This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access

# Adaptive Noise Cancellation Using Deep Cerebellar Model Articulation Controller

Yu Tsao, *Member*, *IEEE*, Hao-Chun Chu, Shih-Hau Fang, *Senior Member*, *IEEE*, Junghsi Lee*, and Chih-Min Lin, *Fellow*, *IEEE*

**Abstract—This paper proposes a deep cerebellar model articulation controller (DCMAC) for adaptive noise cancellation (ANC). We expand upon the conventional CMAC by stacking single-layer CMAC models into multiple layers to form a DCMAC model, and derive a backpropagation training algorithm to learn the DCMAC parameters. Compared with conventional CMAC, the DCMAC can characterize nonlinear transformations more effectively because of its deep structure. Experimental results confirm that the proposed DCMAC model outperforms the CMAC in terms of residual noise in an ANC task, showing that DCMAC provides enhanced capability to model channel characteristics.**

***Index Terms*—*cerebellar model articulation controller, deep learning, adaptive noise cancellation***

## I. INTRODUCTION

The goal of an adaptive noise cancellation (ANC) system is to remove the noise component from signals. In ANC systems, linear filters are widely used for their simple structure and satisfactory performance under general conditions, where least mean square (LMS) [1] and normalized LMS [2] are two effective criteria to estimate the filter parameters. However, when the unknown system has a nonlinear and complex response, a linear filter may not provide optimal performance. Accordingly, some nonlinear adaptive filters have been developed. Successful examples include the unscented Kalman filter [3, 4] and the Volterra filter [5, 6]. Meanwhile, the cerebellar model articulation controller (CMAC), which belongs to the feed forward neural networks, has been used as a complex piecewise linear filter [7, 8]. Experimental results confirmed that CMAC can provide satisfactory performance in terms of mean squared error (MSE) for nonlinear systems [9, 10].

A CMAC model is a partially connected perceptron-like associative memory network [11]. Owing to its peculiar structure, it overcomes fast growing problems and learning difficulties when the amount of training data is limited as compared to other neural networks [8, 12, 13]. Moreover, because of its simple computation and good generalization capability, the CMAC model has been widely used to control complex dynamical systems [14], nonlinear systems [9, 10], robot manipulators [15], and multi-input multi-output (MIMO) control [16, 147].

More recently, deep learning has become a part of many state-of-the-art systems, particularly computer vision [18-20] and speech recognition [21-23]. Numerous studies indicate that by stacking several shallow structures into a single deep structure, the overall system could achieve better data representation and, thus, more effectively deal with nonlinear and high complexity tasks. Successful examples include stacking denoising autoencoders [24], stacking sparse coding [25], multilayer nonnegative matrix factorization [26], and deep neural networks [27, 28]. In this study, we propose a deep CMAC (DCMAC) framework, which stacks several layers of single-layered CMACs. In addition, we derive a backpropagation algorithm to train the DCMAC effectively and efficiently. Experimental results on ANC tasks show that the DCMAC provides better results than conventional CMAC in terms of MSE scores.

The rest of this paper is organized as follows: Section 2 introduces the structure of the CMAC and the learning algorithm to compute the parameters, and presents the structure of the DCMAC and the backpropagation algorithm. Section 3 shows the experimental setup and results. Finally, the conclusion and future work are presented in section 4.

## II. PROPOSED ALGORITHM

### *2.1 System Overview*

Fig. 1 shows the block diagram of a typical ANC system containing two microphones, one external and the other internal. The external microphone receives the noise source signal $n(k)$, while the internal microphone receives the noisy signal $v(k)$. The noisy signal is a mixture of the signal of interest $s(k)$ and the damage noise signal $g(k)$. Therefore, $v(k) = s(k) + g(k)$, where $g(k)$ is generated by passing the noise signal $n(k)$ through an unknown channel $F(\cdot)$. The transformation between $n(k)$ and $g(k)$ is usually nonlinear in real-world conditions [29]. The ANC system aims to compute a filter, $\hat{F}(\cdot)$, which transforms $n(k)$ to $h(k)$, so that the final output, $(v(k) - h(k))$, is close to the signal of interest, $s(k)$. The filter $\hat{F}(\cdot)$ is modeled by a parametric function, whose parameters are usually estimated by minimizing the MSE.

