Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 4030146, 12 pages
https://doi.org/10.1155/2017/4030146

Research Article

A MapReduce Based High Performance Neural Network in Enabling Fast Stability Assessment of Power Systems

Yang Liu,[1] Youbo Liu,[1] Junyong Liu,[1] Maozhen Li,[2] Tingjian Liu,[1] Gareth Taylor,[2] and Kunyu Zuo[1]

[1] *School of Electrical Engineering and Information, Sichuan University, Chengdu 610065, China*
[2] *School of Electronic and Computer Engineering, Brunel University London, Uxbridge UB8 3PH, UK*

Correspondence should be addressed to Youbo Liu; liuyoubo@scu.edu.cn

Received 2 May 2016; Revised 7 August 2016; Accepted 23 November 2016; Published 8 February 2017

Academic Editor: Huaguang Zhang

Transient stability assessment is playing a vital role in modern power systems. For this purpose, machine learning techniques have been widely employed to find critical conditions and recognize transient behaviors based on massive data analysis. However, an ever increasing volume of data generated from power systems poses a number of challenges to traditional machine learning techniques, which are computationally intensive running on standalone computers. This paper presents a MapReduce based high performance neural network to enable fast stability assessment of power systems. Hadoop, which is an open-source implementation of the MapReduce model, is first employed to parallelize the neural network. The parallel neural network is further enhanced with HaLoop to reduce the computation overhead incurred in the iteration process of the neural network. In addition, ensemble techniques are employed to accommodate the accuracy loss of the parallelized neural network in classification. The parallelized neural network is evaluated with both the IEEE 68-node system and a real power system from the aspects of computation speedup and stability assessment.

1. Introduction

In recent decades, dozens of large power blackouts have occurred. Loss of stability has been widely recognized as the most critical factor that leads to power system collapse. Meanwhile, modern power systems are exposed to higher risks than ever before due to the increasingly stressed operation conditions caused by renewable energy penetrations, electricity market gaming, insufficient awareness technique, and shortage of investments [1]. These situations consequently reduce the dynamic stability of power systems when the severe disturbances occur.

Transient stability assessment (TSA) is an effective resort to evaluate dynamic security under various operations in control centers. To facilitate TSA, machine learning technologies have been widely applied in the past two decades, which is well summarized in an early literature [2]. Most of the existing works of the transient stability identification are focused on binary stable state prediction using clustering and classification. For example, Support Vector Machine, Decision Tree, and Artificial Neural Network (ANN) are the widely used approaches to detecting instability of power systems by using postfault trajectories within a few cycles [3–5]. On the other hand, a few of machine learning techniques have been investigated to enable dynamic coherency identification of power systems, providing critical information for system equivalents [6], islanding control [7], and area detection [8]. But coherency analysis has limited ability to determine the most disturbed units, which may lead to the eventual desynchronization.

Besides awareness of globally stable status, it is important for emergency control to understand which generator or group of generators have a tendency of desynchronization. Traditional stability predicators cannot point out the leading units while the coherency-based classification needs a longer time window to observe perturbance trajectories. The most feasible solution is to establish a set of trained predictors for each generator to enable individual identification [9]. But it is

admitted that it is computational intensive due to the fact that a power system normally has hundreds of generators, which generate massive volumes of data. Few machine learning techniques have considered the impact of the critical unstable generators (CUGs) in TSA of power systems. As a result, it has become a challenge for standalone machine learning techniques running on single computers to deal with TSA taking into account the impact of massive CUGs [10]. For this purpose, the application of high performance computing techniques has become a necessity.

This paper presents HBPNN, a high performance back propagation neural network using MapReduce computing model. Hadoop [11–13], which is an open-source implementation of MapReduce, is first employed to parallelize the neural network. The parallelized neural network is further enhanced using HaLoop [14] to reduce the computation overhead incurred in the iteration process of the neural network. In addition, ensemble techniques are employed to maintain high accuracy in classification when datasets are split into small data chunks and processed in parallel nodes. The parallelized neural network is evaluated with both the IEEE 68-node system and a real power system from the aspects of computation speedup and stability assessment.

The rest of the paper is organized as follows. Section 2 discusses the related work about the application of machine learning techniques for TSA. Section 3 presents in detail the design of HBPNN. Section 4 evaluates the performance of the parallelized neural networks and analyzes the experimental results. Section 5 concludes the paper and points out the future work.

2. Related Work

As wide area monitoring systems (WAMS) are now being deployed in large number of power systems, phasor measurement unit (PMU) is playing an ever increasingly vital role in dynamic security assessment [15]. A number of researches have been carried out to assess transient stability using PMU data. Among these research efforts, PMU trajectories based indicators are considered as efficient estimators to understand dynamic behaviors of power systems, especially in severe disturbances. For example, Alvarez and Mercado proposed seven trajectory based indices, which are suitable for fuzzy inference on real-time dynamic vulnerability [16]. Furthermore, Makarov et al. [17] presented a review on PMU-based security assessment offering a clear roadmap for further development.

Machine learning techniques have been widely employed for instability detection or stability margin estimation. However, few studies have been carried out for TSA by identifying CUGs in power systems due to massive volumes of data generated from the large number of the CUGs. For this purpose, this paper employs back propagation neural network (BPNN) to identify CUGs in a timely manner.

BPNN has proven to be effective in classification due to its gradient-descent feature that results in its remarkable function approximation. However, large-scale data processing brings a significant challenge to BPNN in computation. Rizwan et al. [18] employed a neural network on solar energy estimation. It is admitted that the large volume of data makes the data processing an extremely complex task, which affects the training efficiency severely. Wang et al. [19] pointed out that large-scale neural network becomes one of the mainstream tools for processing massive data. Al-Masri et al. [10] also applied adaptive neural network to evaluate stability for every single generator, aiming at providing more detailed stability information. But real power systems usually have hundreds of generators. It is admitted that standalone neural networks running on single computers can hardly handle the problem in a reasonable time.

In order to speed up the efficiency of BPNN, distributed computing technologies have been employed [20–22]. Gu et al. [23] presented a parallel neural network using in-memory data processing techniques to accelerate neural network. However, in their work the training data is simply segmented into data chunks without considering accuracy loss. Liu et al. [24] presented a MapReduce based parallel BPNN in processing a large set of mobile data. This work further employs AdaBoosting algorithm to accommodate the loss of accuracy of the parallelized neural work. Although AdaBoosting is a popular sampling technique, it may enlarge the weights of wrongly classified instances, which would deteriorate the algorithm accuracy. Another major limitation of this research lies in that it does not consider the high overhead of Hadoop in dealing with input and output files in the iteration process.

To solve the issue of processing large-scale data using BPNN in power system for stability analysis especially for identification of CUGs, the presented work in this paper employs HaLoop to reduce the high overhead incurred in computation iterations. It also proves feasibility of MapReduce based high performance neural network on efficient stability assessment, providing a general tool to parallelize the machine learning algorithms to facilitate coordinated training to a large number of generators.

3. The Design of HBPNN

3.1. BPNN. BPNN has been proved to be effective in classification. It employs feed-forward and back propagation mechanisms to train the parameters of the network.

In the feed-forward phase, let

(i) w_{ij} denote weight from ith neuron to jth neuron,

(ii) θ_j denote bias for varying the activity of the jth neuron,

(iii) o_{lj} denote output of the jth neuron from last layer,

(iv) o_{cj} denote output of the jth neuron of the current layer,

(v) I_j denote input of the jth neuron in hidden and output layers.

