Hindawi Publishing Corporation
International Journal of Photoenergy
Volume 2013, Article ID 938162, 8 pages
http://dx.doi.org/10.1155/2013/938162

Research Article

Application of CMAC Neural Network to Solar Energy Heliostat Field Fault Diagnosis

Neng-Sheng Pai, Her-Terng Yau, Tzu-Hsiang Hung, and Chin-Pao Hung

Department of Electrical Engineering, National Chin-Yi University of Technology, Taichung 41170, Taiwan

Correspondence should be addressed to Her-Terng Yau; pan1012@ms52.hinet.net

Received 14 August 2012; Accepted 16 November 2012

Academic Editor: Mahmoud M. El-Nahass

Solar energy heliostat fields comprise numerous sun tracking platforms. As a result, fault detection is a highly challenging problem. Accordingly, the present study proposes a cerebellar model arithmetic computer (CMAC) neutral network for automatically diagnosing faults within the heliostat field in accordance with the rotational speed, vibration, and temperature characteristics of the individual heliostat transmission systems. As compared with radial basis function (RBF) neural network and back propagation (BP) neural network in the heliostat field fault diagnosis, the experimental results show that the proposed neural network has a low training time, good robustness, and a reliable diagnostic performance. As a result, it provides an ideal solution for fault diagnosis in modern, large-scale heliostat fields.

1. Introduction

Heliostat fields play an essential role in concentrating the solar irradiation in tower solar power plant systems [1]. A modern heliostat field comprises hundreds or even thousands of individual heliostats, each of which is adjusted continuously so as to direct the incident light onto the receiver [2, 3]. However, the transmission systems of the heliostats are prone to various vibration, temperature, and rotational speed errors; and thus, the overall efficiency of the solar power plant is reduced [4–6]. Due to the sheer scale of modern heliostat fields, fault diagnosis is a highly challenging problem. Moreover, even if a faulty heliostat is successfully located, determining the precise nature of the fault is not easily achieved using traditional linear mapping techniques [7]. In a recent study, Cheng-Yu et al. [8] proposed a radial basis function (RBF) neural network for fault diagnosis in heliostat fields. The results showed that the proposed system had a better classification performance than diagnostic systems based on a back propagation (BP) neural network [9] or hybrid artificial neural network (ANN)/genetic algorithm (GA) scheme [10]. However, in both cases, the improved detection performance was obtained at the expense of a longer training time.

Cerebellar model arithmetic computer (CMAC) neural networks are capable of classifying highly complex non linear dynamic systems with a high degree of accuracy and a short learning time. As a result, CMACs have found extensive use for fault diagnosis in such systems as automobile engines, internal combustion engines, and generators [11, 12]. Multilayer perceptron (MLP) or RBF neural networks achieve a mapping between the input and output data by means of a systematic weighting and aggregation of the excitation function outputs of multiple nodes configured in a small number of hidden layers. Such networks have a small scale, but incur a long computational time due to the hierarchical nature of the layer-by-layer mapping process. By contrast, in CMAC neural networks, the mapping process is performed by directly summing multiple weightings stored in the memory lattice. In other words, the mapping process involves only a simple addition operation. As a result, although a larger memory lattice is required to achieve the same mapping ability as that obtained using an MLP or RBF approach, the mapping time is significantly reduced. Thus, CMACs have emerged as a highly attractive solution for real-time fault diagnosis in complex, nondynamic systems [13].

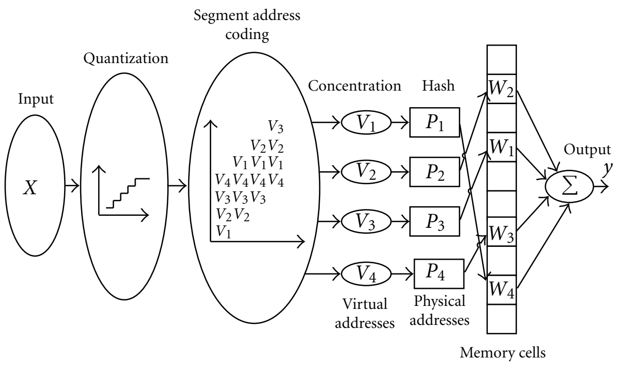

Figure 1: Schematic diagram of CMAC neural network.

Table 1: Gear fault modes.

