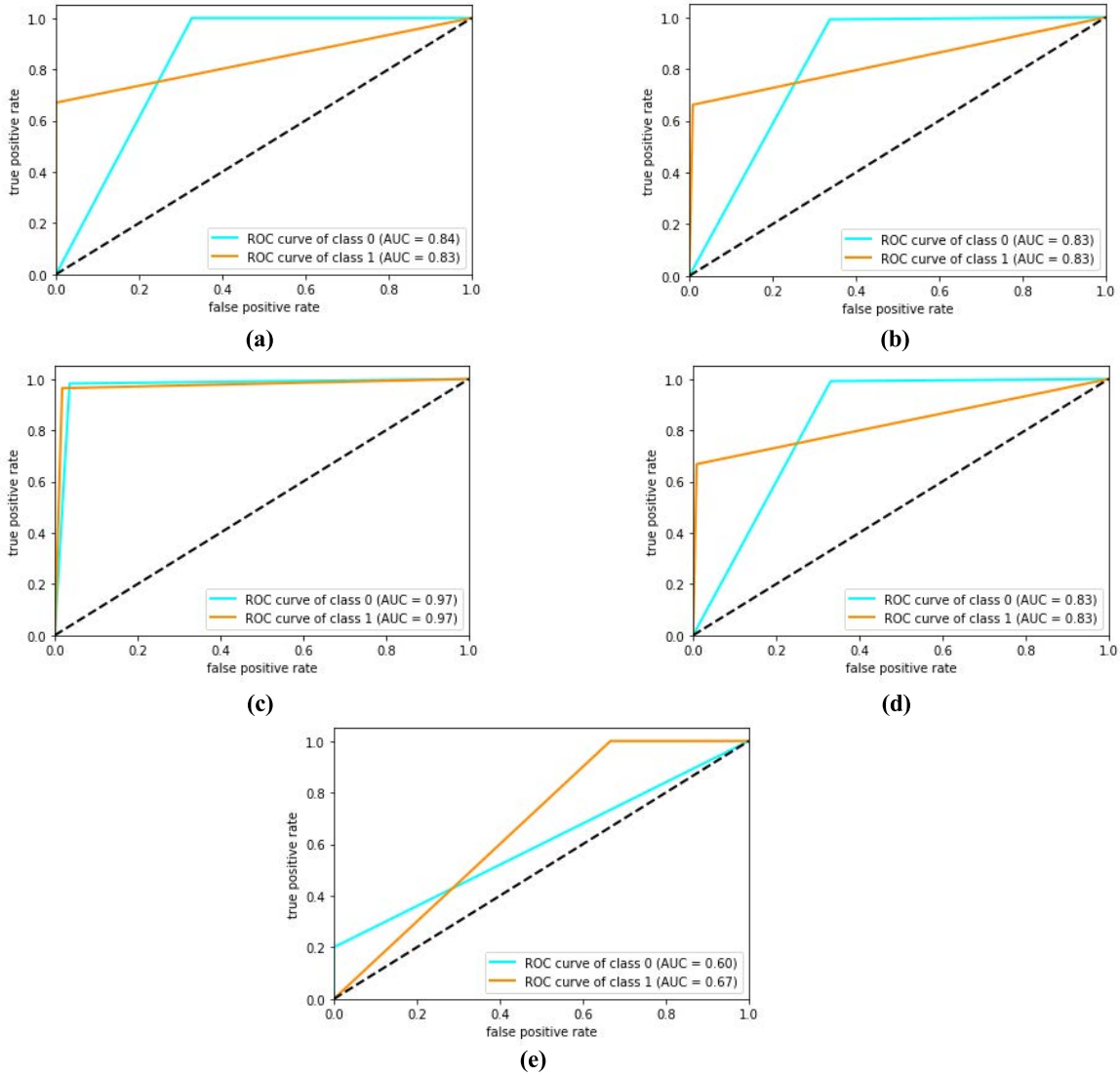
the load forecasting results of the autoregressive MLP-based load behavior predictors evaluated in terms of the different metrics. As computed from Table 4, average F_1 scores of 0.79 and 0.84 are obtained for class 0 (appliance status ''off'') and class 1 (appliance status ''on''), respectively, across all the relevant appliances (note that the PC was always ''on''), and an average value of the weighted average F_1 score of 0.98 is also obtained. As reported in Table 4, these load behavior predictors based on this benchmark model are able to learn from and forecast the on/off operation routines of the appliances with an acceptable level of performance in multistep time-series load modeling and forecasting.