Therefore, I_j can be represented by

$$I_j = \sum_i w_{ij} o_{lj} + \theta_j. \quad (1)$$

In the neuron, the nonlinear equation is sigmoid function; therefore the output of the jth neuron from the current layer to next layer can be represented by

$$o_{cj} = \frac{1}{1 + e^{-I_j}}. \tag{2}$$

The output layer finally outputs its o_{cj}. The feed-forward phase is completed.

In the back propagation phase, let

(i) Err_j denote the error-sensitivity of certain layer,

(ii) t_j denote the desirable output of neuron j in the output layer,

(iii) Err_k denote error-sensitivity of one neuron in the last layer,

(iv) w_{kj} represent corresponding weight of Err_k.

Therefore, Err_j in the output layer and in the hidden layers can be represented by

$$\begin{aligned} \mathrm{Err}_j &= o_j\left(1 - o_j\right)\left(t_j - o_j\right), \\ \mathrm{Err}_j &= o_j\left(1 - o_j\right)\sum_k \mathrm{Err}_k w_{kj}. \end{aligned} \tag{3}$$

The weight w_{ij} and bias θ_j can be tuned, where η denotes the learning speed:

$$\begin{aligned} w_{ij} &= w_{ij} + \eta \mathrm{Err}_j o_j, \\ \theta_j &= \theta_j + \eta \mathrm{Err}_j. \end{aligned} \tag{4}$$

The back propagation phase is completed. Afterward, a second round of training starts. BPNN terminates if (5) or (6) is satisfied or a certain number of iterations has been reached.

$$\min\left(E\left[e^2\right]\right) = \min\left(E\left[(t-o)^2\right]\right), \tag{5}$$

$$\min\left(E\left[e^T e\right]\right) = \min\left(E\left[(t-o)^T (t-o)\right]\right). \tag{6}$$

For executing a classification task, a trained BPNN only needs to execute the feed-forward phase. The classification result can be achieved from the output layer of the network.

3.2. Time-Domain Simulation. The time-domain simulation of power system is modeled by means of differential algebraic equations (DAEs); the details of the model can be found in [25]. The outputs of the simulation, which are the status trajectories, can be utilized as the simulated PMU data for further analysis. In this study, an open-source package PST [26] is employed to simulate dynamic trajectories of concerned parameters for random faults in a certain interval of cycles.

3.3. BPNN Based Transient Stability Assessment. If a power angle difference $\Delta\delta_{ij}$ between any two generators i and j exceeds a specified threshold, for example, 270 or 360 degrees, the status of the system is considered as unstable.

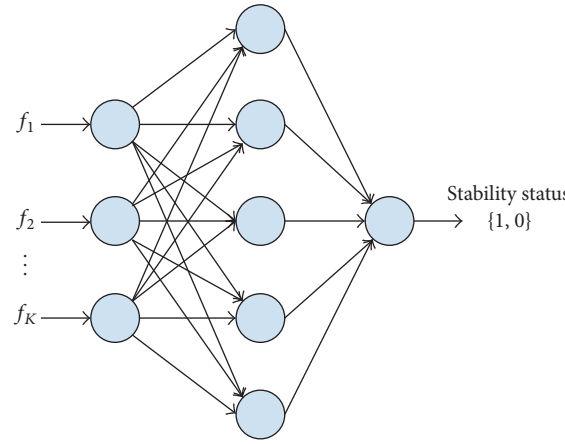


Figure 1: A BPNN based TSA.

Alternatively, the criterion using the center of inertia (COI) is usually applied to identify power system stability, which is expressed as

$$\left|\delta_i - \delta_{\mathrm{COI}}\right| \le \delta_{\max} \quad \forall i, \tag{7}$$

$$\delta_{\mathrm{COI}} = \frac{1}{M_T}\sum_{i=1}^{n} M_i \delta_i, \quad M_T = \sum_{i=1}^{N} M_i, \tag{8}$$

where δ_i and M_i represent rotor angle and inertia constant of generator i, M_T is the sum of M_i, N is the number of generators, and $\delta_{\max}$ is instability threshold which is defined as 180 degrees in this paper.

The training phase of BPNN based TSA is illustrated in Figure 1.

In Figure 1 $f_1, f_2, \ldots, f_K$ are the inputs of the network. The output is usually an integer value with 0 indicating instability while 1 indicates stability. After the training process is accomplished, if a fault occurs, the features obtained from a few cycles of the postfault trajectories will be fed into the trained network to extrapolate stability status within the subsequent several seconds. The majority of the existing works focus on improving accuracy of global stability prediction by improving the standalone BPNNs [8] as well as novel input features [27]. However, the stability margin, a value quantifying how far the current condition is from the loss of synchronization, is a crucial indicator that enables a clearer awareness of the dynamic impact level.

In this work, two trajectory based stability margin indicators, TSI and IS [28], are used as training targets, which are given as follows:

$$\mathrm{TSI} = \frac{360 - \delta\left(T\right)_{\max}}{360 + \delta\left(T\right)_{\max}} \times 100, \quad -100 < \mathrm{TSI} < 100,$$

$$S_T = \sqrt{\int_0^T \left\{\sum_{i=1}^{N} M_i \left[\delta_i\left(t\right) - \delta_{\mathrm{COI}}\left(t\right)\right]^2\right\} dt},$$

