Received November 6, 2020, accepted November 25, 2020, date of publication December 1, 2020,
date of current version December 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3041442

Combined Prediction Energy Model at Software Architecture Level

JUNKE LI[1], (Member, IEEE), KAI LIU[1], MINGJIANG LI[1], AND DEGUANG LI[2], (Member, IEEE)
[1]School of Computer and Information, Qiannan Normal University for Nationalities, Duyun 558000, China
[2]School of Information, Luoyang Normal University, Luoyang 471934, China

Corresponding author: Junke Li (ljk2006ljk@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61862051 and Grant 61802162, in part by the Science and Technology Foundation of Guizhou Province under Grant [2019]1447, in part by the Natural Science Foundation of Guizhou Education Department under Grant [2019]071, in part by the Nature Science Foundation of Qiannan under Grant [2018]05, and in part by the Qiannan Normal University for Nationalities under Grant QNSY2018BS012 and Grant QNSYRC201714.

ABSTRACT Accurate prediction of software energy consumption is of great significance for the sustainable development of the environment. In order to overcome the limitations of a single prediction method and further improve the prediction accuracy, a combined prediction energy model of adaboost algorithm and RBF (radial basis function) neural network at software architecture level is proposed. Firstly, three kinds of energy prediction models are established by polynomial regression, support vector machine and neural network respectively. Secondly, the RBF neural network is used to nonlinear combine the predicted values of the above three models. Finally, RBF integrated by adaboost algorithm is used as high-precision prediction of energy consumption. Experimental results show that the prediction accuracy of the combined prediction model is higher than that of the single model.

INDEX TERMS Green computing, energy consumption prediction, Adaboost algorithm, RBF neural network, combined prediction model.

I. INTRODUCTION

ICT (information and communication technology) industry is an important driving force for social development and world economic growth in the 21st century. Its energy consumption accounts for 10% of the total global power consumption [1]; its total carbon emissions reach 2% $\sim$ 2.5% of the total global carbon emissions, which is more obvious in developed countries, reaching 10% [2]. According to a research report from the European Union, if the temperature rise is to be controlled below 2°C in 2020, the carbon emission must be reduced by 15%-30% [3]. With the continuous investment in ICT industry in the world, its scale will continue to grow, and its energy consumption will also continue to grow. In order to promote the sustainable development of ICT industry, green computing [4], [5] has become the consensus of global researchers. To implement green computing, the first task is to measure energy consumption.

To measure the energy consumption, it is necessary to understand the execution process of the software running in the computer. In order to execute the software in computer system, the corresponding instructions (data processing and transmission instructions, program control instructions, input and output instructions, etc.) will be converted first, while the underlying hardware will change the state of different circuits to achieve the corresponding functions, and then generate different energy consumption. Based on this, from the perspective of software, the main function of software is to complete its corresponding functions, and in this process, it will generate additional "by-products" called energy consumption. From the perspective of energy consumption, the energy consumption is generated by hardware, and the amount is determined by software. In order to better understand, manage and control the energy of computer system, the producers, researchers and designers have carried out researches from many aspects. For example, at the bottom of the hardware, saving energy are conducted by improving the manufacturing process, optimizing the circuit structure. At instruction level, the instructions optimized by compiler, instruction conversion and rearrangement, and loop structure optimization [9]–[13] are used for energy management; in the source code layer, code expression change,

The associate editor coordinating the review of this manuscript and approving it for publication was Kathiravan Srinivasan.

data representation and program structure rearrangement are optimized, and redundant calculation is eliminated, and data storage space are compressed [14]–[20]; at process level, the process level energy management [21]–[23] are used; at architecture layer, macro-modeling, system structure selection, transformation and simplification [24]–[26] are used.

To manage and optimize the power consumption of software, the measurement and estimation of power consumption is the first work. At present, this work is mainly carried out at several levels. At the hardware level, the power consumption is measured by power meter and power sensor [6]–[8]; at the instruction level, the energy consumption is mainly obtained by statistical method, that is, the energy of software is obtained by accumulating the energy of each instruction or each kind of instructions, instruction pairs, pipeline or cache and other hardware structures during the execution [9]–[13]; at the source code level, the energy consumption of single line code is mainly obtained by linear regression, and then the energy consumption of software is obtained by accumulating the energy of each single line code [14]–[20]; at the process level, the relationship between the resource utilization rate and energy of the process is established, and then, it accumulates time to obtain energy consumption of process [21]–[23]; at the architecture level, the relationship between software high-level features and energy consumption is explored to predict software energy consumption [24]–[26]. Generally, software can be considered as a complex system, and then its architecture can be represented by complex network to study different characteristics of software [28]–[30]. Li *et al.* [27] have studied the power consumption model at architecture level from this point of view. Inspired by the above researches and related theories of power model, this article studies the accuracy of power model of complex network representation at software architecture level.

Although there has power model at architecture level, the prediction accuracy is still not ideal. Therefore it is necessary to do further research on improving the prediction accuracy. Therefore, this article proposes a combined prediction method based on adaboost idea and RBF (radial basis function) neural network. First of all, three single prediction models are optimized, including polynomial regression prediction model, SVR (support vector regression) prediction model and BP (back propagation neural network); then, RBF is used to optimize the prediction values of three single prediction models. Finally, according to the prediction results of each single prediction model and adaboost RBF algorithm, the weight coefficient of three single predictors is updated through multiple iterations; after getting the best weight coefficient, the output values of the three single predictors are fused by RBF and the best weight coefficient to get the high-precision prediction values at the architecture level. The experimental results show that the Root Mean Square Error of adaboost RBF power consumption model is 0.9362, which shows that our model is reasonable and effective. Thus our contributions in this article are:

(1) This article presents an architecture level software power model of adaboost RBF, which can estimate the software power consumption with small error and meet the requirements of high-level software power consumption modeling.

(2) We verify the adaboost RBF power model on the actual computer platform, and the results show that the model is reasonable and effective.

(3) Experiments show that software power consumption can be analyzed from a high level, which is of great significance to the architecture design of low-power software.

II. RELATED WORK

Researchers and scientists have proposed many software power models to achieve the energy. These methods are working at different levels, therefore, they can be divided into different categories as shown in table 1. The following is the brief introduction of these approaches.

A. DEVICE-BASED LEVEL

In this level, power are usually got by power meters [6], special designed power devices [7] and hardware with integrated power sensors [8]. The direct way to get power is using power meters, which uses the relationship between power consumption and voltage and current in Physics. This way is simple and it can give researchers the intuitive understanding the power dissipation of devices or the full system, but it is not convenient to control the process of measurement or difficulty to place the meter in space-limited device. Therefore, the integrated power sensor is invented to get the power more convenient and can be integrated into the device to monitor the power consumed and obtain power consumption data without additional power meter. Although we can get power information accurately by hardware, this method requires additional power meter and special operation knowledge to ordinary users.