Fault mode	Fault description	Output expression ($T1 - T7$)
A	Normal gear	(1, 0, 0, 0, 0, 0, 0)
B	Gear spalling	(0, 1, 0, 0, 0, 0, 0)
C	Gear wear	(0, 0, 1, 0, 0, 0, 0)
D	Gear scoring	(0, 0, 0, 1, 0, 0, 0)
E	Gear breaking	(0, 0, 0, 0, 1, 0, 0)
F	Pitting of tooth flank	(0, 0, 0, 0, 0, 1, 0)
G	Plastic deformation of tooth flank	(0, 0, 0, 0, 0, 0, 1)

2. Overview of CMAC Neural Networks

In CMAC neural networks, each memory address within the memory lattice stores a particular weighting value. During the training process, an input sample (vector) is selected and input to the CMAC, where it is quantized and encoded. The encoded vector is then processed by a hash function and used to excite a particular subset of the memory addresses within the lattice [14, 15]. The output vector corresponding to the input vector is then obtained by simply summing the weightings stored in the excited memory addresses. The output vector is compared with the ideal (target) output vector, and the weightings stored in the excited memory addresses are tuned by allocating the error between the two vectors equally among them. The process is then repeated using a new training sample. Given the input of a signal with an identical form to that of one of the previous samples, the same set of memory addresses is excited once again and the signal obtained by summing the tuned weighting values stored in the excited memory addresses is equal to the ideal output signal. Given the input of a signal vector containing noise, the similarity between the new input signal and the original signal is reduced. As a consequence, only some of the original memory addresses may be excited once again. Nonetheless, the output signal obtained by summing the weightings in the excited addresses retains many of the characteristics of the original output signal. As for the original training sample, the weightings of the memory addresses excited by the distorted signal are further tuned by distributing the error between the distorted output signal and the ideal output signal evenly among them. Having retuned the memorized weightings, the subsequent input of a signal vector with a similar degree of distortion yields an output signal with a form close to that of the target output signal.

Figure 1 illustrates the basic framework of a CMAC neural network [16]. Assume that the number of excited memory addresses (A^*) is equal to 4. Assume further that an input vector x_1 is applied to the input nodes of the CMAC. The weightings stored in the excited memory addresses are thus updated as w_1, w_2, w_3, and w_4, respectively. The output vector is obtained by summing the updated weightings of the four addresses and is then compared with the ideal output value. As described above, the error between the two vectors is allocated equally among the four weightings such that the subsequent reinput of the same signal yields the ideal output vector. Given the input of a new signal x_2 similar to x_1, the excited memory addresses may be w_1, w_2, w_3, and w_4 once again or may be w_1, w_2, w_3, and w_5 (e.g.). Generally speaking, the greater the degree of similarity between the memory addresses excited by two different input signals, the greater the degree of similarityy between the two output signals. In the present example, three of the memory addresses excited by signal x_2 are identical to those excited by signal x_1 (i.e., w_1, w_2, and w_3). Thus, assuming that the tuning process has been completed, the output signal will be close to the original output value.

3. Design of Proposed CMAC Fault Diagnosis System

As described in Section 1, a heliostat field typically contains hundreds if not thousands of individual heliostats, each with its own controller. The heliostat field monitoring system communicates with all of the controllers in the heliostat field and performs fault diagnosis on the basis of the information received. Of all the various components within each heliostat, the turning gear is most commonly affected by faults and errors [10]. Thus, in developing the proposed CMAC neural network, the present study focuses specifically on the transmission system of each heliostat.

3.1. Fault Modes. The gears within the heliostat transmission system experience a range of common faults, including spalling, wear, scoring, breakage, pitting, and plastic deformation. As shown in Table 1, these gear faults are annotated in this study as fault modes B ~ G, respectively. (Note that fault mode A denotes a normal gear operation.) For each fault mode, the output signal of the CMAC neural network has the form of a 7-element vector ($T1$, $T2$, $T3$, $T4$, $T5$, $T6$, $T7$). Note that for each vector, an element value of "0" indicates a normal output, while an element value of "0" indicates an abnormal output (i.e., a fault).

3.2. Selection of Sensing Points for Fault Diagnosis Purposes. Heliostats utilize a double-shaft transmission mechanism to accomplish rotational adjustments of the mirror in the azimuth and elevation planes, respectively. As shown in

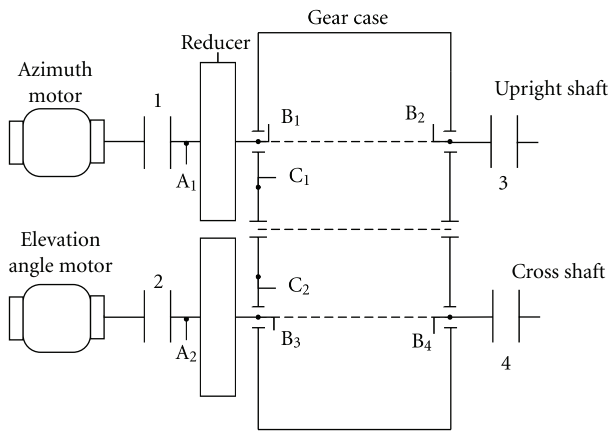

FIGURE 2: Sensor positions in heliostat transmission system.