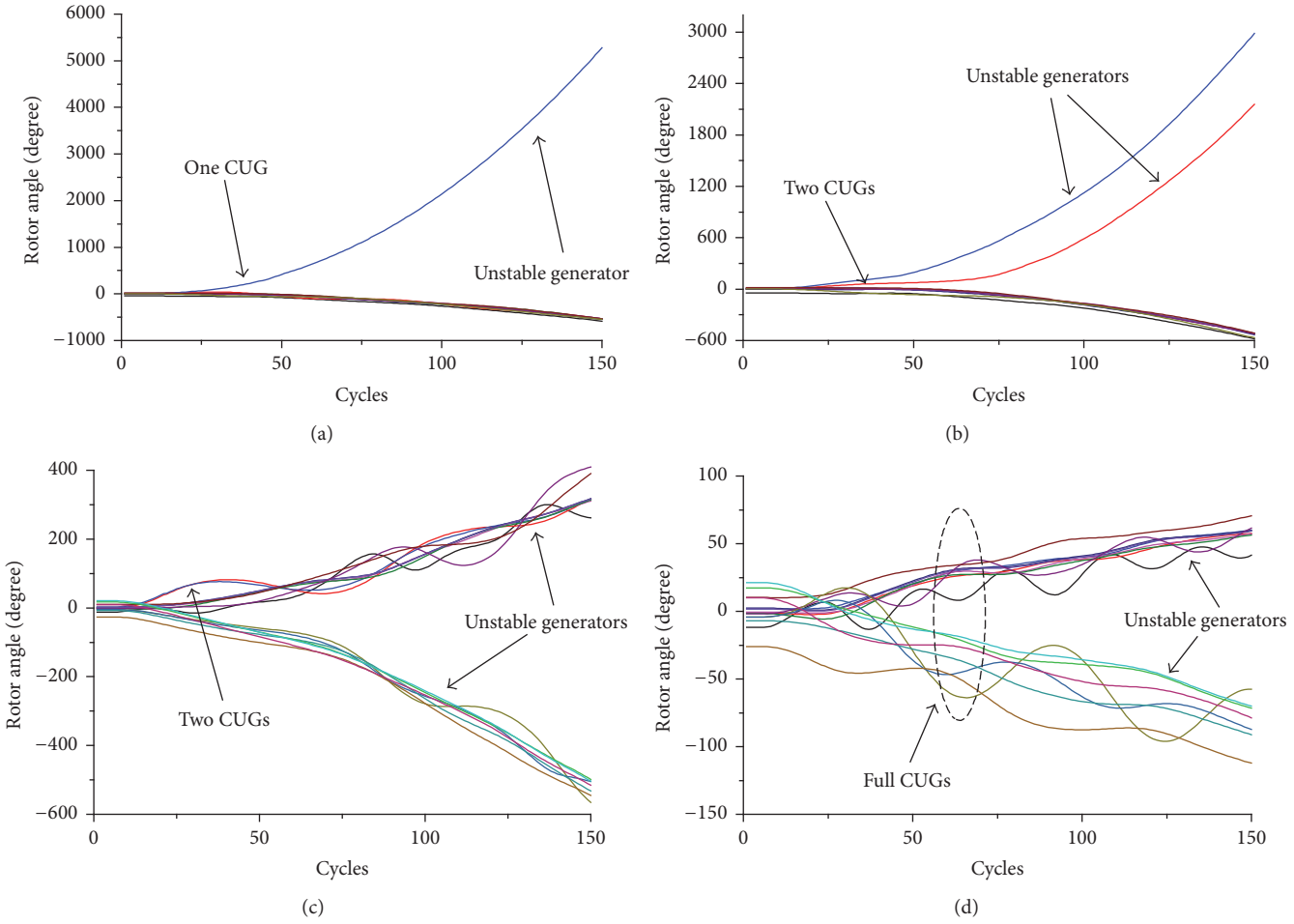


Figure 2: Four scenarios of CUGs.

$$\mathrm{IS} = 1 + \tanh\left(\frac{S_T}{\left|S_{T_{\max}} - S_T\right|}\right), \quad 1 \le \mathrm{IS} \le 2, \tag{9}$$

where $\delta(T)_{\max}$ is the maximal power angle difference between any generator pairs during the period of T and $\delta_i(t)$ is power angle of generator i at time point t.

Although there exist a wide range of features in previous works, most of them share similar parameters. According to these studies, the combination of these features can achieve an adequate accuracy of stability prediction. Moreover, these features not only are related to stability status but also contain the inherent information of stable margins. Therefore, the same set of input features is selected for BPNN training.

3.4. CUG Identification. CUGs are defined as the first group of the generators whose rotor angle is different from the rest of the generators exceeding a given threshold. Actually, CUGs are the most potential candidates of generator tripping that can be utilized to reduce transient power mismatch in a timely manner [29]. Figure 2 shows the power angle trajectories of different CUGs in the IEEE 68-node testing system.

The unstable generators belonged to the CUGs, because their leading (or lagging) rotor angle against other units must exceed the given threshold which is usually set to be equal to or little smaller than the wide-accepted instability criterion. For example, Figures 2(a) and 2(b) illustrate rotor angle trajectories of the CUGs, which also contain all the unstable generators. In this situation, all the generators are determined as unstable ones at the end of observation time window, 150 cycles. But, before that, none of the generators reaches the CUG threshold criterion. Therefore, the strict two-cluster instability pattern corresponds to the situation that all the generators are CUGs, such as the case of Figure 2(d). However, unlike Figures 2(a), 2(b), and 2(d), Figure 2(c) offers the different pattern in which the CUGs only are part of unstable units. Although it belongs to the leading cluster, ahead of other leading generators, the two generators indicated in Figure 2(c) meet the CUGs identification criterion at the very beginning of time windows. These two units are considered to be the most effective objects for the further control strategy.

For this purpose, the cycles of postfault rotor angle trajectories are clustered to identify CUGs from unstable

Table 1: Critical unstable generator indicator examples.

	Critical unstable generator indicator															
(a)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
(b)	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0
(c)	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
(d)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

generators, which are used as the target outputs of BPNN in the training process:

(1) Execute five seconds' time-domain simulation for a permanent fault followed by a clearing action; then collect the output rotor angle trajectory of each generator.

(2) Scan any two rotor angle trajectories cycle by cycle from the initial point of postfault duration. If there is an angle difference $\Delta\delta_{ij}$ exceeding critical unstable threshold, the power system is considered to be critically unstable; meanwhile, record this time point t.

(3) Extract rotor angle trajectory $\delta_i(t + \Delta t)$ for each generator, where Δt refers to CUG validation interval. However, if taking Δt as a relatively long period, such as 3 s, it is almost not possible to distinguish them from the subsequent unstable generators. According to the experience, Δt is preferably set to be 50 cycles, that is, 1 s.

(4) Perform *k-means* clustering to divide all $\delta_i(t + \Delta t)$ trajectories into two groups. Then calculate the COI trajectory of the clustered rotor angles for each group with time interval $t + \Delta t$ using (8).

(5) If the following constraint cannot be satisfied, the generators contained in group k which breaks (10) are tagged as the CUGs with a binary integer of 1.

$$\left|\delta_{k\,\mathrm{COI}}(t + \Delta t) - \delta_{\mathrm{COI}}(t + \Delta t)\right| \leq 180^{\circ}, \quad k \in \{1, 2\}. \tag{10}$$

Following the above identification procedure, the CUGs of the 16-machine testing system illustrated in Figure 2 can be indicated as shown in Table 1.

In Table 1, the CUG status is tagged by using the binary values, one means CUG, and zero means non-CUG.

3.5. Parallelizing BPNN

3.5.1. MapReduce, Hadoop, and HaLoop. MapReduce is a distributed computing model in enabling big data processing. The model supplies two types of functions: Map and Reduce. Map operates the mapping functions for major computing tasks while Reduce operates the collecting and outputting operations. The data in the processing flow is modeled using (key (K)-value (V)) pairs. Map processes each input *key-value* pair $\{K1, V1\}$ and outputs intermediate output $\{K2, V2\}$. Reduce collects the output pairs with the same keys and executes merging, shuffling operations. Finally, Reduce outputs the final results $\{V2\}$.

Hadoop framework is an open-source implementation [11] of MapReduce. The framework offers scalability, fault tolerance, load balancing, and a series of benefits for parallel and distributed computing in both homogeneous and heterogeneous environments. HaLoop [14] is also based on MapReduce and reuses most of the source code of Hadoop but facilitates data intensive applications with iterations.

3.5.2. Bootstrapping and Majority Voting. Bootstrapping is a kind of sampling algorithm [30]. Benefiting from sampling with replacement, the bootstrapped samples are able to simulate the sample distribution of the original dataset. Therefore, in our parallelization work, although the original training dataset is divided into subsets, due to the employment of the bootstrapping, the generalization of the trained neural network can be maintained to some extent. Majority voting is able to indicate the major element from a dataset based on voting. It enables HBPNN to create a strong classifier using a number of weak classifiers so that the classification accuracy can be maintained.