B. INSTRUCTION LEVEL

The essence of software execution is instruction execution. Based on this simple idea, Tiwari *et al.* [9] has proposed the instruction-level model to get the power and gives the concept of software power consumption firstly. He points that total power consumption Ep of the program p has three parts, which is the base power cost of instruction, power consumption caused by circuit state switch between instructions, and the power consumption caused by other effects between instructions. Therefore, the theoretical basis of this power model can be expressed as equation (1).

$$E_p = \sum\nolimits_m B_m \times N_m + \sum\nolimits_{m,n} O_{m,n} \times N_{m,n} + \sum\nolimits_q E_q \quad (1)$$

In (1), p is the program; E_p is the energy consumption of program p; B_m and N_m are respectively the base power cost of instruction and execution time of the instruction m; $O_{m,n}$ and $N_{m,n}$ respectively represent the power consumption caused by circuit state switch between instruction m and

TABLE 1. Different approaches of energy measurement.

power model at different level				
Device-based level	Instruction level	Source code level	Process level	Architecture level
[6–8]	[9–13]	[14–20]	[21–23]	[24–27]

instruction n and the number of occurrences in the program; E_q is the power consumption in other case between instructions, such as pipeline stalls, Cache miss, etc., which are determined by corresponding hardware circuit. Especially, N_m and $N_{m,n}$ are determined by program execution path and can be obtained by dynamic analysis of the program. Equation (1) points out the energy composition of instruction level and defines the direction of instruction level energy consumption modeling. It is applied to two types of processors (intel 486dx2 and Fujitsu sparse 934). Based on equation (1), Bazzaz *et al.* [10], Tan *et al.* [11], Lee and Ro [12] and Ashouri *et al.* [13] have proposed fine-grained approach for power consumption analysis and prediction, and also put forward new methods for low power applications.

C. SOURCE CODE LEVEL

Previous studies at source code level overlook cache storage analysis and overheads due to concurrent program execution at runtime. From this view, Ahmad [14] has put forward an enhanced static-code-analysis-based lightweight energy estimation framework that has considered overheads associated with the application runtime execution environment, cache storage analysis, and the application inactivity period for energy estimation of applications. Ralph *et al.* [15] has showed that reducing the energy consumption of software systems though optimizations techniques such as code-change can efficiently improve the energy efficiency of software systems. Then, they propose joint code modifications which can produce a greater effect compared with that of accumulating modification operators with hard constraint and approximation. Shan *et al.* [16] has studied the offloading decision problem by code receiving and executing energy cost to determine each tasks whether to offload task data or load task code blocks. Banerjee *et al.* [17] has developed a framework that uses a combination of static and dynamic code analysis techniques to detect, validate and repair energy bugs in Android apps, which generates repair expressions to fix the validated energy bugs. Based on the trace of energy usage and the timestamps of programs execution events, Wei *et al.* [18] has presented a lightweight function-level profiling tool for measuring the energy consumption of program code called FPowerTool which measures energy by sampling the value of hardware counters built in the CPU and the sampled value is associated to the corresponding code segments by offline analysis. Li *et al.* [19] has analyzed the resources occupied by CUDA source program. Then, machine learning is used to correlate resources and energy consumption, and a source code level energy prediction model is proposed. Mukhanov *et al.* [20] has presented a novel fine-grained energy profiling tool based on probabilistic analysis for fine-grained energy accounting, which associates energy information effectively with source code at a fine-grained level.

D. PROCESS LEVEL

The energy estimation method at process level is mainly associates the process ID and the resource utilization rate with the energy, and obtains the energy of process through the cumulative time. Based on this idea, a process level power analysis tool, Energy guard, is proposed to obtain the running information, resources and utilization ratio of each process through the daemons running in the kernel space, and then provide the real-time information of the power consumed by each process [21], which is used to eliminate the energy consumed by the application's abnormal behavior. Colmant *et al.* [22] has focused on the energy estimation of VM-based systems and has proposed a fine-grained monitoring middleware called BitWatts. It provides real-time and power estimation of software processes running at any level of virtualization in the system, which can automatically learn an application-agnostic power model to estimate the power consumption of applications. The experiments demonstrates that BitWatts performs well both in number of monitored processes and virtualization levels. Lu and Yao [23] has presented a thread-voting dynamic voltage and frequency scaling (DVFS) technique for many core networks-on-chip to save energy which uses runtime performance indicatives from either network-level or thread-level to guide DVFS decisions. It allows each thread to ''vote'' for a V/F level in its own performance interest, and a region-based V/F controller make dynamic per-region V/F decision according to the major vote.

E. ARCHITECTURE LEVEL

Wu *et al.* [24] has proposed a generally applicable energy estimation methodology for accelerators that allows design specifications comprised of user-defined high-level compound components and user-defined low-level primitive components, which can be characterized by third-party energy estimation plug-ins. Jagroep *et al.* [25] has proposed an energy perspective on software architecture as a means to provide insight and enable analysis on the architectural elements that are the actual drivers behind the energy consumption. After getting potential quality attribute, they provides a means to quantify energy consumption aspects related to software. Horcas *et al.* [26] has pointed that different configurations of quality attributes in software architecture influenced energy efficiency. They represented the variability of quality attributes, as well as the energy efficiency and performance experiment results as a constraint satisfaction

problem to help software developers to build more energy efficient software. Li *et al.* [27] has explored architecture level software power consumption model through complex network, extracts software architecture level parameters and obtains software power consumption through a prediction model. Although the single prediction model can get the prediction results, its accuracy is still not ideal. Therefore, based on [27], this article improves the energy consumption prediction effect at the architecture level through the combination prediction model of Adaboost RBF method.

III. EXISTING POWER MODEL

The current research on predicting model can be divided into single predicting model and combined predicting model. They are described as follows.

A. SINGLE PREDICTING MODEL

The predicting method of energy described in related works section is mainly dependent on the single prediction method, and the specific description of this model is as follows.

1) POLYNOMIAL REGRESSION

The polynomial regression method is based on the premise that the data meet a certain fixed rule. It analyzes the historical data and finds the mathematical model to predict the data trend. Polynomial regression method is a commonly used model in prediction, which can be expressed as equation (2).

$$\hat{y} = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots b_nx_n \tag{2}$$

where $\hat{y}$ is the dependent variable; $x_1, x_2, x_3, \ldots, x_n$ are the independent variable; $b_1, b_2, b_3, \ldots, b_n$ is the regression coefficient, which are determined by the least square method; that is, selecting $b_1, b_2, b_3, \ldots, b_n$ minimizes equation (3).

$$\sum(y_{actual}-\hat{y})^2=\sum[y_{actual}-(b_0+b_1x_1+b_2x_2+b_3x_3+\cdots b_nx_n)]^2 \tag{3}$$

2) SUPPORT VECTOR MACHINE METHOD

Support vector machine (SVM) [31] is to realize the transformation between linear and nonlinear problems through spatial variable dimension mapping, and to solve the optimal hyperplane in high-dimensional feature space. SVR is the application of SVM in dealing with regression problems. For training samples $\{(x_1, y_1), (x_2, y_2), \ldots (x_i, y_i), \ldots (x_n, y_n)\}$, SVR is to find the function $f(x) = \omega^T x + b$, where x is the independent variable vector of the training sample set, ω is the optimal weight vector, b is the optimal bias, so that the difference between its output and the preset output y_i is less than or equal to ε, and the curve of function $f(x)$ is as stable as possible, that is to solve the following optimization problems in equation (4).

$$\begin{aligned} &\min \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &s.t.\ y_i - f(x_i) < \varepsilon + \xi_i \\ &f(x_i) - y_i < \varepsilon + \xi_i^* \\ &\xi_i \geq 0 \\ &\xi_i^* \geq 0 \end{aligned} \tag{4}$$

In equation (4), the constant C is the adjustment factor of the equilibrium $f(x)$ stationarity and the elastic range ε of the error condition; $f(x_i)$ is the optimal function value corresponding to the ith training sample set; ξ_i and ξ_i^* are the relaxation variables introduced to relax the error condition.