Figure 2, the major components in each branch of the transmission system include an asynchronous motor, a reducer, a gear-cross shaft, and a gear upright shaft. Note that in Figure 2, labels 1, 2, 3, and 4 denote mechanical couplings; A_1 and A_2 are current vortex sensors, detecting the rotational speed of the corresponding motor shaft; B_1, B_2, B_3, and B_4 are acceleration sensors, detecting the gear vibration; and C_1 and C_2 are temperature sensors, detecting the gear case temperature. (Note that the gear case temperature is taken as an indication of the oil temperature within the case.) The analog signals generated by the velocity, vibration, and temperature sensors are amplified, converted to a digital form, and then transferred to the monitoring system for fault diagnostic purposes.

In the transmission system shown in Figure 2, the engagement vibration of the gears is transferred to the bearing via the shaft and is subsequently transferred to the gear case via the bearing pedestal. To measure the vibration accurately, the acceleration sensor should be attached to a position of high stiffness. Thus, in the present study, the acceleration sensors were attached to the bearing pedestals in a vertical direction. Furthermore, as shown in Figure 2, the current vortex sensors used to detect the rotational speed of the asynchronous motors were attached to the output side of the respective couplers. For each sensor in Figure 2, the measured signal was normalized as follows:

$$y_i = \frac{(x_i - x_{\min})}{(x_{\max} - x_{\min})}. \tag{1}$$

The normalized measurement data obtained from the rotational speed sensors (A_1, A_2), vibration sensors (B_1, B_2, B_3, B_4) and temperature sensors (C_1, C_2) were then used to construct an 8-element input vector for the CMAC neural network for fault diagnosis purposes.

4. CMAC Neural Network Training

Table 2 shows the sample data used to train the CMAC neural network. Table 3 shows the corresponding fault modes of the 20 training samples. In implementing the CMAC neural network model shown in Figure 1, the quantization step size is specified as 64 bits and the encoded fault following quantization has a length of 48 bits. Since there are seven output classes (see Table 1), a total of six memory layers are required. Furthermore, since the input signal has the form of an 8-element vector (see Section 3), each memory layer is partitioned into eight groups, with each group having six bits.

An output value can be obtained after quantization, concatenation, excited address coding, and totaling of the excited address weightings in the CMAC neural network. In performing the training process, given a fault sample of the ith ($i = 1, 2, \ldots$) type, only the ith layer of memory is excited and trained. In the subsequent diagnosis phase, if the same addresses in each group of memory bits are excited, the fault type is identified in accordance with the output value of each layer.

4.1. Quantization. The input data for the CMAC neural network developed in the present study all fall within a given range, that is, $[X_{\min}, X_{\max}]$. As shown in Figure 3, each input data instance is quantized using an equidistant quantization scheme based on the maximum and minimum values of the corresponding range.

In performing the quantization process, the quantized value of any input value less than $X_{\min}$ is set to 0, while the quantized value of any input value greater than $X_{\max}$ is set to $X_{\max}$. The quantized values of the remaining input values are determined in accordance with

$$Q_{xi}(xi) = \frac{\operatorname{ceil}(xi - \min)}{[(\max - \min)/Q]}, \tag{2}$$

where ceil (x) is a Matlab function which causes the quantization process to yield the integer value closest to x towards infinity.

4.2. Excited Address Coding and CMAC Output Calculation. As described above, the recoded fault code following quantization comprises 48 bits, partitioned into eight 6-bit groups. Assume that the quantized fault code has the form 101000000000000000000000000100001010000011101101b. For this particular example, the eight excited addresses coded sequentially from the LSB to the MSB are as follows: $a1 = 101101\text{b} = 45$, $a2 = 000011 = 3$, $a3 = 001010 = 10$, $a4 = 000100 = 4$, $a5 = 000000 = 0$, $a6 = 000000 = 0$, $a7 = 000000 = 0$, and $a8 = 101000 = 40$. Assuming that all of the excited addresses store an initial weighting of zero, the total memory weighting excited by the first input sample is equal to 0, W_0^2, W_0^5, W_0^8, W_0^{17}, W_0^{25}. The output of the CMAC neural network can thus be expressed as

$$y = \sum_{i=1}^{A^*} W_i^{ai}, \tag{3}$$

where A^* is the total number of excited memory addresses.