In this work, an LSTM, the stacked LSTM model depicted in Fig. 7, is also evaluated in comparison with the autoregressive MLP-based model for solving the same multistep time-series load modeling and forecasting problem. Here,



(a) (b)

FIGURE 10. Results of the TV model as an illustrative example of a load behavior predictor: (a) a comparison of predicted and actual URs and (b) a comparison of predicted and actual on/off operation statuses for smart home automation/time-series load forecasting.



(a) (b) (c) (d) (e)

FIGURE 11. ROC curves of the load behavior predictors: (a) the electric water boiler model, (b) the steamer model, (c) the TV model, (d) the range hood model, and (e) the PC model.

the model used to learn from the URs of the individual monitored appliances was configured experimentally as follows. The stacked LSTM model consisted of a visible input layer with 15 (=*n_steps_in*) inputs; 4 hidden LSTM layers where 100 hidden units were used each; and an output (dense) layer producing 5 (=*n_steps_out*) outputs for

TABLE 4. Load forecasting results of the autoregressive MLP-based load behavior predictors evaluated in terms of accuracy, precision, recall and F_1 score.

load model		metrics			
		accuracy	precision	recall	F_1 score
electric water boiler	off	1.00	1.00	1.00	1.00
	on		0.99	0.67	0.80
	weighted avg	-	1.00	1.00	0.99
steamer	off	0.97	0.98	0.99	0.99
	on		**0.81**	0.66	0.73
	weighted avg	-	0.97	0.97	0.97
TV	off	0.96	0.97	0.97	0.97
	on		0.95	0.95	0.95
	weighted avg	-	0.96	0.96	0.96
range hood	off	0.98	0.98	0.99	0.99
	on		**0.78**	0.67	0.72
	weighted avg	-	0.98	0.98	0.98
PC	off	1.00	0.00	0.00	0.00
	on		1.00	1.00	1.00
	weighted avg	-	1.00	1.00	1.00

TABLE 5. Load forecasting results of the stacked LSTM-based load behavior predictors evaluated in terms of accuracy, precision, recall and F_1 score.

load model		metrics			
		accuracy	precision	recall	F_1 score
electric water boiler	off	1.00	1.00	1.00	1.00
	on		**1.00**	0.67	0.80
	weighted avg	-	1.00	1.00	0.99
steamer	off	0.97	0.98	0.99	0.99
	on		0.76	**0.72**	**0.74**
	weighted avg	-	0.97	0.97	0.97
TV	off	0.96	0.97	0.97	0.97
	on		0.95	0.95	0.95
	weighted avg	-	0.96	0.96	0.96
range hood	off	0.98	**0.99**	0.99	0.99
	on		0.74	**0.74**	**0.74**
	weighted avg	-	0.98	0.98	0.98
PC	off	1.00	0.00	0.00	0.00
	on		1.00	1.00	1.00
	weighted avg	-	1.00	1.00	1.00

a forecast. The activation functions used in all the hidden LSTM layers were of the default type (tanh). In practice, based on available runtime hardware and constraints, the LSTM layer to be stacked, specified and trained will choose a fast NVIDIA® CUDA Deep Neural Network library (cuDNN)-based implementation or a pure TensorFlow implementation to train the model. If a GPU is available and the layer meets the criteria of using the cuDNN implementation, the layer will be parallelized for training (i.e., the model will be trained on the GPU to speed up the training

TABLE 6. Five-day electricity demand, in kWh, estimated by the autoregressive MLP-based load behavior predictors for the 5 individual monitored electrical home appliances.

estimated electricity demand (kWh)	electric water boiler	steamer	TV	range hood	PC
actual	1.53	5.07	9.36	0.73	18.23
forecasted	1.03	4.35	9.35	0.60	18.24
mean absolute percentage error (MAPE) (%) *	32.68	14.21	0.11	17.81	0.06

* Less than 10%: highly accurately estimated; 10–20%: well estimated; 20–50%: reasonably estimated; and greater than 50%: inaccurately estimated.