3) NEURAL NETWORK METHOD

BP neural network [32] is an artificial neural network model trained by error back propagation algorithm. BP neural network is composed of many neurons, which can express any complex nonlinear function. The training process of BP neural network includes the forward transmission of signal and the reverse transmission of error. When the signal is transmitted forward, the signal enters from the input layer and passes to the output layer after the nonlinear operation of multiple hidden layers. When the error is returned in the reverse direction, the network adjusts the network weight and threshold according to the deviation between the output value and the expected output value, so that the output tends to the expected value in the near future.

B. COMBINED ENERGY MOEDL

For the accuracy of single prediction method, the combined predicting methods have been studied gradually which are entropy weight combined prediction model and optimization algorithm-based combined model.

1) ENTROPY WEIGHT COMBINED PREDICTION MODEL

Entropy weight method is one of the representative combined prediction methods. Information entropy is a description of the degree of system irregularity. The idea is to determine the combination coefficient of each single model in the combined prediction model according to the degree of error variation of the single model prediction.

Take the absolute value of the error between the predicted value of a single prediction model at the current b sampling points and the actual energy consumption as the row vector, then the evaluation matrix $E = (e_{ij})_{a\times b}$ is obtained, where e_{ij} represents the prediction error at j time of the ith prediction model. Then the evaluation matrix E is normalized by row to get the normalized matrix $P = (P_{ij})_{a\times b}$, and the information entropy of the i prediction model can be calculated by equation (5).

$$H_i = -\frac{1}{\ln b}\sum_{j=1}^{b} p_{ij}\ln p_{ij} \tag{5}$$

The larger the information entropy of a single prediction model is, the less useful information it contributes to the combined prediction model, then its weight coefficient in the final model should be small, otherwise the larger the weight coefficient. Therefore, the weight of the single model

in the final combined prediction model can be calculated by equation (6).

$$w_i = \frac{1 - H_i}{a - \sum_{i=1}^{a} H_i} \tag{6}$$

2) OPTIMIZATION ALGORITHM-BASED COMBINED PREDICTION MODEL

Tian [33] has proposed a short-term traffic flow prediction approach based on combination model fusion. After getting the prediction results of improved extreme learning machine, seasonal auto regressive integrated moving average and auto regressive moving average, the fruit fly optimization algorithm is proposed to optimize the weight coefficient of the combination model. The fitness value of fruit fly optimization algorithm is chosen as the root square mean error (RMSE) between actual and prediction value which can be expressed as equation (7).

$$\begin{aligned} \min(RMSE) = \min(& \sqrt{\frac{1}{N} \sum_{i=1}^{k+n+1} (y_i(t) - \omega_i y\hat{}(t))^2} \\ & s.t.\ \omega_i \in [\omega_{\min}, \omega_{\max}] \end{aligned} \tag{7}$$

where $\hat{y}(t)$ is the predictive value of each prediction model, ω_i is the weight coefficient of each prediction model, k is the number of improved extreme learning machine (ELM) models, n is the number of seasonal autoregressive integrated moving average (SARIMA) models, the length of traffic flow time series is N.

References [34]–[39] also use combined prediction method with optimization algorithm to get high-precision results. The difference is that they use different fitness function and different optimization algorithm, such as particle swarm optimization, genetic algorithm, artificial bee colony algorithm, firefly algorithm.

IV. PROPOSED COMBINED POWER MODEL

Our proposed energy prediction model is to build a strong predictor based on the three single predictors. Its specific structure is shown in figure 1. The left side of figure 1 is the three single predictors and the right side is the structure of RBF. The output of three single predictors are the input of the RBF, the output of RBF is the output of the combined power model. The combined power model is to get suitable weight of input of RBF.

RBF adopts multi-dimensional space interpolation technology, which is simple in structure, simple in training and fast in learning convergence. It can approach any nonlinear function, and is widely used in pattern recognition, nonlinear control and other fields. It is usually composed of input layer, hidden layer and output layer. The basic idea is that RBF, as the ''base'' of the hidden element, constitutes the hidden layer space, so the input vector can be directly mapped to the hidden layer space without weight connection. The hidden layer transforms the input vector, and transforms the input data of low-dimensional mode into high-dimensional space, so that the problem of linear non-separability in low-dimensional space can be linearly separable in high-dimensional space. The output of the network is the linear weighted sum of the output of the hidden element. The mapping of network from input to output is nonlinear, while the output of network is linear for adjustable parameters. The weight of the network can be directly solved by the linear equations, thus speeding up the learning speed and avoiding the local minimum problem.

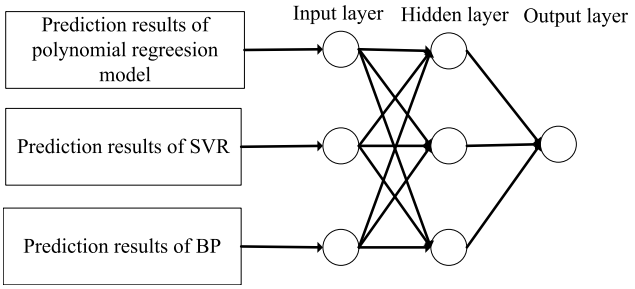

FIGURE 1. The structure of adaboost RBF combined prediction energy model.

If the input dimension of RBF is n_{i}, the output dimension is n_{o}, and the number of samples in the training sample set is m, then there are n_{i} neurons in the input layer of RBF, that is, the number of neurons in the input layer is consistent with that of the input sample dimension; the number of neurons in the hidden layer is determined by the expected training objectives; there are n_{o} neurons in the output layer. Therefore, the structure of RBF can be determined according to training samples and training objectives.

The combined estimation method of adaboost RBF is mainly divided into three steps. First, select several single power prediction models; second, use RBF to acquire and optimize its nonlinear parameters with the best performance; thirdly, use the best parameters of RBF to get the result of training samples. Fourthly, the adaboost algorithm [40] is used to adjust the weight distribution of training samples in RBF several times. Finally, three single predictors are fused into a strengthened combination predictor by getting best weight distribution of RBF to obtain high-precision prediction value. The specific steps are as follows.

Step 1: normalize the feature of architecture level $\{x_p\}$ to get the normalized series $\{\bar{x}_p\}$.

Step 2: polynomial regression model, SVR model and BP neural network model are used to predict $\{\bar{x}_p\}$ respectively, and the results are $\hat{y}_{1p}, \hat{y}_{2p}$ and $\hat{y}_{3p}$ respectively, $p = 1, 2, \cdots, n$.

Step 3: take $\hat{y}_{1p}, \hat{y}_{2p}$ and $\hat{y}_{3p}$ as the input of RBF, and the actual energy value y_p as the output of RBF to form the sample set $T = \{(\hat{y}_{11}, \hat{y}_{21}, \hat{y}_{31}, y_1), (\hat{y}_{12}, \hat{y}_{22}, \hat{y}_{32}, y_2), \cdots, (\hat{y}_{1n}, \hat{y}_{2n}, \hat{y}_{3n}, y_n)\}$, which is divided into training sample set T_1 which has n_1 samples and test sample set T_2 which has n_2 samples, $n = n_1 + n_2$.