4.3. Update Weightings. In the CMAC neural network developed in the present study, the weightings stored in the

Table 2: Training samples.

No.	A_1	A_2	B_1	B_2	B_3	B_4	C_1	C_2
1	0.58	0.64	0.19	0.25	0.23	0.14	0.36	0.41
2	0.65	0.89	0.16	0.25	0.48	0.35	0.38	0.45
3	0.54	0.59	0.62	0.68	0.36	0.41	0.58	0.37
4	0.59	0.64	0.39	0.42	0.61	0.54	0.35	0.60
5	0.60	0.40	0.62	0.58	0.79	0.87	0.32	0.60
6	0.55	0.64	0.12	0.20	0.35	0.40	0.45	0.39
7	0.59	0.15	0.57	0.53	0.86	0.74	0.59	0.90
8	0.53	0.70	0.13	0.24	0.32	0.41	0.36	0.61
9	0.90	0.81	0.33	0.43	0.49	0.31	0.46	0.40
10	0.55	0.59	0.61	0.68	0.57	0.51	0.69	0.62
11	0.51	0.56	0.31	0.40	0.46	0.41	0.50	0.67
12	0.10	0.21	0.85	0.79	0.86	0.73	0.76	0.88
13	0.50	0.61	0.40	0.35	0.45	0.41	0.47	0.34
14	0.33	0.45	0.80	0.76	0.91	0.81	0.59	0.63
15	0.53	0.58	0.32	0.41	0.09	0.15	0.30	0.41
16	0.50	0.78	0.12	0.24	0.35	0.31	0.42	0.46
17	0.32	0.41	0.75	0.81	0.56	0.67	0.62	0.40
18	0.53	0.61	0.22	0.13	0.23	0.24	0.39	0.43
19	0.55	0.60	0.36	0.39	0.19	0.08	0.57	0.35
20	0.15	0.36	0.81	0.92	0.54	0.61	0.75	0.51

Table 3: Fault modes of training samples.

No.	Fault mode
1	(1, 0, 0, 0, 0, 0, 0)
2	(0, 1, 0, 0, 0, 0, 0)
3	(0, 0, 0, 1, 0, 0, 0)
4	(0, 0, 0, 1, 0, 0, 0)
5	(0, 0, 0, 0, 0, 0, 1)
6	(0, 0, 1, 0, 0, 0, 0)
7	(0, 0, 0, 0, 1, 0, 0)
8	(0, 0, 0, 0, 0, 1, 0)
9	(0, 1, 0, 0, 0, 0, 0)
10	(0, 0, 0, 1, 0, 0, 0)
11	(0, 0, 0, 0, 0, 1, 0)
12	(0, 0, 0, 0, 1, 0, 0)
13	(0, 0, 1, 0, 0, 0, 0)
14	(0, 0, 0, 0, 0, 0, 1)
15	(0, 0, 1, 0, 0, 0, 0)
16	(0, 1, 0, 0, 0, 0, 0)
17	(0, 0, 0, 0, 0, 0, 1)
18	(1, 0, 0, 0, 0, 0, 0)
19	(0, 0, 0, 0, 0, 1, 0)
20	(0, 0, 0, 0, 1, 0, 0)

memory lattice are updated using the method of the steepest descent [15, 16], that is,

$$W^{ai}_{(\text{new})} \longleftarrow W^{ai}_{(\text{old})} + \beta \frac{Y_d - Y}{A^*}, \quad i = 1, 2, \ldots A^*, \qquad (4)$$

Figure 3: Schematic representation of equidistant quantization scheme.

where $W^{ai}_{(\text{new})}$ is the adjusted weighting, $W^{ai}_{(\text{old})}$ is the previous weighting, ai is the excited memory address, β is the learning gain ($0 < \beta \leq 1$), Y_d is the target value (set as 1 in the present study), and Y is the actual output value. It is noted that β can be sent directly to 1 if each fault type has only one group. However, for more than one sample data, β usually has a value slightly less than 1.

The memory consumption of each layer in the CMAC is related to the number of bits per group (m). Moreover, the total memory consumption of the CMAC is determined by the number of groups, the number of the length of the encoded fault following quantization (n), and the number of fault types (f). In the CMAC developed in the present study, $A^* = 8$, $m = 6$, $n = 48$, and $f = 6$. The total number of memory addresses in the CMAC is given by

$$M_{\text{total}} = A^* \cdot f \cdot 2^m = \text{ceil}\left(\frac{n}{m}\right) \cdot f \cdot 2^m. \qquad (5)$$

Thus, the total number of memory addresses in the present CMAC is equal to $8 \times 6 \times 32 = 1536$.