TABLE 7. Five-day electricity demand, in kWh, estimated by the stacked LSTM-based load behavior predictors for the 5 individual monitored electrical home appliances.

estimated electricity demand (kWh)	electric water boiler	steamer	TV	range hood	PC
actual	1.53	5.07	9.36	0.73	18.23
forecasted	1.05	4.71	9.37	0.70	18.24
MAPE (%)	**31.37**	**7.11**	0.12	**3.85**	0.06

TABLE 8. Computation time by GPU-accelerated autoregressive MLP-based load behavior learners trained in serial processing vs. parallel processing.

AI methodology implemented	elapsed time (mins.)	improvement in computation time (%)
GPU-accelerated autoregressive MLP-based load behavior learners trained in *serial* processing	20.15	-
GPU-accelerated autoregressive MLP-based load behavior learners trained in *parallel* processing *	16.22	+19.50

* A total of 2 parallel workers were used for the 5 GPU-accelerated autoregressive MLP-based load behavior learners for the monitored electrical home appliances of interest for smart home automation/time-series load forecasting. The GPU memory usage was 1.56 GB.

process) [44]. For the model used in this evaluation, the kernel, recurrent kernel and bias regularizers were based on an L2 regularization penalty, where the regularization factors were set to 0.001. Finally, the same weight optimizer (Adam) was used here as was used for the autoregressive MLP-based model, and the learning rate was also set to 0.002. Table 5 reports the load forecasting results of the stacked LSTM-based load behavior predictors evaluated in terms of the same evaluation metrics. As computed from Table 5, average F_1 scores of 0.79 and 0.85 are obtained for class 0 and class 1, respectively, across all the relevant appliances, and an average value of the weighted average F_1 score of 0.98 is also obtained. As reported in Table 5, the load behavior predictors based on this model considered for comparison are also capable of learning from and forecasting the on/off operation routines of the appliances with a satisfactory level of performance in multistep time-series load modeling and forecasting. In fact, as seen by comparing the results shown in Tables 4 and 5, the LSTM model slightly outperforms the autoregressive MLP model. Table 6 shows the five-day electricity demand, in kWh, forecasted by the autoregressive MLP-based load behavior predictors, and Table 7 shows the five-day electricity demand (in kWh) forecasted by the stacked LSTM-based load behavior predictors. As shown in Tables 6 and 7 and compared with the estimates obtained in [23], overall, the electricity demand estimates are precisely forecasted. Hence, the load forecasting performance is improved in this work. Moreover, as shown in Table 7 and compared from Table 6 showing the five-day electricity demand forecasting results of the autoregressive MLP-based load behavior predictors, overall, the electricity demand estimates are more precisely forecasted by the stacked LSTM-based load behavior predictors.

Table 8 reports the computation time by the GPU-accelerated autoregressive MLP-based load behavior learners trained in serial processing vs. parallel processing.

TABLE 9. Computation time by GPU-accelerated stacked LSTM-based load behavior learners trained in serial processing vs. parallel processing.

AI methodology implemented	elapsed time (mins.)	improvement in computation time (%)
GPU-accelerated stacked LSTM-based load behavior learners trained in *serial* processing	44.34	-
GPU-accelerated stacked LSTM-based load behavior learners trained in *parallel* processing *	36.58	+17.50

* A total of 2 parallel workers were used for the 5 GPU-accelerated stacked LSTM-based load behavior learners for the monitored electrical home appliances of interest for smart home automation/time-series load forecasting. The GPU memory usage was 2.45 GB.