Step 4: initialize the training sample set T_1, and the weight distribution Q_m can be expressed as $Q_m = \{w_{11}, w_{12}, \ldots w_{1i}, \ldots w_{1n_1}\}, m = 1$.

Where: Q_m is the weight distribution of T_1 in the mth RBF training; w_{1i} is the weight of the ith training sample, which can be expressed as $w_{1i} = \frac{1}{n_1}$.

Step 5: training network method is as follows.

(1) The RBF is trained by the training sample set T_1 with weight distribution Q_m, and the single predictor $R_m(\hat{y}_{11}, \hat{y}_{21}, \hat{y}_{31})$ is obtained;

(2) The maximum error of R_m on training sample set T_1 is calculated as follows:

$$E_m = \max |y_i - R_{mi}(\hat{y}_1, \hat{y}_2, \hat{y}_3)|, \quad i = 1, 2, \ldots, n_1$$

where: y_i is the actual energy corresponding to the ith training sample; $R_{mi}(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ is the prediction result of $R_m(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ for the ith training sample. The linear relative error of each training sample is $e_{mi} = \frac{|y_i - R_{mi}(\hat{y}_1, \hat{y}_2, \hat{y}_3)|}{E_m}$.

Then the error of $R_{mi}(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ on the training sample set T_1 is $e_m = \sum_{i=1}^{n} w_{mi} e_{mi}$.Where, w_{mi} is the weight of the ith training sample in the mth training.

(3) Calculate the coefficient α_m of $R_m(\hat{y}_1, \hat{y}_2, \hat{y}_3)$ as $\alpha_m = \frac{1}{2} \ln \frac{1-e_m}{e_m}$.

(4) Update the weight distribution of training sample set T_1 by equation (8) to obtain a training sample set with weight distribution Q_{m+1}, which is used for the next training.

$$\begin{aligned} Q_{m+1} &= \{w_{m+1,1}, w_{m+1,2}, \ldots w_{m+1,i}, \ldots w_{m+1,n_1}\} \\ w_{m+1,i} &= \frac{w_{mi}}{Z_m} \alpha_m^{1-e_{mi}} \end{aligned} \tag{8}$$

where: $w_{m+1,i}$ is the weight of the ith training sample in the $m+1$ training; Z_m is the normalization operator, which ensures that the weight ratio of the samples is consistent and the sum of the weights is 1.

(5) Judge the relationship between m and M, where M is the maximum number of training. If $m < M$, let $m = m + 1$, repeat steps (1)∼(4), after training RBF M times, obtain M single predictors; if $m \geq M$, carry out step 6.

Step 6: combine the M single predictors into the enhanced predictor, and use the model to predict the test sample set T_2. The enhanced predictor can be expressed as $f(\hat{y}_1, \hat{y}_2, \hat{y}_3) = \sum_{m=1}^{M} \frac{\alpha_m}{\sum_{m=1}^{M} \alpha_m} R_m(\hat{y}_1, \hat{y}_2, \hat{y}_3)$

In conclusion, the flow chart of Adaboost RBF combination prediction method is shown in figure 2.

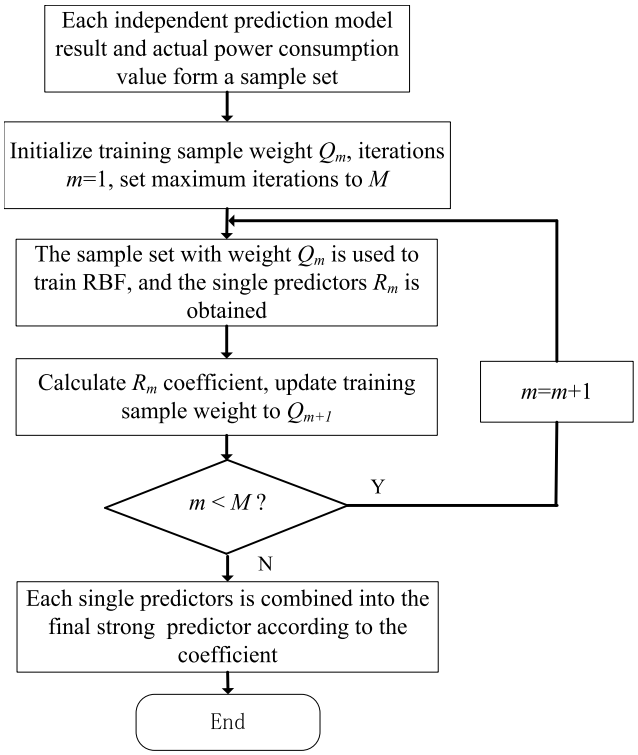

FIGURE 2. The flow chart of adaboost RBF combination energy model.

V. EXPERIMENT

A. EXPERIMENTAL PLATFORM

In order to verify the proposed software power model, the accuracy of the model is tested by a test set composed of open source games, media players and general software. The test programs include Italc, UniversalMediaServer, Jabref, Docfetcher, YOYOPlayer, TripleA, jbubblebraker, JIExplorer, Universal password manager, Ftpserver, FreeCell and Jajuk. For the convenience of description, we use the abbreviations Ita, UMS, Jab, Dof, YOP, TrA, jbb, JIE, Upm, Fts, FRC and Jaj instead of the above test procedures respectively. The five features of the test set are shown in table 2. In the experimental environment, the configuration of the hardware and software platform is shown in table 3; the measurement of the instantaneous power consumption and cumulative power consumption of the software adopts the Hoiki3334 multi-function power measurement. Since the sample size is relatively small, we use the LOOCV (Leave-One-Out-Cross-Validation) method to evaluate the prediction performance of the model.

B. EVALUATION INDEX

In this article, the e_{RMSE}(root mean square error), the e_{MAPE} (mean absolute percentage error), e_{RRMSE}(relative root mean square error), e_{MAE}(mean absolute error), e_{SSE}(square sum error) and the R^2(R square) are selected as the evaluation indexes of the prediction model, and these indexes can be calculated by equation (9).

$$\begin{aligned} e_{RMSE} &= \sqrt{\frac{1}{N} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2} \\ e_{MAPE} &= \frac{1}{N} \sum_{t=1}^{N} |\frac{y_t - \hat{y}_t}{y_t}| \times 100\% \end{aligned}$$

TABLE 2. Characteristic values of the test programs.

Feature	Ita	UMS	Jab	Dof	YOP	TrA	jbb	JIE	Upm	Fts	FRC	Jaj
V	91	2784	5140	4290	2674	7266	222	1080	382	2028	185	4428
E	60	1980	4070	2884	3153	14196	159	819	201	1822	159	2267
L	3	10	17	8	9	13	5	7	5	8	5	13
C	0.104	0.254	0.198	0.205	0.5	0.397	0.295	0.227	0.163	0.294	0.295	0.194
K	1.429	3.492	3.84	2.931	3.134	3.876	1.632	3.112	1.712	3.562	1.632	3.612
Power	4.49	8.86	3.78	2.64	3.61	8.63	2.82	3.72	3.87	6.14	2.79	5.83

TABLE 3. Software and hardware experiment platform.