In (5), n/m is not guaranteed to be exactly divisible. Thus, ceil represents an unconditional carry function. In other words, the insufficient bit of MSB will fill 0 in the proper order automatically during grouping. Considering the consumption of memory, the ceil function is neglected and (5) is differentiated with respect to m to determine the

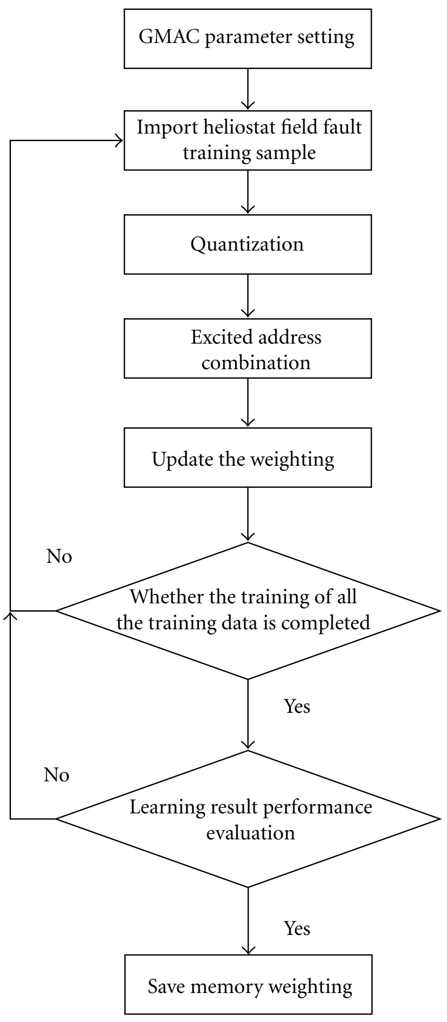
GMAC parameter setting
Import heliostat field fault training sample
Quantization
Excited address combination
Update the weighting
No
Whether the training of all the training data is completed
Yes
No
Learning result performance evaluation
Yes
Save memory weighting

FIGURE 4: Flowchart of offline learning process.

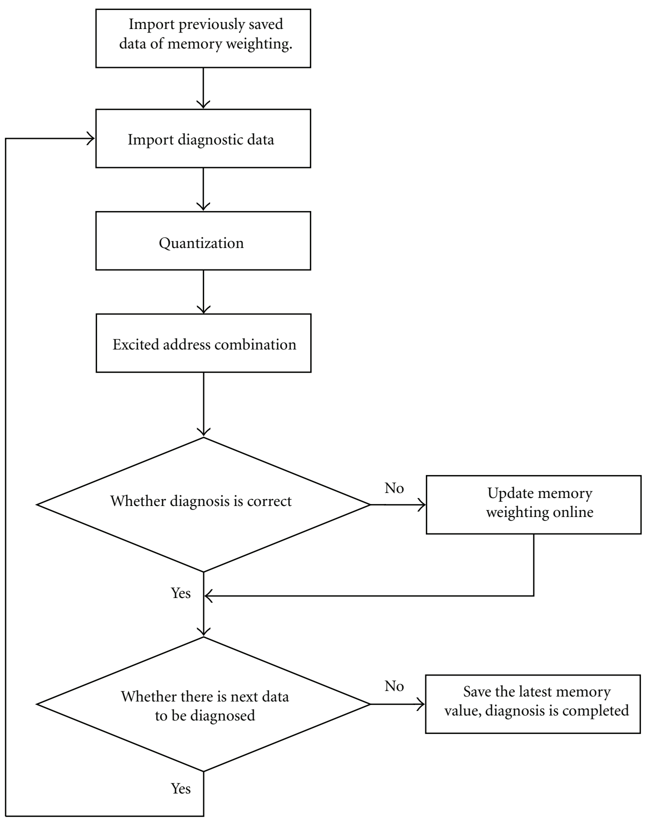
Import previously saved data of memory weighting.
Import diagnostic data
Quantization
Excited address combination
Whether diagnosis is correct
No
Update memory weighting online
Yes
Whether there is next data to be diagnosed
No
Save the latest memory value, diagnosis is completed
Yes

FIGURE 5: Flowchart of online learning process.

number of bits per group which minimizes the total memory consumption, That is,

$$\frac{\partial M_{\text{total}}}{\partial m} = \left(-\frac{n}{m^2}\right) \cdot f \cdot 2^m + \frac{n}{m} \cdot Ln2 \cdot 2^m = 0. \quad (6)$$

4.4. Convergence of CMAC. The convergence properties of CMAC neural networks have been extensively examined in the literature [17]. In the CMAC developed in the present study, the memory consumption is reduced by means of an appropriate encoding mode which ensures that no collisions occur during the weighting process. (Note that a collision is defined here as the case where two different input signals excite the same set of memory addresses.) As a result, the convergence of the learning process is ensured.