In total, 2 parallel workers were used to arrive at the results, and the GPU memory usage was 1.56 GB. As reported in Table 8, a 19.50% improvement in computation time was achieved through parallel processing. Similarly, Table 9 reports the computation time by the GPU-accelerated stacked LSTM-based load behavior learners trained in serial processing vs. parallel processing. As reported in Table 9, the time improvement achieved through parallelization was 17.50%, where 2 parallel workers were used and the GPU memory usage of 2.45 GB was needed.

As demonstrated in this section, the effectiveness of the presented parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism based on energy decomposition has been confirmed.

IV. CONCLUSION

Collecting and further analyzing consumers' electrical energy consumption data from smart meters can enable several useful consumer-centric use cases, such as home automation. Because the electrical energy consumption patterns mined from smart meter data are indicative of residents' daily life, it is possible to develop a smart home automation approach based on energy decomposition. To this end, an SHEMS utilizing a parallel-processing-implemented, GPU-accelerated neurocomputing-based time-series load modeling and forecasting mechanism based on energy decomposition has been proposed and demonstrated in this work for smart home applications. In this work, the energy decomposition process has been improved and further investigated as a basis for time-series load modeling and forecasting for smart home automation. A mechanism for time-series load modeling and forecasting facilitated by energy decomposition has been developed here for modeling and forecasting the load behavior of relevant electrical home appliances based on parallel-processing-implemented, GPU-accelerated neurocomputing to predict the appliances' operational routines. The effectiveness of the presented work has been confirmed. The presented work is different from other recent solutions for home automation involving home security and healthcare; other recent solutions depend on the installation of sensors and/or other devices such as video cameras that have high costs for installation and annual maintenance. Three future research directions are drawn below. The first is to investigate a home controller based on the NVIDIA® Jetson Nano™ Developer Kit—a small, compact and powerful embedded system that can run multiple DNNs in parallel for applications such as the smart home application realized in this work—to support the future implementation of the SHEMS presented in this work within an edge–cloud collaborative paradigm. To achieve automatic billing and energy management services, the microcontroller units (MCUs) driving current smart meters employ high-precision analog front-end (AFE) circuits and sophisticated data processing units. The basic components of a smart meter's AFE include metering equipment, analog-to-digital converters (ADCs) and algorithms used by the MCU acting as the meter's host processor to interpret the acquired raw data. In a smart grid, smart meters are required to measure power factor and/or monitor power quality in most industrial and many commercial field applications, although it is typically not required in residential sectors. Therefore, the sampling rate of a smart meter's ADCs should be sufficient to comply with the Nyquist theorem for the types and analysis purposes of the measurements the meter is required to make. In the case of harmonic analysis via the FFT for power quality monitoring, the sampling rate should be at least twice the fundamental frequency multiplied by the order of the harmonics of interest. For 60-Hz electric distribution, a sampling rate of 2–4 kHz is usually sufficient, as this is sufficient for, for instance, common line perturbations related to power quality monitoring. The high-rate sampling capability used to monitor line perturbations can also be used to monitor P and Q to improve the meter's accuracy. Therefore, the second future research direction is to consider the possibility of acquiring the composite electricity current and voltage signals required in this work from the meter with which the SHEMS communicates with it; the work done here can be a value-added user-centric service provided by utilities to

customers. In [46], the hybrid load forecasting scheme based on characteristic load decomposition (CLD) by pilot signals for a mixed-use complex having clusters by distinguishable commercial, residential and industrial loads is proposed, which is developed from the perspective of building-level load forecasting. The third future research direction is to extend the work done here from the appliance-level load forecasting perspective to improve the capabilities of the above scheme. This future work will help the CLD scheme precisely target the representative pilot loads of each cluster (in [46], it is assumed that a larger building size corresponds to higher typical power consumption, and loads with the widest gross area in a mixed-use complex are considered as representative pilot loads).