Category	Configuration
Hardware Platform	Name:ThinkPad E530 CPU: Intel CORE i5 3210M 2.5GHz Dual core/Four thread Ivy Bridge Cache:3MB L3 cache 512KB L2 cache, 64KB L1cache Memory:DDR3 4GB Frequency 1600MHz Chipset:Intel HM77
Software Environment	Ubuntu14.04.3
	Matlab 2018b
Power meter	Name:HOIKI 3332 Accuracy:±0.1 % rdg Power range:15mW∼30kW Sampling frequency: 1 Hz∼100 kHz Cumulative measurement range: 0∼±999999MWh

$$
\begin{aligned}
e_{RRMSE} &= \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(\frac{y_t-\hat{y}_t}{y_t}\right)^2} \\
e_{MAE} &= \frac{1}{N}\sum_{t=1}^{N}|y_t-\hat{y}_t| \\
e_{SSE} &= \sum_{t=1}^{N}(y_t-\hat{y}_t)^2 \\
R^2 &= 1-\frac{\sum_{t=1}^{N}(y_t-\hat{y}_t)^2}{\sum_{t=1}^{N}(y_t-\bar{y}_t)^2}
\end{aligned}
\tag{9}
$$

where: n is the number of estimation; y_t is the actual energy consumption value of the tth program; $\hat{y}_t$ is the predicted energy consumption value of the tth program.

C. ACQUISITION OF PROGRAM FEATURES

Li *et al.* [27] has showed the characteristics at the architecture level and detailed the calculation process. Therefore, these characteristics are used in this article, namely, number of nodes, number of direct edges, average path length, clustering coefficient, average degree. Their specific calculation methods are listed in table 4.

In table 4, V_i is node i and its initial value is 1. WMC_i is the WMC (Weighted Methods per Class) value of class i. e_{ij} is edges between node i and node j. W_{ij} is the total number of methods in node i dependent on node j. N is the number of nodes in the network. d_{ij} is the short path between two nodes V_i and V_j.E_i is the existing links connecting to other nodes of node i, and $k_i * (k_i - 1)$ is the maximum possible number of such links of node i. C_i is clustering coefficient of node i. C is the clustering coefficient of the whole network. More details about these characteristics can be found in [27].

TABLE 4. Calculation of the characteristics.

Characteristics	Abbreviation	Calculation
number of nodes	V	$V=\sum_{i=1}^{N} v_i * WMC_i$
number of direct edges	E	$E=\sum_{i,j=1,i\neq j}^{N} e_{ij} * W_{ij}$
average path length	L	$L=\frac{1}{N(N-1)}\sum_{v_i\neq v_j}^{N} d_{ij}$
clustering coefficient	C	$C_i=\frac{E_i}{k_i(k_i-1)}$ $C=\frac{1}{N}\sum_i C_i$
average degree	K	$K=\frac{E}{N}$

D. ESTIMATION RESULTS AND COMPARATIVE ANALYSIS

1) ESTABLISH POLYNOMIAL REGRESSION MODEL

The original energy consumption data is normalized, and then the regression coefficient is solved by the least square method. The regression coefficient (−0.0011, 0.0006, 0.0069, −4.0814, 2.3046) with the larger square of the correlation coefficient is taken as the final fitting model.

TABLE 5. Comparison with single prediction model.

LOOCV group	Methods	e_{RMSE}	e_{MAPE}	e_{RRMSE}	e_{MAE}	e_{SSE}	R^2
Group 1	polynomial regression	1.4024	0.2425	0.2989	1.0981	23.5991	0.5391
	SVR	1.3064	0.2231	0.2373	1.0719	20.4812	0.6000
	BP	0.9768	0.2005	0.2397	0.8408	11.4494	0.7763
	Adaboost RBF	0.9039	0.1339	0.1525	0.6941	9.80544	0.8085
Group 2	polynomial regression	1.4119	0.2492	0.2968	1.1171	23.9206	0.5328
	SVR	1.1490	0.1931	0.2052	0.9344	15.8426	0.6905
	BP	1.5136	0.1956	0.2973	0.9460	27.4901	0.4631
	Adaboost RBF	0.8160	0.1645	0.1788	0.7343	7.99055	0.8439
Group3	polynomial regression	1.4669	0.2281	0.2706	1.0971	25.8227	0.4956
	SVR	1.2022	0.2040	0.2171	0.9830	17.3436	0.6612
	BP	1.3109	0.2072	0.2319	1.0306	20.6205	0.5972
	Adboost RBF	0.7178	0.0800	0.1015	0.4522	6.18250	0.8792
Group 4	polynomial regression	1.9313	0.2445	0.4787	1.0726	44.7610	0.1258
	SVR	1.4662	0.2499	0.2676	1.1986	25.7978	0.4961
	BP	1.4249	0.2215	0.3074	1.0093	24.3630	0.5241
	Adaboost RBF	0.9768	0.2005	0.2397	0.8408	11.4494	0.7763
Group 5	polynomial regression	1.4456	0.2610	0.3204	1.1556	25.0781	0.5102
	SVR	1.6725	0.2077	0.2407	1.1271	33.5651	0.3444
	BP	1.1030	0.2048	0.2552	0.9163	14.5987	0.7148
	Adaboost RBF	1.2665	0.1631	0.1894	0.9073	19.2488	0.6240
Average	polynomial regression	1.5316	0.2450	0.3331	1.1082	28.6363	0.4407
	SVR	1.3593	0.2155	0.2336	1.0630	22.6061	0.5585
	BP	1.2658	0.2059	0.2663	0.9486	19.7044	0.6152
	Adaboost RBF	0.9362	0.1484	0.1724	0.7258	10.9354	0.7864

Finally, the data are de-normalized to get the final energy consumption data.

2) ESTABLISH SVR MODEL

We use support vector regression method with precomputed kernel function, and the penalty factor c and width function g are cross calculated twice by grid search method. The optimal parameter pair is $c = 1.6782$, $g = 1.8769$.

3) ESTABLISH BP MODEL

Because the single hidden layer BP neural network has a good fitting effect, the single hidden layer BP neural network structure is adopted. The network structure is set as 5-6-1, the feature of program is taken as the input of network, and the corresponding energy consumption value is taken as the output of network. The prediction error and training time of the network are weighed through repeated experiments. The excitation functions of hidden layer and output layer are *tansig* function and *purelin* function respectively. The maximum number of network training is 200, the training target is 0.00004, and the learning rate is 0.02.

4) ENTROPY WEIGHT COMBINATION PREDICTION METHOD

After getting the results of three single prediction model, we calculate the absolute value of error, and generate evaluation matrix. We use equation (5) to obtain the entropy information of the each prediction model, after that the weight of each single prediction model can be calculated by equation (6). In this model, a is 3, b is 11.

5) PSO COMBINATION MODEL

For PSO algorithm, the most important thing is to establish the fitness function. Our fitness function also minimizes the error between actual and prediction value which can be expressed by RMSE shown by equation (10). After that, we should also set the parameter of $c1$, $c2$, max generations times (*maxgen*), size of population (sizepop), and velocity range (*velran*) and value range of population (*popvalran*). The optimal parameter pair is $c1 = 1.3467$, $c2 = 1.4678$, $maxgen = 50$, $sizepop = 100$, *velran* is between 0 and 1, *and popvalran* is between 0 and 1.

$$\min(RMSE) = \min\left(\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i(t)) - \sum_{k=1}^{3}\omega_i y\hat{}(t))^2}\right) \quad s.t.\ \omega_i \in [0,1] \tag{10}$$

6) COMPARATION WITH THE SINGLE PREDICTION MODEL

The evaluation indexes of error for the above three prediction models are using equation (9) and the compared results with that of adboost RBF under the test set samples are shown in table 5. For better evaluating the effects, we divided test program into five groups which are used the LOOCV method.