Let the total weighting of the excited addresses on the ith ($i = 1, 2, 3, \ldots$) memory layer be equal to 1 and represent the ith fault. Furthermore, let the number of data samples corresponding to the ith fault type be equal to d. The following evaluation index is then introduced:

$$E = \sum_{i-1}^{d} (Y_i - 1)^2, \quad (7)$$

where the value of E represents the learning effect. Define ε as a number greater than 0. The training process terminates once the condition $E < \varepsilon$ is achieved.

5. Offline and Online Learning Modes

The learning modes and fault diagnosis rules proposed in this study are summarized as follows.

5.1. Offline Learning Mode .

Step 1. Determine CMAC parameter settings (i.e., quantization step size: 64, memory layer: 48 bits, 8 groups, 6 bits per group).

Step 2. Import training samples into CMAC, quantize sample data, and sum excited memory addresses to obtain output signal.

Step 3. Compare output value (y) with ideal value (Y_d) and use (4) to update memory weightings if required.

Table 4: Training samples ordered by fault type.

No.	A_1	A_2	B_1	B_2	B_3	B_4	C_1	C_2
1	0.58	0.64	0.19	0.25	0.23	0.14	0.36	0.41
2	0.53	0.61	0.22	0.13	0.23	0.24	0.39	0.43
3	0.65	0.89	0.16	0.25	0.48	0.35	0.38	0.45
4	0.90	0.81	0.33	0.43	0.49	0.31	0.46	0.40
5	0.50	0.78	0.12	0.24	0.35	0.31	0.42	0.46
6	0.55	0.64	0.12	0.20	0.35	0.40	0.45	0.39
7	0.50	0.61	0.40	0.35	0.45	0.41	0.47	0.34
8	0.53	0.58	0.32	0.41	0.09	0.15	0.30	0.41
9	0.54	0.59	0.62	0.68	0.36	0.41	0.58	0.37
10	0.59	0.64	0.39	0.42	0.61	0.54	0.35	0.60
11	0.55	0.59	0.61	0.68	0.57	0.51	0.69	0.62
12	0.59	0.15	0.57	0.53	0.86	0.74	0.59	0.90
13	0.10	0.21	0.85	0.79	0.86	0.73	0.76	0.88
14	0.15	0.36	0.81	0.92	0.54	0.61	0.75	0.51
15	0.53	0.70	0.13	0.24	0.32	0.41	0.36	0.61
16	0.51	0.56	0.31	0.40	0.46	0.41	0.50	0.67
17	0.55	0.60	0.36	0.39	0.19	0.08	0.57	0.35
18	0.60	0.40	0.62	0.58	0.79	0.87	0.32	0.60
19	0.33	0.45	0.80	0.76	0.91	0.81	0.59	0.63
20	0.32	0.41	0.75	0.81	0.56	0.67	0.62	0.40

Table 5: Fault modes of ordered training samples.

No.	Fault mode
1	(1, 0, 0, 0, 0, 0, 0)
2	(1, 0, 0, 0, 0, 0, 0)
3	(0, 1, 0, 0, 0, 0, 0)
4	(0, 1, 0, 0, 0, 0, 0)
5	(0, 1, 0, 0, 0, 0, 0)
6	(0, 0, 1, 0, 0, 0, 0)
7	(0, 0, 1, 0, 0, 0, 0)
8	(0, 0, 1, 0, 0, 0, 0)
9	(0, 0, 0, 1, 0, 0, 0)
10	(0, 0, 0, 1, 0, 0, 0)
11	(0, 0, 0, 1, 0, 0, 0)
12	(0, 0, 0, 0, 1, 0, 0)
13	(0, 0, 0, 0, 1, 0, 0)
14	(0, 0, 0, 0, 1, 0, 0)
15	(0, 0, 0, 0, 0, 1, 0)
16	(0, 0, 0, 0, 0, 1, 0)
17	(0, 0, 0, 0, 0, 1, 0)
18	(0, 0, 0, 0, 0, 0, 1)
19	(0, 0, 0, 0, 0, 0, 1)
20	(0, 0, 0, 0, 0, 0, 1)

Step 4. Check if all of the input training samples have been processed. If not, return to Step 2; else, proceed to Step 5.