3.5.3. HBPNN Design. Motivated by the previous work of MapReduce based BPNN proposed by Liu et al. [32], the algorithm contains two phases including the generation of the bootstrapped samples and the parallelization of the BPNN. Initially, HBPNN inputs the original training dataset and generates a number of m bootstrapped samples according to the number of mappers employed. Each sample is saved in one data chunk in the HDFS. The data structure for each saved training instance in the data chunk is defined as below:

$$\{\mathrm{instance}_i, \mathrm{class}_j, \mathrm{instancetype}\},$$

where $\mathrm{instance}_i$ represents the ith instance in a data chunk; class_j represents the jth class that $\mathrm{instance}_i$ belonged to; instancetype field is filled a string "training" to inform the algorithm that $\mathrm{instance}_i$ is a training instance.

Afterward, the parallelization phase starts. Each mapper firstly initializes the BPNN algorithm and then inputs one data chunk. Therefore the instances saved in the data chunk can be finally input into the mapper one by one. If the instance type is "training," the BPNN in the mapper starts the training phase using the instance. In this case, $\mathrm{instance}_i$ is employed to execute the feed-forward phase using (1) and (2) while class_j is employed to execute the back propagation phase using (3) to (4). As long as all the instances marked as "training" have been processed, the BPNN has been trained. As a result, a number of m trained classifiers (mappers) are created in the Hadoop cluster.

In the classification phase, each testing $\mathrm{instance}_t$ is input into all m mappers. In each mapper, $\mathrm{instance}_t$ is classified by

Table 2: Input features of stability and margin of HBPNN.

Symbol	Definition
P_i^{acc}	The accelerating power of each generator at the time of one cycle after the fault clearing
$\left(\frac{dv_i}{dt}, \frac{d\delta_i}{dt}\right)$	The rate of change of both bus voltage and angles of each generator [9]
KE_{sum}	The total value of generators' kinetic energy at the time of one cycle after fault clearing, given in [30]
ISGA	An integral square generator angle index given in [28]
RTI_{max}	Maximal RTI index [31] in the interval from T_{cl} to T_e
V_{area}	The maximal integral area of voltage amplitude variation of all the generator busbars, given in (11)

the BPNN using (1) and (2). And then the mapper outputs an intermediate output in {key-value} form:

$$\{\text{instance}_t, \text{class}_l\},$$

where class_l denotes the classification result of instance_t of one mapper, so that m mappers output m outputs.

HBPNN starts one reducer to collect the intermediate outputs from m mappers. After sorting and merging, a collection which contains m classified results for the instancet is formed.

The collection of instance_t $\Rightarrow$ $\{\text{instance}_t, \text{class}_1\}$, $\{\text{instance}_t, \text{class}_2\}$, $\{\text{instance}_t, \text{class}_3\}$, $\{\text{instance}_t, \text{class}_j\}$, $\vdots$, $\{\text{instance}_t, \text{class}_l\}$ $\Rightarrow$ m classified results

Inside the collection, majority voting is executed to select the final classification result which is ultimately output in the form of

$$\{\text{instance}_t, \text{result}\},$$

where result represents the final classification result. The pseudo code of HBPNN is shown by Algorithm 1.

Algorithm 1 (HBPNN).

In the training phase

(1) HBPNN generates a number of m bootstrapped training samples which are saved in m data chunks in HDFS.

(2) Each data chunk is input into one mapper.

(3) Each mapper initializes one BPNN.

(4) For each mapper:

BPNN inputs one instance instance_i.

If instance_i is a "training" instance

BPNN trains its parameters

Until all the training instances are processed.

In the classification phase

(5) For each testing instance instance_t:

All the m mappers input instance_t.

BPNN in each mapper executes feed-forward to classify instance_t.

Each mapper outputs $\{\text{instance}_t, \text{class}_l\}$.

(6) One reducer collects the classified results of instance_t from all mappers.

(7) In the reducer, a collection of instance_t is formed:

(8) Majority voting is executed in the reducer to select the ultimate classification result for instance_t.

(9) Until all the testing instances are classified, algorithm terminates.

3.6. Feature Selection. Assume that a PMU has been deployed on each generator bus; full parameter trajectories of generators as well as related indices proposed in previous literatures can be introduced as features. However, many features are strongly correlated with others. Therefore, the Pearson correlation coefficients (PCC) method [33] is used to reduce the redundancy of statistical index-based features. Any two features d_1 and d_2 satisfying $|\text{PPC}_{d_1 d_2}| > 0.85$ condition are regarded to be highly correlated. Tables 2 and 3 illustrate the selected features fed to train HBPNN for the CUGs and global stability, respectively. Specifically, the size of the time window used to observe features is from the fault clearing time T_{cl} to the following 10 cycles represented as T_e.

Beside the referred features, Tables 2 and 3 also include two defined indices, V_{area} and ISGS_i, which can be formulated as follows:

$$V_{\text{area}} = \max\left\{\int_{T_{\text{cl}}}^{T_e} \left(V_i^0 - V_i(t)\right) dt\right\}, \tag{11}$$

$$\text{ISGS}_i = \int_{T_{\text{cl}}}^{T_e} M_i \left[\omega_i(t) - \omega_{\text{COI}}(t)\right]^2 dt, \tag{12}$$

where $\omega_i(t)$ and $\omega_{\text{COI}}(t)$ represent rotor speed of generator i and COI at the time point t, respectively, T_{cl} is the time point of fault clearing, and T_e represents the time window used to observe the features.

3.7. Automated Sample Generation. In this work, a random fault simulator has been developed to generate massive samples [34]. Random fault refers to stochastic three-phase short circuits of any transmission lines. In addition, fault

Table 3: Input features of CUGs of HBPNN.

Symbol	Definition
$KE_i^{T_{cl}}$	The kinetic energy of this generator at the time of one cycle after the fault clearing, which is given in [30]
$ISGS_i$	The integral area of rotor speed deviation between generator i and COI, which is given in (12)
$\delta_i^{COI}(t)$	The absolute value of rotor angle deviate between generator i and COI at each cycle point from T_{cl} to T_e
$V_i(t)$	The voltage amplitude of generator i at each cycle point from T_{cl} to T_e including prefault value

clearing time is randomly set to 0.1 s to 0.35 s. The samples generation is listed as below:

(1) Load base case: if the initial outage exists, trip the component and calculate power flow.

(2) Change P and Q on each bus by multiplying a random number in the range of [0.8, 1.4] to simulate the load level, distributing unbalance load to all the generators in proportion to their base generation.

(3) Implement three-phase fault on a randomly selected component at time T_f, and clear fault at $T_f + \mu$, where μ is a random decimal in [0.1, 0.35].

(4) Perform time-domain simulation for the above randomly configured operation and fault scenario, and collect output trajectories to calculate features defined in Tables 2 and 3 as well as the related targets.

3.8. The Architecture of HBPNN. After random faults simulation is accomplished, the entire samples are stored in HDFS. HBPNN separates the training data into pieces and employs bootstrapping to generate bootstrapped samples. Each piece is saved in one data chunk. And then HBPNN initializes distributed neural networks in multiple mappers. These networks can be categorized into three types, the CUG identification, stability assessment, and margin assessment. Afterwards, each mapper inputs one data chunk and executes the training for the large-scale input data. As long as the stability, margin, and CUG networks are sufficiently trained, they can be utilized as the enhanced classifiers of TSA. When the testing data is fed into HBPNN, the parallel neural network can efficiently classify each instance and output into its final classification. Figure 3 shows the overall architecture of HBPNN.