It can be seen from table 5 that the prediction errors of each single prediction model are large compared with that of the adaboost RBF. In group 1, group 4 and group 5, the value of evaluation index of BP are the better than that of polynomial regression and SVR and the effect of adaboost RBF are best among that of BP, polynomial regression and SVR. In group 2 and group 3, the value of evaluation index of SVR are the better than that of polynomial regression and BP and the effect of adaboost RBF are better than that of BP, polynomial regression and SVR. From the average of the five groups of experimental data, we can see that adaboost RBF has the best effect, followed by BP, SVR, and finally polynomial regression. Figure 3 shows the comparison among the predicted value of adaboost RBF, actual energy

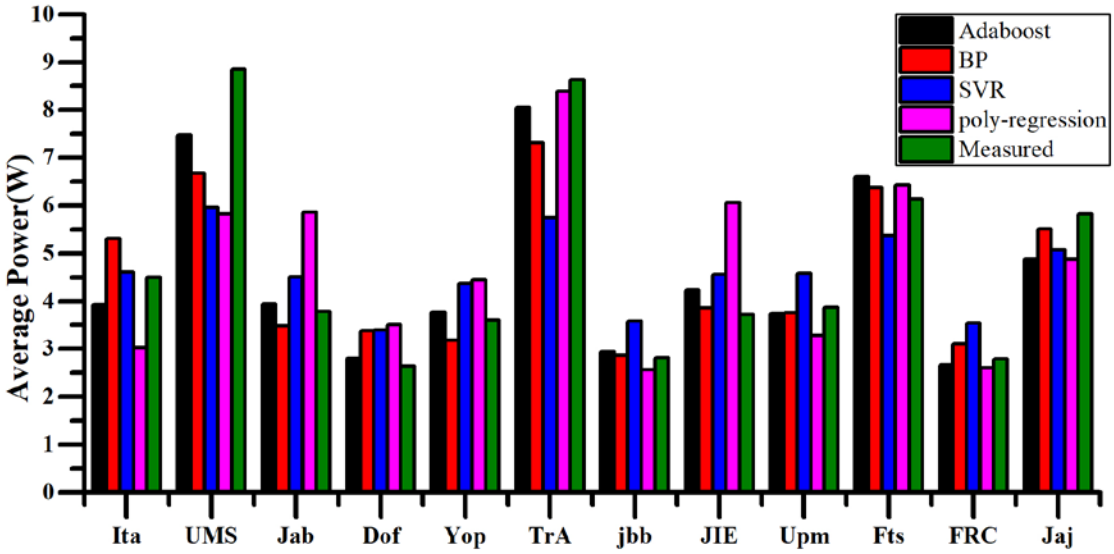

FIGURE 3. Compared with single prediction model.

TABLE 6. Comparison with combination prediction model.

LOOCV group	Methods	e_{RMSE}	e_{MAPE}	e_{RRMSE}	e_{MAE}	e_{SSE}	R^2
Group 1	entropy weight combination model	1.0822	0.1597	0.2156	0.7820	14.0540	0.7255
	PSO	1.1201	0.2085	0.2713	0.8729	15.0556	0.7060
	adaboost RBF	0.9039	0.1339	0.1525	0.6941	9.80544	0.8085
Group 2	entropy weight combination model	1.1888	0.1717	0.2171	0.8466	16.9596	0.6688
	PSO	1.1376	0.2072	0.2652	0.8772	15.5297	0.6967
	adaboost RBF	0.8160	0.1645	0.1788	0.7343	7.99055	0.8439
Group3	entropy weight combination model	1.1440	0.1470	0.1818	0.7706	15.7037	0.6933
	PSO	1.0759	0.1841	0.2247	0.8251	13.8900	0.7287
	adaboost RBF	0.7178	0.0800	0.1015	0.4522	6.18250	0.8792
Group 4	entropy weight combination model	1.2570	0.1947	0.2788	0.8946	18.9611	0.6297
	PSO	1.3688	0.2593	0.3472	1.0472	22.4840	0.5609
	adaboost RBF	0.9768	0.2005	0.2397	0.8408	11.4494	0.7763
Group 5	entropy weight combination model	1.1184	0.1879	0.2060	0.9036	15.0103	0.7069
	PSO	1.0496	0.2183	0.2478	0.9293	13.2200	0.7418
	Adaboost RBF	1.2665	0.1631	0.1894	0.9073	19.2488	0.6240
Average	entropy weight combination model	1.1216	0.1643	0.2131	0.8039	15.3228	0.7008
	PSO	1.1756	0.2148	0.2771	0.9056	16.7398	0.6731
	Adaboost RBF	0.9362	0.1484	0.1724	0.7258	10.9354	0.7864

value, the predicted value of BP, the predicted value of SVR and the predicted value of polynomial regression. From the figure 3, we can see the predicted value of adaboost RBF model proposed in this article are closer to the actual energy value than those of each single prediction model. It can be seen from table 5 that R^2 of adaboost RBF has the bigger value than that of three single prediction model, which shows that the prediction value of adaboost RBF has the highest fitting degree with the actual curve. The e_{RMSE}, e_{MAPE}, e_{RRMSE}, e_{MAE}, and e_{SSE} of the adaboost RBF model are significantly smaller than those of other single prediction models. Compared with polynomial regression model, SVR prediction model and BP neural network prediction model, adaboost RBF improves the accuracy of e_{RMSE} by 38.87%, 31.12% and 26.04% on average respectively, e_{MAPE} by 39.44%, 31.16% and 27.94% on average respectively, e_{RRMSE} by 48.24%, 26.20% and 35.26% on average respectively, e_{MAE} by 34.50%, 31.72% and 23.49% on average respectively, e_{SSE} by 61.81%, 51.63% and 44.50% on average respectively, which shows that the method proposed in this article can effectively combine several prediction methods and significantly improve the prediction accuracy.

Table 6 shows the effect among combined prediction model. In group 1, group 2, group 3 and group 4, the e_{RMSE}, e_{MAPE}, e_{RRMSE}, e_{MAE}, and e_{SSE} of the adaboost RBF model are significantly smaller than those of other combined prediction models (entropy weight combination model and PSO) and the R^2 of adaboost RBF has the bigger value than that of two combined prediction models. This shows adaboost RBF has the better accuracy. In group 5, the e_{MAPE}, e_{RRMSE} and e_{MAE} of adaboost RBF are smaller than that of other combined prediction model and the e_{SSE} and e_{RMSE} are bigger that of adaboost RBF. The R^2 of adaboost RBF has the smaller value than that of two combined prediction models.