Step 5. Evaluate the learning performance. If $E < \varepsilon$, save the current set of memory weightings; else, return to Step 2.

In practice, the time required to complete the offline learning process depends on the number of training data samples. In the present study, the CMAC is trained using just 20 samples (see Table 1). Hence, the training time is very short (less than one second). Figure 4 presents a flowchart showing the overall framework of the offline learning process.

5.2. Online Learning Mode. Having completed the offline learning process, the fault diagnosis procedure is performed as follows.

Step 6. Import memory weightings obtained in offline learning mode.

Step 7. Import diagnostic data.

Step 8. Perform quantization, binary combination coding, concatenation, and excited address coding. Sum the weightings of the excited addresses in order to obtain the output values of the various nodes.

Step 9. Compare the diagnosis result with the target result. If the diagnosis is correct, proceed to Step 10; else skip to Step 11.

Step 10. Check whether or not all of the input diagnostic data have been classified. If not all of the diagnostic data have been processed, return to Step 7; else skip to Step 12.

Step 11. Use (4) to update the memory weightings and then return to Step 10.

Table 6: Diagnosis results for training samples given in Table 4.

No.	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$
1	1.000	0	0	0	0	0	0
2	1.000	0	0	0	0	0	0
3	0	1.000	0	0	0	0	0
4	0	1.000	0	0	0	0	0
5	0	1.000	0	0	0	0	0
6	0	0	1.000	0	0	0	0
7	0	0	1.000	0	0	0	0
8	0	0	1.000	0	0	0	0
9	0	0	0	1.000	0	0	0
10	0	0	0	1.000	0	0	0
11	0	0	0	1.000	0	0	0
12	0	0	0	0	1.000	0	0
13	0	0	0	0	1.000	0	0
14	0	0	0	0	1.000	0	0
15	0	0	0	0	0	1.000	0
16	0	0	0	0	0	1.000	0
17	0	0	0	0	0	1.000	0
18	0	0	0	0	0	0	1.000
19	0	0	0	0	0	0	1.000
20	0	0	0	0	0	0	1.000

Table 7: Distorted test sample data.

No.	A_1	A_2	B_1	B_2	B_3	B_4	C_1	C_2
1	*0.69*	0.64	0.19	0.25	0.23	0.14	*0.50*	0.41
2	0.65	0.89	0.16	0.25	*0.53*	0.35	0.38	0.45
3	0.54	0.59	*0.80*	0.68	0.36	0.41	0.58	*0.48*
4	0.59	0.64	0.39	0.42	0.61	0.54	0.35	*0.72*
5	*0.84*	0.40	0.62	*0.81*	0.79	0.87	0.32	0.60
6	0.55	*0.77*	0.12	0.20	0.35	*0.52*	0.45	0.39
7	0.59	0.15	0.57	*0.63*	0.86	0.74	0.59	0.90
8	0.53	0.70	0.13	0.24	*0.35*	0.41	*0.47*	0.61
9	0.90	*0.89*	0.33	0.43	0.49	0.31	0.46	*0.52*
10	0.55	0.59	0.61	0.68	*0.63*	0.51	0.69	0.62

Table 8: Diagnosis results for distorted training samples given in Table 7.

No.	$T1$	$T2$	$T3$	$T4$	$T5$	$T6$	$T7$
1	**0.778**	0.113	0.236	0.125	0	0.111	0
2	0.111	**0.790**	0	0	0	0	0
3	0	0	0.458	**0.818**	0	0.347	0.111
4	0.222	0	0.236	**0.875**	0.1	0	0
5	0	0	0	0.216	0	0	**0.861**
6	0.111	0.226	**0.667**	0.182	0	0.125	0
7	0.111	0	0	0.125	**0.90**	0	0.111
8	0.111	0.226	0.569	0.091	0	**0.778**	0
9	0	**0.71**	0	0	0	0	0
10	0	0	0.236	**0.909**	0	0.125	0

Step 12. Save the latest memory weighting values. Terminate the diagnosis procedure.

Figure 5 presents a flowchart showing the overall framework of the online learning process.

6. Experimental Results and Analysis

For convenience, Tables 4 and 5 list the 20 training samples given in Table 2 in order of their fault type.

The learning process was repeated 10 times using the sample data given in Table 4. The corresponding diagnosis results are presented in Table 6. It appears from a comparison of Tables 5 and 6 that the CMAC achieves a 100% success rate in diagnosing the input data samples.

To evaluate the robustness of the proposed CMAC diagnosis system, the first 10 training samples in Table 2 were distorted using a random 10~40% interference signal. The resulting test samples are shown in Table 7, in which the distorted values are shown in italics.