REFERENCES

[1] A. M. Pirbazari, M. Farmanbar, A. Chakravorty, and C. Rong, "Short-term load forecasting using smart meter data: A generalization analysis," *Processes*, vol. 8, no. 4, p. 484, Apr. 2020.
[2] S. Y. Chen and Z. L. Hu, "A smart grid-based home energy aware system," *Sens. Mater.*, vol. 29, pp. 1513–1522, 2017.
[3] T. L. Chen, T. C. Kang, C. Y. Chang, T. C. Hsiao, and C. C. Chen, "Smart home power management based on Internet of Things and smart sensor networks," *Sens. Mater.*, vol. 33, pp. 1687–1702, 2021.
[4] M. Jin, S. Liu, S. Schiavon, and C. Spanos, "Automated mobile sensing: Towards high-granularity agile indoor environmental quality monitoring," *Building Environ.*, vol. 127, pp. 268–276, Jan. 2018.
[5] B. Dong, D. Yan, Z. Li, Y. Jin, X. Feng, and H. Fontenot, "Modeling occupancy and behavior for better building design and operation—A critical review," *Building Simul.*, vol. 11, no. 5, pp. 899–921, Oct. 2018.
[6] W. L. Hsu, W. T. Chen, H. H. Kuo, Y. C. Shiau, T. Y. Chern, S. C. Lai, and W. H. Fan, "Establishment of smart living environment control system," *Sens. Mater.*, vol. 32, pp. 183–195, Jan. 2020.
[7] W. L. Hsu, H. H. Tsai, M. L. Yang, S. C. Lai, M. C. Ho, and Y. C. Shiau, "Development of smart residential environment control system," *Sens. Mater.*, vol. 33, pp. 3361–3377, Jan. 2021.
[8] Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "Building power consumption datasets: Survey, taxonomy and future directions," *Energy Buildings*, vol. 227, Nov. 2020, Art. no. 110404.
[9] A. Sayed, Y. Himeur, A. Alsalemi, F. Bensaali, and A. Amira, "Intelligent edge-based recommender system for internet of energy applications," *IEEE Syst. J.*, vol. 16, no. 3, pp. 5001–5010, Sep. 2021.
[10] B. Völker, A. Reinhardt, A. Faustine, and L. Pereira, "Watt's up at home? Smart meter data analytics from a consumer-centric perspective," *Energies*, vol. 14, no. 3, p. 719, Jan. 2021.
[11] A. Tundis, A. Faizan, and M. Mühlhäuser, "A feature-based model for the identification of electrical devices in smart environments," *Sensors*, vol. 19, no. 11, p. 2611, Jun. 2019.
[12] Y. H. Lin and M. S. Tsai, "An advanced home energy management system facilitated by nonintrusive load monitoring with automated multiobjective power scheduling," *IEEE Trans. Smart Grid*, vol. 6, no. 4, pp. 1839–1851, Jul. 2015.
[13] Y.-H. Lin and Y.-C. Hu, "Residential consumer-centric demand-side management based on energy disaggregation-piloting constrained swarm intelligence: Towards edge computing," *Sensors*, vol. 18, no. 5, p. 1365, Apr. 2018.
[14] K. K. Radhakrishnan, H. D. Chinh, M. Gupta, S. K. Panda, and C. J. Spanos, "Context-aware plug-load identification toward enhanced energy efficiency in the built environment," *IEEE Trans. Ind. Appl.*, vol. 56, no. 6, pp. 6781–6791, Nov. 2020.
[15] D. A. M. Lemes, T. W. Cabral, G. Fraidenraich, L. G. P. Meloni, E. R. De Lima, and F. B. Neto, "Load disaggregation based on time window for HEMS application," *IEEE Access*, vol. 9, pp. 70746–70757, 2021.
[16] M. Lu and Z. Li, "A hybrid event detection approach for non-intrusive load monitoring," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 528–540, Jan. 2020.
[17] T.-T.-H. Le, H. Kang, and H. Kim, "Household appliance classification using lower odd-numbered harmonics and the bagging decision tree," *IEEE Access*, vol. 8, pp. 55937–55952, 2020.
[18] P. R. Z. Taveira, C. H. V. De Moraes, and G. Lambert-Torres, "Non-intrusive identification of loads by random forest and fireworks optimization," *IEEE Access*, vol. 8, pp. 75060–75072, 2020.