4. Experimental Results

4.1. HBPNN Validation. In order to evaluate the performance of HBPNN, a number of experiments have been carried out in a physical Hadoop computer cluster with 1 Gbps network bandwidth. The cluster contains five nodes, in which 4 nodes are DataNodes and the other one is NameNode. The deployed frameworks are Hadoop and HaLoop. In addition, the cluster configurations and details of the generated dataset are listed in Tables 4 and 5, respectively.

Table 4: Cluster detail.

	CPU	Memory	SSD	OS
NameNode	Core i7@3 GHz	8 GB	750 GB	Fedora
DataNodes	Core i7@3.8 GHz	32 GB	250 GB	Fedora

Table 5: Dataset detail.

Data	Instance length	Number of classes	Output
CUG	24	2	[0, 0] and [0, 1]

As each input in input layer of HBPNN only accepts the value between 0 and 1, each instance is normalized before inputting into HBPNN. For one instance $\text{instance}_k = \{a_1 \ a_2 \ a_3 \ \cdots \ a_{in}\}$, let $a_{\max}$, $a_{\min}$, and na_i denote the maximum element, minimum element, and normalized a_i, respectively, and then

$$na_i = \frac{a_i - a_{\min}}{a_{\max} - a_{\min}}. \tag{13}$$

The precision p can be calculated using

$$p = \frac{r}{r + w} \times 100\%, \tag{14}$$

where r and w represent the number of correctly classified and wrongly classified instances, respectively.

4.1.1. Precision Validation. In the experiments 1000 training instances and 1000 testing instances were generated. Ten mappers were employed and each of them processed the training instances varied from 10 to 1000. Figure 4(a) shows that the accuracy of HBPNN increases with an increasing number of training instances. Figure 4(a) also indicates that when the number of training instances is small, the HBPNN based on bootstrapping sampling outperforms the original BPNN in terms of accuracy.

Figure 4(b) shows the stability of HBPNN in processing small numbers of training instances for five times. This experiment focuses on the algorithm stability. In the tests, HBPNN and the original BPNN were trained by only ten instances. Although a low number of training instances leads to low accuracy, the results show HBPNN is more stable than BPNN in all the five cases. And even with such a low number of the training instances, HBPNN can also give higher accuracy than the standalone BPNN.

4.1.2. Computation Efficiency. A number of tests were conducted to evaluate the efficiency of HBPNN in computation using Hadoop and HaLoop, respectively. It can be observed from Figure 5(a) that, along with an increasing size of data, the parallel HBPNN performs faster than the standalone BPNN. It is worth noting that the HaLoop based HBPNN is slightly faster than the Hadoop based HBPNN due to the reduced computation overhead in dealing with iterations which is further illustrated in Figure 5(b).

4.2. HBPNN Application. HBPNN was applied in two power system cases. The first case is a 68-node testing system

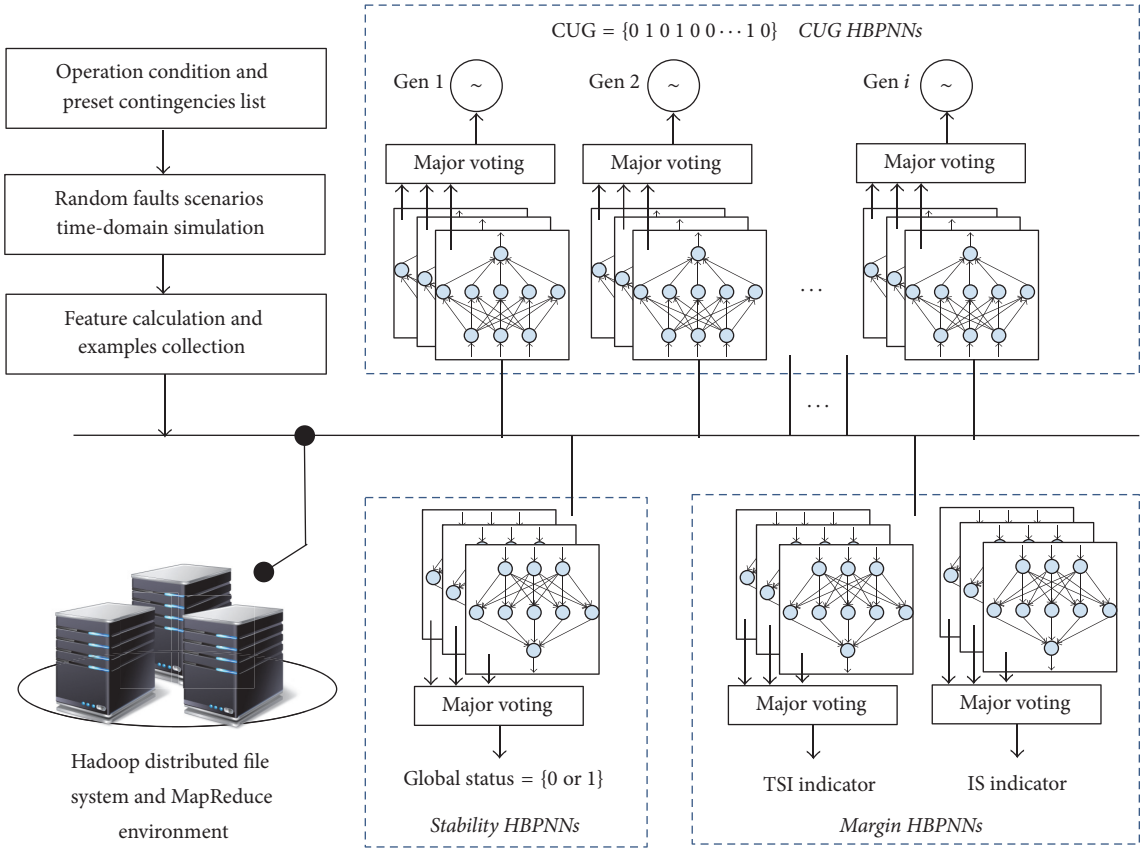
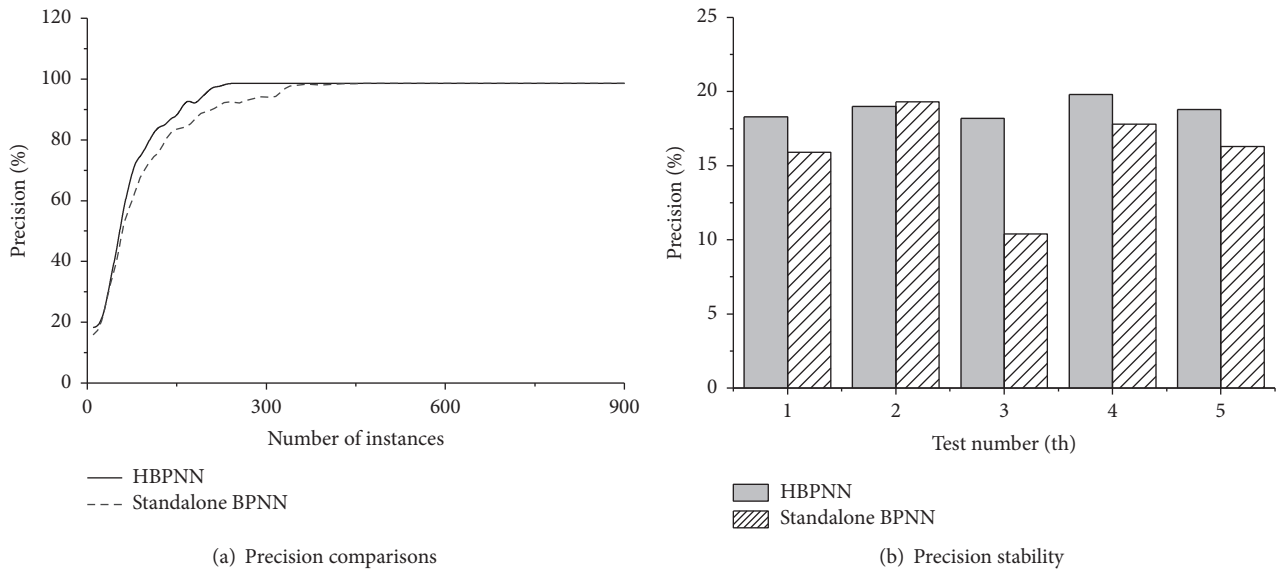


Figure 3: The architecture of HBPNN.