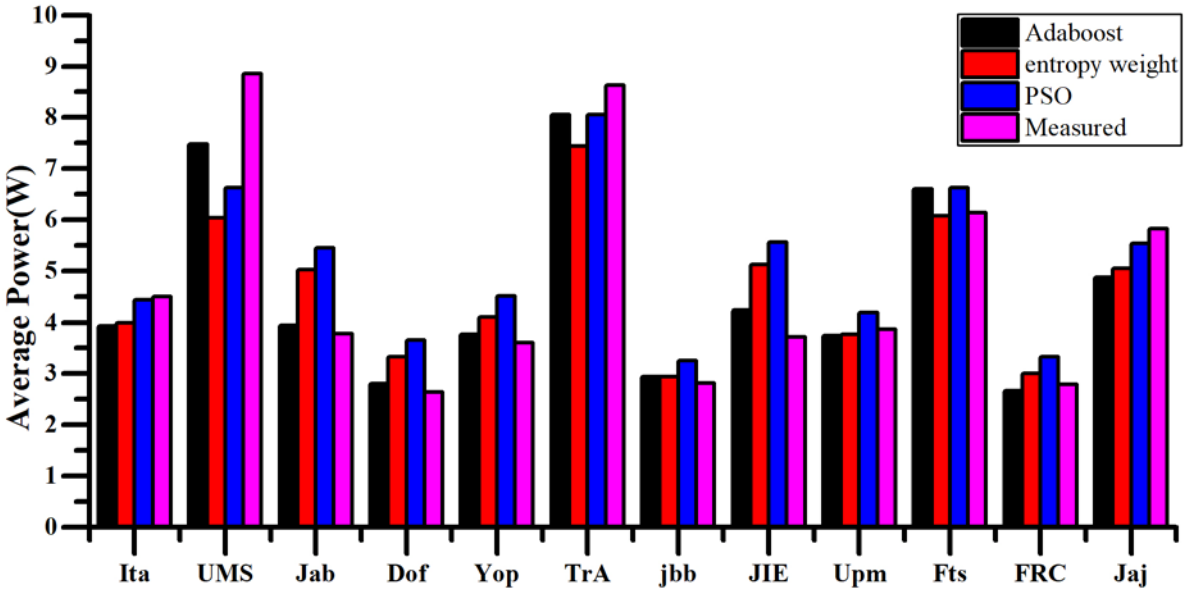

FIGURE 4. Compared with combined prediction model.

This shows that adaboost RBF behaves worse than that of two combined prediction model in this group. On average, the e_{RMSE}, e_{MAPE}, e_{RRMSE}, e_{MAE}, and e_{SSE} of the adaboost RBF model are significantly smaller than those of other combined prediction models and the R^2 of adaboost RBF has the bigger value than that of two combined prediction models. This shows that adaboost RBF has the higher accuracy in most cases. Figure 4 shows the comparison among the predicted value of adaboost RBF, actual energy value, and the predicted value of entropy weight combination model and the predicted value of PSO combined model. From the figure, we can see the predicted value of adaboost RBF model proposed in this article are closer to the actual energy value than those of each combined prediction model. It can be seen from figure 4 and table 6 that the R^2 of adaboost RBF are bigger than that of two combined model which shows that adaboost RBF has the highest fitting degree with the actual curve. Compared with entropy weight combined model and PSO combined model, adaboost RBF improves the accuracy of e_{RMSE} by 16.53% and 20.36% on average respectively, e_{MAPE} by 9.68% and 30.91% on average respectively, e_{RRMSE} by 19.10% and 37.78% on average respectively, e_{MAE} by 9.71% and 19.85% on average respectively, e_{SSE} by 28.63% and 34.67% on average respectively, which shows that the method proposed in this article can effectively combine several prediction methods and significantly improve the prediction accuracy. Through the analysis of the experimental data, we can draw the following conclusions: (1) the architecture level software power consumption model based on complex network is effective. Through the comparison of test sets, the average value of e_{RMSE}, e_{MAPE}, e_{RRMSE}, e_{MAE}, and e_{SSE} of our model are significantly reduced, which proves the effectiveness of our proposed method. (2) As an approach tool, adaboost RBF combined estimation method can solve this problem well and effectively reflect the nonlinear relationship. (3) It is verified that there is a certain correlation between software features and software power consumption, and the impact of each feature on software power consumption is different again.

VI. CONCLUSION

Energy consumption prediction is a very important problem. The current single prediction method cannot ensure the best prediction results in all task objects. In order to make full use of the regular pattern between features, integrate the advantages of each single prediction model, and further improve the accuracy of energy consumption prediction, this article proposes a combined energy prediction method named adaboost RBF. By using RBF to combine several single prediction models for nonlinear optimization to build prediction model, the advantages of each method are complementary, and the robustness effect of prediction model is significantly improved. Combined with adaboost algorithm to train RBF, the combination form of each single prediction model can be adjusted adaptively according to the characteristics of the analysis object, which greatly improves the prediction accuracy and scope of application. Through the verification of the measured data, the adaboost RBF will provide important clues for the research of energy consumption prediction, energy saving scheduling and resource allocation. In addition, the model can also be used as a general prediction model, which has broad application prospects in the fields of stock index prediction, traffic flow prediction, network flow prediction and logistics index prediction.

ACKNOWLEDGMENT

(Junke Li, Kai Liu, Mingjiang Li, and Deguang Li contributed equally to this work.)

REFERENCES

[1] P. Tafidis and J. Bandeira, "Interregional European cooperation platform to promote sustainable transport through ICT: An overview of best practices," in *Proc. 10th Int. Conf. Pervas. Technol. Rel. Assistive Environ.*, Jun. 2017, pp. 255–260.

[2] C. Louche, T. Busch, P. Crifo, and A. Marcus, "Financial markets and the transition to a low-carbon economy: Challenging the dominant logics," *Org. Environ.*, vol. 32, no. 1, pp. 3–17, Mar. 2019.

[3] *Climate Change and Land. The Approved Summary for Policymakers*, IPCC, Geneva, Switzerland, 2018.

[4] J. Cao and M. Huan, "Study on the mechanism of energy structure optimization to low-carbon economy," in *Proc. 6th Int. Forum Decis. Sci.*, Singapore: Springer, 2020, pp. 229–242.

[5] M. S. Raisinghani and E. C. Idemudia, *Green Information Systems for Sustainability*. Hershey, PA, USA: IGI Global, 2019, pp. 565–579.

[6] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, and J. Dongarra, "Investigating power capping toward energy-efficient scientific applications," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 6, Mar. 2019, Art. no. e4485.

[7] NVIDIA Corporation. (2014). *NVIDIA's Next Generation CUDA Compute Architecture: Kelper GK110/210*. White Paper V1.1. [Online]. Available: http://www.nvidia.com

[8] J. Li, B. Guo, S. Yan, D. Li, and Y. Huang, "Research on power data correction approach of GPU build-in sensor," *J. Univ. Electron. Sci. Technol. China*, vol. 45, no. 2, pp. 282–287, 2016.

[9] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 437–445, Dec. 1994.

[10] M. Bazzaz, M. Salehi, and A. Ejlali, "An accurate instruction-level energy estimation model and tool for embedded systems," *IEEE Trans. Instrum. Meas.*, vol. 62, no. 7, pp. 1927–1934, Jul. 2013.

[11] J. Tan, K. Yan, S. L. Song, and X. Fu, "Energy-efficient GPU l2 cache design using instruction-level data locality similarity," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 6, pp. 1–18, Oct. 2020.