[19] P. Franco, J. M. Martinez, Y.-C. Kim, and M. A. Ahmed, "IoT based approach for load monitoring and activity recognition in smart Homes," *IEEE Access*, vol. 9, pp. 45325–45339, 2021.
[20] Y. Himeur, A. Alsalemi, F. Bensaali, A. Amira, and A. Al-Kababji, "Recent trends of smart non-intrusive load monitoring in buildings: A review, open challenges and future directions," *Int. J. Intell. Syst., Early Access*, vol. 37, pp. 1–56, Oct. 2022.
[21] E. McKenna, I. Richardson, and M. Thomson, "Smart meter data: Balancing consumer privacy concerns with legitimate applications," *Energy Policy*, vol. 41, pp. 807–814, Feb. 2012.
[22] L. G. Fahad and S. F. Tahir, "Activity recognition and anomaly detection in smart Homes," *Neurocomputing*, vol. 423, pp. 362–372, Jan. 2021.
[23] Y.-H. Lin, "An advanced smart home energy management system considering identification of ADLs based on non-intrusive load monitoring," *Electr. Eng.*, vol. 104, no. 5, pp. 3391–3409, Oct. 2022.
[24] M. A. Devlin and B. P. Hayes, "Non-intrusive load monitoring and classification of activities of daily living using residential smart meter data," *IEEE Trans. Consum. Electron.*, vol. 65, no. 3, pp. 339–348, Aug. 2019.
[25] M. Amayri, S. Ploix, H. Kazmi, Q. D. Ngo, and E. A. E. Safadi, "Estimating occupancy from measurements and knowledge using the Bayesian network for energy management," *J. Sensors*, vol. 2019, Apr. 2019, Art. no. 7129872.
[26] J. Wang, N. Spicher, J. M. Warnecke, M. Haghi, J. Schwartze, and T. M. Deserno, "Unobtrusive health monitoring in private spaces: The smart home," *Sensors*, vol. 21, no. 3, p. 864, Jan. 2021.
[27] A. Ruano, A. Hernandez, J. Ureña, M. Ruano, and J. Garcia, "NILM techniques for intelligent home energy management and ambient assisted living: A review," *Energies*, vol. 12, no. 11, p. 2203, Jun. 2019.
[28] Y.-C. Hu, Y.-H. Lin, and C.-H. Lin, "Artificial intelligence, accelerated in parallel computing and applied to nonintrusive appliance load monitoring for residential demand-side management in a smart grid: A comparative study," *Appl. Sci.*, vol. 10, no. 22, p. 8114, Nov. 2020.
[29] Y.-H. Lin, "Trainingless multi-objective evolutionary computing-based nonintrusive load monitoring: Part of smart-home energy management for demand-side management," *J. Building Eng.*, vol. 33, Jan. 2021, Art. no. 101601.
[30] *Scikit-Learn: Machine Learning in Python*. Accessed: Mar. 23, 2022. [Online]. Available: https://scikit-learn.org
[31] *TensorFlow*. Accessed: Mar. 28, 2022. [Online]. Available: https://www.tensorflow.org/
[32] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Mach. Learn.*, vol. 46, nos. 1–3, pp. 389–422, 2002.
[33] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[34] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.
[35] S.-H. Noh, "Performance comparison of CNN models using gradient flow analysis," *Informatics*, vol. 8, no. 3, p. 53, Aug. 2021.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[38] D. Dheda, L. Cheng, and A. M. Abu-Mahfouz, "Long short term memory water quality predictive model discrepancy mitigation through genetic algorithm optimisation and ensemble modeling," *IEEE Access*, vol. 10, pp. 24638–24658, 2022.
[39] L. Matindife, Y. Sun, and Z. Wang, "Image-based mains signal disaggregation and load recognition," *Complex Intell. Syst.*, vol. 7, no. 2, pp. 901–927, 2021.
[40] S. Zhai, H. Zhou, Z. Wang, and G. He, "Analysis of dynamic appliance flexibility considering user behavior via non-intrusive load monitoring and deep user modeling," *CSEE J. Power Energy Syst.*, vol. 6, no. 1, pp. 41–51, Mar. 2020.