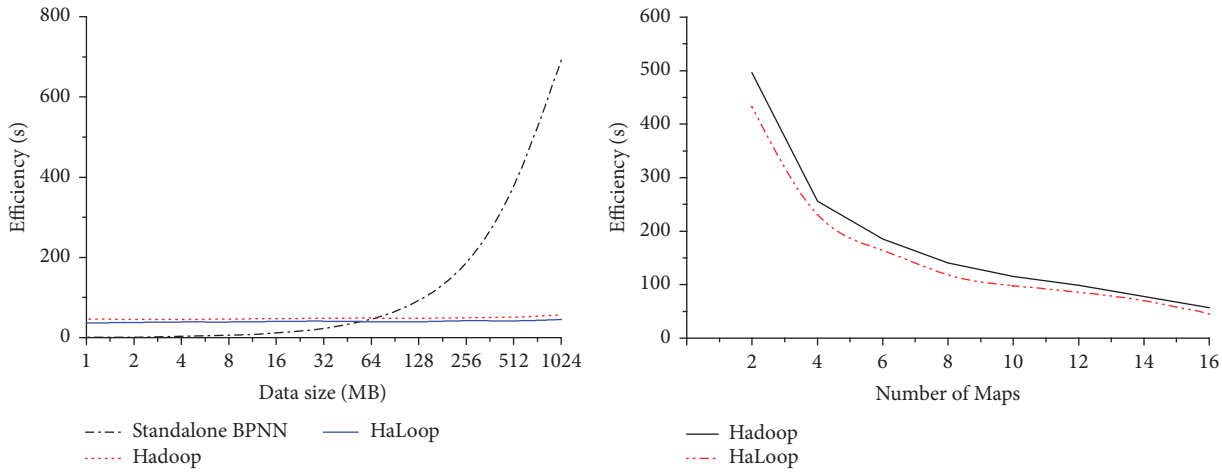
(a) Precision comparisons

(b) Precision stability

Figure 4: Comparison between HBPNN and standalone BPNN.

including 16 generators. The second case is a real power system of Sichuan Grid in China, which has 878 busbars, 1096 lines, and 109 generators. The details of the data samples are listed in Table 6. The configurations of HBPNN are shown in Table 7.

In this evaluation, the algorithm precision of the generators status prediction is tested. In terms of precision, when the number of training instances is large, the presented algorithm HBPNN has the same precision compared to that of the standalone HBPNN. Therefore, Figure 6 only lists



(a) Increasing data sizes

(b) Efficiency of Hadoop and HaLoop

Figure 5: Efficiency validation in different distributed platforms.

Table 6: Generated data of test system.

	68-node system		Sichuan power grid	
	Instance number	Data size (MB)	Instance number	Data size (MB)
CUG	12000	93.75	12000	638.78
Stability	12000	11.72	12000	118.03
Margin	12000	11.79	12000	117.62

Table 7: Details of HBPNN for the test.

Scenario	Mapper number	BPNN number	Input 68-node/Sichuan	Hidden layer neurons	Output
CUG	8	8	24/24	15	2
Stability	4	4	52/331	15	2
Margin	4	4	52/331	15	2

the precision of the HBPNN without comparison with a standalone HBPNN algorithm.

Figure 6 recording the CUGs predicting precision of test systems indicates that HBPNN is of satisfactorily high precision in identifying the generators transient status during the postfault trajectories of the power system. The average precisions for all generators of the two test systems are 99.19% and 98.63%, respectively.

In order to validate the feasibility of HBPNN in these two cases, 2400 new samples including random multiple faults scenarios were simulated for each testing system. The details of the sample sets are shown in Table 8.

Figure 7 shows the two example scenarios of the Sichuan grid in the status of stable and unstable cases, respectively. The features related trajectories in 10 cycles were fed into the trained HBPNN, which is able to quickly provide predicted values of the concerned targets. Table 9 shows that HBPNN accurately classifies the two scenarios. In addition, Figure 8 illustrates the accuracy of HBPNN of processing 2400 samples generated by the respective testing systems. It can be observed that the accuracy of the algorithm is more than 90%.

Table 8: Details of new testing samples.

	68-node system		Sichuan power grid	
	Stable	Unstable	Stable	Unstable
N-1	688	112	758	42
N-2	621	179	682	118
N-k ($k \geq 3$)	436	364	523	277

Figure 9 shows that the parallel HBPNN is more efficient than the standalone BPNN in the two testing power systems when the size of data samples is large as shown in Figure 9(c). However, the parallel HBPNN is slower than the standalone BPNN when the size of data is small as shown in Figures 9(a) and 9(b) due to the fact that both Hadoop and HaLoop have extra system overheads. Nevertheless, the HaLoop parallelized HBPNN is always faster than the Hadoop parallelized HBPNN due to the reduced computation overhead in dealing with iterations.

Table 9: Comparison of target and HBPNN output for two test scenarios.

	Scenarios 1 (stable)		Scenarios 2 (unstable)	
	Target output	HBPNN output	Target output	HBPNN output
CUG	Null	Null	7, 9, 11, 64	7, 9, 11, 64
Stability	1	1	0	0
TSI	39.69	41.76	−96.74	−94.62
IS	1.396	1.329	1.0063	1.0027

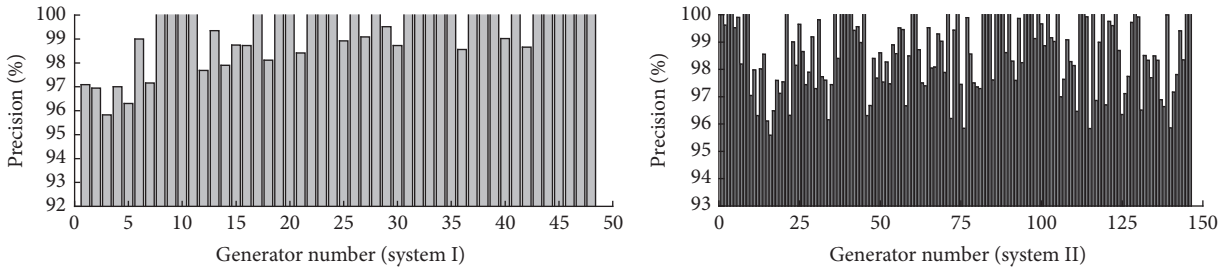


Figure 6: Precision of predicted CUGs.