[12] C. Lee and W. W. Ro, "Simultaneous and speculative thread migration for improving energy efficiency of heterogeneous core architectures," *IEEE Trans. Comput.*, vol. 67, no. 4, pp. 498–512, Apr. 2018.

[13] A. H. Ashouri, W. Killian, J. Cavazos, G. Palermo, and C. Silvano, "A survey on compiler autotuning using machine learning," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, Jan. 2019.

[14] R. W. Ahmad, A. Naveed, J. J. P. C. Rodrigues, A. Gani, S. A. Madani, and J. Shuja, "Enhancement and assessment of a code-analysis-based energy estimation framework," *IEEE Syst. J.*, vol. 13, no. 1, pp. 1052–1059, Mar. 2019.

[15] B. R. Bruce, J. Petke, M. Harman, and E. T. Barr, "Approximate oracles and synergy in software energy search spaces," *IEEE Trans. Softw. Eng.*, vol. 45, no. 11, pp. 1150–1169, Nov. 2019.

[16] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4411–4422, Jun. 2019.

[17] A. Banerjee, L. K. Chong, C. Ballabriga, and A. Roychoudhury, "EnergyPatch: Repairing resource leaks to improve energy-efficiency of Android apps," *IEEE Trans. Softw. Eng.*, vol. 44, no. 5, pp. 470–490, May 2018.

[18] G. Wei, D. Qian, H. Yang, Z. Luan, and L. Wang, "FPowerTool: A function-level power profiling tool," *IEEE Access*, vol. 7, pp. 185710–185719, 2019.

[19] J. Li, B. Guo, Y. Shen, D. Li, and Y. Huang, "Power modeling approach for GPU source program," *J. Elect. Eng. Technol.*, vol. 13, no. 1, pp. 181–191, 2018.

[20] L. Mukhanov, P. Petoumenos, Z. Wang, N. Parasyris, D. S. Nikolopoulos, B. R. De Supinski, and H. Leather, "ALEA: A fine-grained energy profiling tool," *ACM Trans. Archit. Code Optim.*, vol. 14, no. 1, pp. 1–25, Apr. 2017.

[21] H. Chen, Y. Li, and W. Shi, "Fine-grained power management using process-level profiling," *Sustain. Comput., Informat. Syst.*, vol. 2, no. 1, pp. 33–42, Mar. 2012.

[22] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, "Process-level power estimation in VM-based systems," in *Proc. 10th Eur. Conf. Comput. Syst. EuroSys*, Apr. 2015, pp. 1–14, doi: 10.1145/2741948.2741971.

[23] Z. Lu and Y. Yao, "Thread voting DVFS for manycore NoCs," *IEEE Trans. Comput.*, vol. 67, no. 10, pp. 1506–1524, Oct. 2018.

[24] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.

[25] E. Jagroep, J. M. van der Werf, S. Brinkkemper, L. Blom, and R. van Vliet, "Extending software architecture views with an energy consumption perspective," *Computing*, vol. 99, no. 6, pp. 553–573, Jun. 2017.

[26] J.-M. Horcas, M. Pinto, and L. Fuentes, "Variability models for generating efficient configurations of functional quality attributes," *Inf. Softw. Technol.*, vol. 95, pp. 147–164, Mar. 2018.

[27] D. Li, B. Guo, Y. Shen, J. Li, and Y. Huang, "Software power modeling method at architecture level based on complex networks," *Sustain. Comput., Informat. Syst.*, vol. 12, pp. 34–42, Dec. 2016.

[28] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019.

[29] Y. Cai, L. Xiao, R. Kazman, R. Mo, and Q. Feng, "Design rule spaces: A new model for representing and analyzing software architecture," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 657–682, Jul. 2019.

[30] T. Chaikalis and A. Chatzigeorgiou, "Forecasting java software evolution trends employing network models," *IEEE Trans. Softw. Eng.*, vol. 41, no. 6, pp. 582–602, Jun. 2015.

[31] J. Nalepa and M. Kawulok, "Selecting training sets for support vector machines: A review," *Artif. Intell. Rev.*, vol. 52, no. 2, pp. 857–900, Aug. 2019.

[32] J. Li, B. Guo, Y. Shen, D. Li, and Y. Huang, "A modeling approach for energy saving based on GA-BP neural network," *J. Electr. Eng. Technol.*, vol. 11, no. 5, pp. 1289–1298, Sep. 2016.

[33] Z. Tian, "Approach for short-term traffic flow prediction based on empirical mode decomposition and combination model fusion," *IEEE Trans. Intell. Transp. Syst.*, early access, May 8, 2020, doi: 10.1109/tits.2020.2987909.

[34] Z. Tian, "Network traffic prediction method based on wavelet transform and multiple models fusion," *Int. J. Commun. Syst.*, vol. 33, no. 11, Jul. 2020, Art. no. e4415, doi: 10.1002/dac.4415.

[35] Z. Tian, "Short-term wind speed prediction based on LMD and improved FA optimized combined kernel function LSSVM," *Eng. Appl. Artif. Intell.*, vol. 91, May 2020, Art. no. 103573.

[36] Z. S. Tian Li and Y. Wang, "A prediction approach using ensemble empirical mode decomposition-permutation entropy and regularized extreme learning machine for short-term wind speed," *Wind Energy*, vol. 23, no. 7, pp. 177–206, 2020.

[37] Z. Tian, Y. Ren, and G. Wang, "Short-term wind speed prediction based on improved PSO algorithm optimized EM-ELM," *Energy Sour., A, Recovery, Utilization, Environ. Effects*, vol. 41, no. 1, pp. 26–46, Jan. 2019.

[38] Z. Tian, Y. Ren, and G. Wang, "Short-term wind power prediction based on empirical mode decomposition and improved extreme learning machine," *J. Elect. Eng. Technol.*, vol. 13, no. 5, pp. 1841–1851, 2018.

[39] T. Zhongda, L. Shujiang, W. Yanhong, and S. Yi, "A prediction method based on wavelet transform and multiple models fusion for chaotic time series," *Chaos, Solitons Fractals*, vol. 98, pp. 158–172, May 2017.

[40] C. Xiao, N. Chen, C. Hu, K. Wang, J. Gong, and Z. Chen, "Short and mid-term sea surface temperature prediction using time-series satellite data and LSTM-AdaBoost combination approach," *Remote Sens. Environ.*, vol. 233, Nov. 2019, Art. no. 111358.

JUNKE LI (Member, IEEE) received the B.S. degree from Henan Polytechnic University, Henan, China, in 2010, the M.S. degree from Southwest University, Chongqing, China, in 2013, and the Ph.D. degree from Sichuan University, Sichuan, China, in 2017, all in computer science. His primary current research interests include green computing and embedded systems.

KAI LIU received the B.S. degree in computer science from Xinxiang University, Henan, China, in 2008, and the M.S. degree in educational technology from Xinyang Normal University, Henan. Her current research interests include intelligent systems and software systems.

MINGJIANG LI received the B.S. degree in computer science from the Tianjin University of Technology and Education, Tianjin, China, in 1998. His research interests include artificial intelligence and big data.

DEGUANG LI (Member, IEEE) received the B.S. degree from PLA Information Engineering University, Henan, China, in 2010, the M.S. degree from Northeastern University, in 2012, and the Ph.D. degree from Sichuan University, Sichuan, China, in 2017, all in computer science. His research interests include green computing and software evolution.