[41] P. Franco, J. M. Martinez, Y.-C. Kim, and M. A. Ahmed, "A framework for IoT based appliance recognition in smart Homes," *IEEE Access*, vol. 9, pp. 133940–133960, 2021.
[42] *Receiver Operating Characteristic (ROC)—Scikit-Learn*. Accessed: Jun. 2, 2022. [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
[43] *Scikit-Learn: Machine Learning in Python*. Accessed: Apr. 15, 2022. [Online]. Available: https://scikit-learn.org/stable/
[44] *Keras: The Python Deep Learning API*. Accessed: Jul. 12, 2022. [Online]. Available: https://keras.io/
[45] *Multiprocessing—Process-Based Parallelism*. Accessed: Jul. 12, 2022. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html
[46] K. Park, S. Yoon, and E. Hwang, "Hybrid load forecasting for mixed-use complex based on the characteristic load decomposition by pilot signals," *IEEE Access*, vol. 7, pp. 12297–12306, 2019.

YU-HSIU LIN received the Ph.D. degree in mechanical and electrical engineering from the Graduate Institute of Mechanical and Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan, in 2014. Since February 2021, he has been with the Graduate Institute of Automation Technology, National Taipei University of Technology, where he is currently an Assistant Professor. Regarding his work and research experience, from October 2014 to August 2017, he worked with the Smart Network System Institute and the Institute for Information Industry (III), Taiwan, where he worked as a full-time Senior Engineer. From September 2017 to July 2018, he was an Assistant Professor with the Department of Computer Science and Information Management, Providence University, Taichung City, Taiwan. From August 2018 to July 2019, he was also an Assistant Professor with the Department of Electrical Engineering, Southern Taiwan University of Science and Technology, Tainan City, Taiwan. Then, he was an Assistant Professor with the Department of Electrical Engineering, Ming Chi University of Technology, New Taipei City, Taiwan, from August 2019 to January 2021. His current research interests include the AIoT (AI across IoT) technologies, fog/edge–cloud computing, and artificial intelligence/machine learning/deep learning/computational intelligence tools developed for and applied in smart-grid-related research.

HUEI-SHENG TANG is currently pursuing the M.S. degree in computer science and information engineering with the National Ilan University, Yilan, Taiwan. His current research interests include multimedia (computer vision) and AI technology, including data mining, machine learning and deep learning, and applied in smart grid applications, such as smart home energy management (residential demand-side management), smart home automation, and smart home healthcare.

TING-YU SHEN is currently pursuing the M.S. degree in automation technology from the National Taipei University of Technology, Taipei, Taiwan. His current research interests include the AIoT (AI across IoT) technologies, including edge–cloud computing and computational intelligence and developed for and applied in smart-grid-related disciplines, such as demand-side management.

CHIH-HSIEN HSIA (Member, IEEE) was born in Taipei, Taiwan, in 1979. He received the Ph.D. degree from Tamkang University, New Taipei, Taiwan, in 2010. In 2007, he was a Visiting Scholar with Iowa State University, Ames, IA, USA. From 2010 to 2013, he was a Postdoctoral Research Fellow with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei. From 2013 to 2015, he was an Assistant Professor with the Department of Electrical Engineering, Chinese Culture University, Taiwan. He was an Associate Professor with Chinese Culture University and the National Ilan University, Taiwan, from 2015 to 2017, where he was the Director of the Research Planning Division, Research & Development, from 2019 to 2020. He is currently a Professor and the Chairperson of the Department of Computer Science and Information Engineering, National Ilan University, where he is the Director of the Multimedia & Intelligent Technical Laboratory. He received the Outstanding Young Scholar Award of the Taiwan Association of Systems Science and Engineering, in 2020, and the Outstanding Young Scholar Award of the Computer Society of the Republic of China, in 2018. His research interests include digital signal processing (DSP) integrated circuit (IC) design, AI in computer vision, and cognitive learning.

He is the Chapter Chair of the IEEE Young Professionals Group, Taipei Section, and the Director of the IET Taipei Local Network. He has served as an Associate Editor of the *Journal of Imaging Science and Technology*, the *Journal of Imaging*, and the *Journal of Computers*.