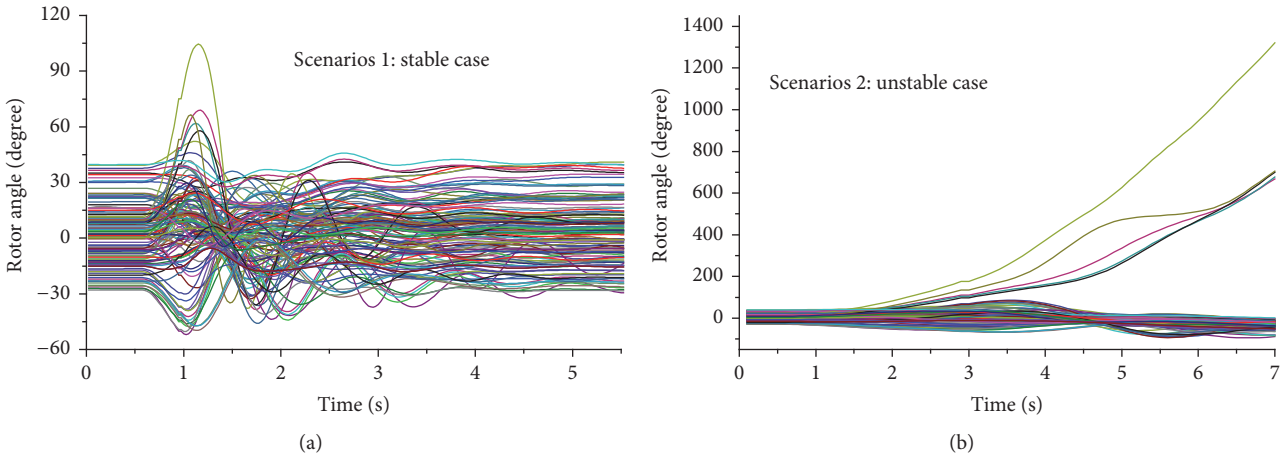


Figure 7: The rotor angle trajectories of two applied scenarios of Sichuan grid.

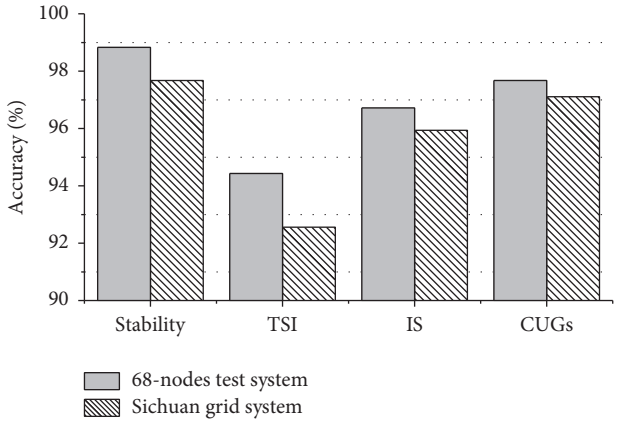


Figure 8: The accuracy of HBPNN in classification in the two testing systems.

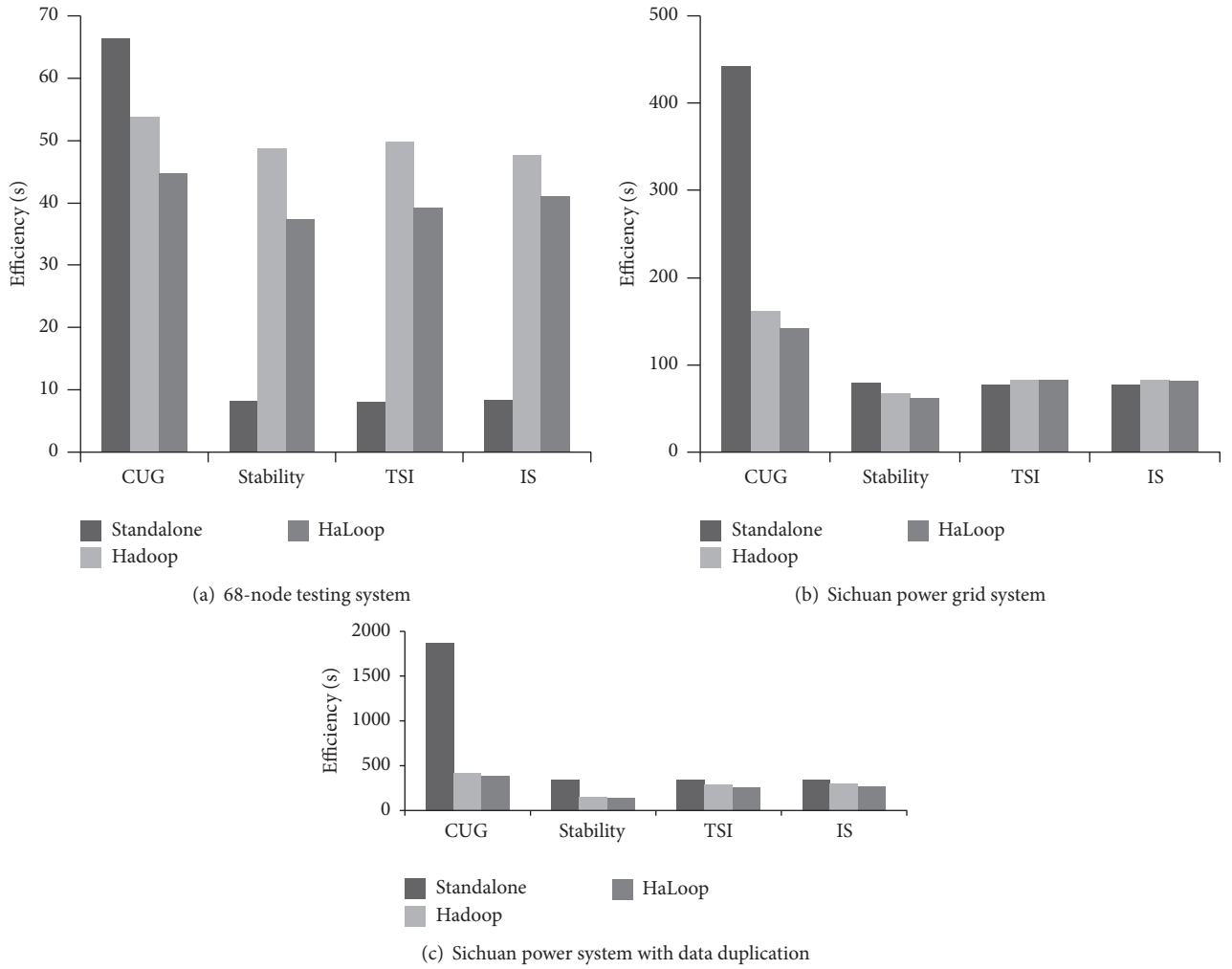


Figure 9: The computation efficiency of HBPNN on the testing power systems.

5. Conclusion

In this paper we have presented HBPNN, a high performance distributed neural network algorithm for fast stability assessment in power systems. HBPNN is designed using Hadoop to train large-scale training data in parallel to speed up the training process. It further employs HaLoop to reduce the iterative overhead that occurred in the training process. HBPNN also employs ensemble techniques to maintain high accuracy in parallelized classification. The work in this paper is able to establish highly scalable computing architecture to enable comprehensive transient stability awareness technique, including global stability prediction, stable margin estimation, and CUGs detection.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (NSFC Project, nos. 51207098 and 51437003).

References

[1] M. Negnevitsky, D. H. Nguyen, and M. Piekutowski, "Risk assessment for power system operation planning with high wind power penetration," *IEEE Transactions on Power Systems*, vol. 30, no. 3, pp. 1359–1368, 2015.