The training procedure was repeated 10 times using the distorted test data given in Table 7. The corresponding diagnosis results are shown in Table 8.

The results presented in Table 8 confirm that the CMAC correctly diagnoses the input fault even when the input data contains 10~40% noise. In other words, the robustness of the proposed fault diagnosis system toward noise in the input data is confirmed.

7. Conclusions

Heliostat fields contain hundreds if not thousands of individual heliostats, each with their own independent controller. As a result, fault diagnosis via a remote monitoring system is highly challenging. Existing RBF and BP neural network approaches for fault diagnosis in heliostat fields achieve an accurate classification performance, but require a long training time. Accordingly, this study has presented a new approach for fault diagnosis in large-scale heliostat fields by means of a CMAC neural network. The experimental results have shown that the proposed system has a short learning time, a high classification performance and a good robustness toward noise in the input signal.

Acknowledgment

The authors would like to thank the National Science Council, Taiwan, for financially supporting this research under Contracts nos. NSC 100-2221-E-027-015 and NSC 100-2628-E-167-002-MY3.

References

[1] S. Schell, "Design and evaluation of esolar's heliostat fields," *Solar Energy*, vol. 85, no. 4, pp. 614–619, 2011.

[2] K.-K. Chong and M. H. Tan, "Comparison study of two different sun-tracking methods in optical efficiency of heliostat field," *International Journal of Photoenergy*, vol. 2012, Article ID 908364, 10 pages, 2012.

[3] D. Fontani, P. Sansoni, F. Francini, D. Jafrancesco, L. Mercatelli, and E. Sani, "Pointing sensors and sun tracking techniques," *International Journal of Photoenergy*, vol. 2011, Article ID 806518, 9 pages, 2011.

[4] X. Wei, Z. Lu, W. Yu, and Z. Wang, "A new code for the design and analysis of the heliostat field layout for power tower system," *Solar Energy*, vol. 84, no. 4, pp. 685–690, 2010.

[5] X. Wei, Z. Lu, Z. Wang, W. Yu, H. Zhang, and Z. Yao, "A new method for the design of the heliostat field layout for solar tower power plant," *Renewable Energy*, vol. 35, no. 9, pp. 1970–1975, 2010.

[6] K. K. Chong and M. H. Tan, "Range of motion study for two different sun-tracking methods in the application of heliostat field," *Solar Energy*, vol. 85, no. 9, pp. 1837–1850, 2011.

[7] M. Sánchez and M. Romero, "Methodology for generation of heliostat field layout in central receiver systems based on yearly normalized energy surfaces," *Solar Energy*, vol. 80, no. 7, pp. 861–874, 2006.

[8] W. Cheng-Yu, W. Ding-Sheng, and G. Tie-Zheng, "Application of RBF neural network to fault diagnosis in heliostats filed," Information Technology 2011-01.

[9] Y. Yang and W. Tang, "Study of remote bearing fault diagnosis based on BP neural network combination," in *Proceedings of the 7th International Conference on Natural Computation*, vol. 2, pp. 618–621, 2011.

[10] Z. Yang, W. I. Hoi, and J. Zhong, "Gearbox fault diagnosis based on artificial neural network and genetic algorithms," in *Proceedings of the International Conference onSystem Science and Engineering*, pp. 37–42, 2011.

[11] C. P. Hung, M. H. Wang, C. H. Cheng, and W. L. Lin, "Fault diagnosis of steam turbine-generator using CMAC neural network approach," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 2988–2993, July 2003.

[12] H. Shiraishi, S. L. Ipri, and D. I. D. Cho, "CMAC neural network controller for fuel-injection systems," *IEEE Transactions on Control Systems Technology*, vol. 3, no. 1, pp. 32–38, 1995.

[13] C. P. Hung and M. H. Wang, "Diagnosis of incipient faults in power transformers using CMAC neural network approach," *Electric Power Systems Research*, vol. 71, no. 3, pp. 235–244, 2004.

[14] W. S. Lin, C. P. Hung, and M. H. Wang, "CMAC_based fault diagnosis of power transformers," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)*, vol. 1, pp. 986–991, May 2002.

[15] J. S. Albus, "A new approach to manipulator control: the cerebellar model articulation controller," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 97, no. 3, pp. 220–227, 1975.

[16] D. A. Handelman, S. H. Lane, and J. J. Gelfand, "Integrating neural networks and knowledge-based systems for intelligent robotic control," *IEEE Control Systems Magazine*, vol. 10, no. 3, pp. 77–87, 1990.

[17] Y. F. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 115–121, 1992.