[2] L. Wehenkel, "Machine-learning approaches to power-system security assessment," *IEEE Expert*, vol. 12, no. 5, pp. 60–72, 1997.

[3] F. R. Gomez, A. D. Rajapakse, U. D. Annakkage, and I. T. Fernando, "Support vector machine-based algorithm for post-fault transient stability status prediction using synchronized measurements," *IEEE Transactions on Power Systems*, vol. 26, no. 3, pp. 1474–1483, 2011.

[4] I. Kamwa, S. R. Samantaray, and G. Joos, "Development of rule-based classifiers for rapid stability assessment of wide-area post disturbance records," in *Proceedings of the IEEE Power and Energy Society General Meeting*, Minneapolis, Minn, USA, July 2010.

[5] Y. Xu, Z. Y. Dong, J. H. Zhao, P. Zhang, and K. P. Wong, "A reliable intelligent system for real-time dynamic security assessment of power systems," *IEEE Transactions on Power Systems*, vol. 27, no. 3, pp. 1253–1263, 2012.

[6] C. Sturk, L. Vanfretti, Y. Chompoobutrgool, and H. Sandberg, "Coherency-independent structured model reduction of power systems," *IEEE Transactions on Power Systems*, vol. 29, no. 5, pp. 2418–2426, 2014.

[7] H. You, V. Vittal, and X. Wang, "Slow coherency-based islanding," *IEEE Transactions on Power Systems*, vol. 19, no. 1, pp. 483–491, 2004.

[8] A. Vahidnia, G. Ledwich, E. Palmer, and A. Ghosh, "Generator coherency and area detection in large power systems," *IET Generation, Transmission & Distribution*, vol. 6, no. 9, pp. 874–883, 2012.

[9] F. Hashiesh, H. E. Mostafa, A.-R. Khatib, I. Helal, and M. M. Mansour, "An intelligent wide area synchrophasor based system for predicting and mitigating transient instabilities," *IEEE Transactions on Smart Grid*, vol. 3, no. 2, pp. 645–652, 2012.

[10] A. N. Al-Masri, M. Z. A. Ab Kadir, H. Hizam, and N. Mariun, "A novel implementation for generator rotor angle stability prediction using an adaptive artificial neural network application for dynamic security assessment," *IEEE Transactions on Power Systems*, vol. 28, no. 3, pp. 2516–2525, 2013.

[11] http://hadoop.apache.org/.

[12] S. Prasad and S. B. Avinash, "Smart meter data analytics using OpenTSDB and Hadoop," in *Proceedings of the 2013 IEEE Innovative Smart Grid Technologies—Asia (ISGT Asia '13)*, November 2013.

[13] M. Khan, P. M. Ashton, M. Li, G. A. Taylor, I. Pisica, and J. Liu, "Parallel detrended fluctuation analysis for fast event detection on massive pmu data," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 360–368, 2015.

[14] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "HaLoop: efficient iterative data processing on large clusters," in *Proceedings of the International Conference on Very Large Data Bases*, Singapore, 2010.

[15] J. De La Ree, V. Centeno, J. S. Thorp, and A. G. Phadke, "Synchronized phasor measurement applications in power systems," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 20–27, 2010.

[16] J. M. G. Alvarez and P. E. Mercado, "Online inference of the dynamic security level of power systems using fuzzy techniques," *IEEE Transactions on Power Systems*, vol. 22, no. 2, pp. 717–726, 2007.

[17] Y. V. Makarov, P. Du, S. Lu et al., "PMU-based wide-area security assessment: concept, method, and implementation," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1325–1332, 2012.

[18] M. Rizwan, M. Jamil, and D. P. Kothari, "Generalized neural network approach for global solar energy estimation in India," *IEEE Transactions on Sustainable Energy*, vol. 3, no. 3, pp. 576–584, 2012.

[19] Y. Wang, B. Li, R. Luo, Y. Chen, N. Xu, and H. Yang, "Energy efficient neural networks for big data analytics," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pp. 1–2, Dresden, Germany, March 2014.

[20] A. A. Ikram, S. Ibrahim, M. Sardaraz, M. Tahir, H. Bajwa, and C. Bach, "Neural network based cloud computing platform for bioinformatics," in *Proceedings of the 9th Annual Conference on Long Island Systems, Applications and Technology (LISAT '13)*, pp. 1–6, IEEE, Farmingdale, NY, USA, May 2013.

[21] V. Rao and S. Rao, "Application of artificial neural networks in capacity planning of cloud based IT infrastructure," in *Proceedings of the 1st IEEE International Conference on Cloud Computing for Emerging Markets (CCEM '12)*, pp. 38–41, Bengaluru, India, October 2012.

[22] A. A. Huqqani, E. Schikuta, and E. Mann, "Parallelized neural networks as a service," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '14)*, pp. 2282–2289, Beijing, China, July 2014.

[23] R. Gu, F. Shen, and Y. Huang, "A parallel computing platform for training large scale neural networks," in *Proceedings of the IEEE International Conference on Big Data*, pp. 376–384, IEEE, Silicon Valley, Calif, USA, October 2013.

[24] Z. Liu, H. Li, and G. Miao, "MapReduce-based backpropagation neural network over large scale mobile data," in *Proceedings of the 6th International Conference on Natural Computation (ICNC '10)*, pp. 1726–1730, IEEE, Yantai, China, August 2010.

[25] P. M. Anderson and A. A. Fouad, *Power System Control and Stability*, IEEE, Piscataway, NJ, USA, 2nd edition, 2003.

[26] http://www.eps.ee.kth.se/personal/vanfretti/pst/Power_System_Toolbox_Webpage/PST.html.

[27] A. D. Rajapakse, F. Gomez, K. Nanayakkara, P. A. Crossley, and V. V. Terzija, "Rotor angle instability prediction using post-disturbance voltage trajectories," *IEEE Transactions on Power Systems*, vol. 25, no. 2, pp. 947–956, 2010.

[28] G. Li and S. M. Rovnyak, "Integral square generator angle index for stability ranking and control," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 926–934, 2005.

[29] Y. Liu, Y. Liu, J. Liu, M. Li, Z. Ma, and G. Taylor, "High-performance predictor for critical unstable generators based on scalable parallelized neural networks," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 3, pp. 414–426, 2016.

[30] N. K. Alham, *Parallelizing support vector machines for scalable image annotation [Ph.D. thesis]*, Brunel University, England, UK, 2011.

[31] K. Verma and K. R. Niazi, "A coherency based generator rescheduling for preventive control of transient stability in power systems," *International Journal of Electrical Power & Energy Systems*, vol. 45, no. 1, pp. 10–18, 2013.

[32] Y. Liu, J. Yang, Y. Huang, L. Xu, S. Li, and M. Qi, "MapReduce based parallel neural networks in enabling large scale machine learning," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 297672, 13 pages, 2015.

[33] J. Benesty, "Pearson correlation coefficient," in *Noise Reduction in Speech Processing*, Springer, Berlin, Germany, 2009.

[34] K. Meng, Z. Y. Dong, K. P. Wong, Y. Xu, and F. J. Luo, "Speed-up the computing efficiency of power system simulator for engineering-based power system transient stability simulations," *IET Generation, Transmission & Distribution*, vol. 4, no. 5, pp. 652–661, 2010.