Yu Tsao is with the Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan (e-mail: yu.tsao@citi.sinica.edu.tw).
Hao-Chun Chu is with the Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan.
Shih-Hau Fang is with the Department of Electrical Engineering, Yuan Ze University (YZU), Taoyuan, and MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan. (e-mail: shfang@saturn.yzu.edu.tw).
Junghsi Lee is with the Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan (e-mail: eejlee@saturn.yzu.edu.tw).
Chih-Min Lin is with the Department of Electrical Engineering, Yuan Ze University, Taoyuan, Taiwan (e-mail: cml@saturn.yzu.edu.tw).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access

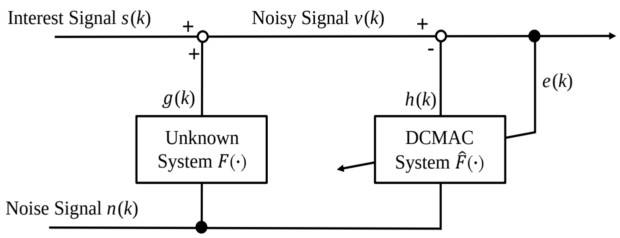



Fig. 1. Block diagram of an adaptive noise cancellation system.

Recently, the concept of deep learning has garnered great attention. Inspired by deep learning, we propose a DCMAC framework, which stacks several layers of the single-layered CMAC, to construct the filter $\hat{F}(\cdot)$, as indicated in Fig. 1. Fig. 2 shows the architecture of the DCMAC, which is composed of a plurality of CMAC layers. The A, R, and W in Fig. 2 denote the association memory space, receptive field space, and weight memory space, respectively, in a CMAC model. In the next section we will detail these three spaces. In Fig. 2, the DCAMC is formed by $L$ CAMCs. The input signal to the DCAMC is $\boldsymbol{x}$, and the output signal is $\boldsymbol{y}^L$. The output of the first layer CMAC ($\boldsymbol{y}^1$) is treated as the input for the next CMAC layer. By using such multi-layer processing, the DCMAC can better characterize the nonlinear transformations, and thus achieve an improved noise cancellation performance.

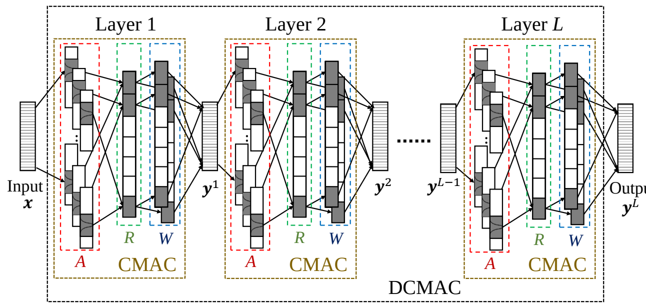



Fig. 2. Architecture of the deep CMAC (DCMAC).

*2.2 Structure of a CMAC Model*

This section reviews the structure and parameter-learning algorithm of the CMAC.

*A. Structure of a CMAC*

Fig. 3 shows a CMAC model with five spaces: an input space, an association memory space, a receptive field space, a weight memory space, and an output space. The main functions of these five spaces are as follows:

1) Input space: This space is the input of the CMAC. In Fig. 3, the input vector is $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^T \in R^N$, where $N$ is the feature dimension.

2) Association memory space: This space holds the excitation functions of the CMAC, and it has a multi-layer concept. Please note that the layers here (indicating the depth of association memory space) are different from those presented in Section 2.1 (indicating the number of CMACs in a DCMAC). To avoid confusion, we call the layer for the association memory "AS_layer" and the layer for the CMAC number "layer" in the following discussion. Fig. 4 shows an example of an association memory space with a two-dimensional input vector, $\boldsymbol{x} = [x_1, x_2]^T$ with $N = 2$. The LB and UB are lower bound and upper bound, respectively. We first divide $x_1$ into blocks (A, B) and $x_2$ into blocks (a, b). Next, by shifting each variable an element, we obtain blocks (C, D) and blocks (c, d) for the second AS_layer. Likewise, by shifting another variable, we can generate another AS_layer. In Fig. 4, we have four AS_layers, with each AS_layer having two blocks, and therefore, the block number is eight ($N_B = 8$) for one variable; accordingly, the overall association memory space has 16 blocks ($N_A = N_B \times N$). Each block contains an excitation function, which must be a continuously bounded function, such as the Gaussian, triangular, or wavelet function. In this study, we use the Gaussian function [as shown in Fig. 4]:

$$\varphi_{ij} = \exp\left[\frac{-(x_i - m_{ij})^2}{\sigma_{ij}^2}\right] \quad \text{for} \quad j = 1,2,\cdots,N_B \quad \text{and} \quad i = 1,2,\cdots,N, \tag{1}$$

where $m_{ij}$ and $\sigma_{ij}$ represent the associative memory functions within the mean and variance, respectively, of the $i$-th input of the $j$-th block.

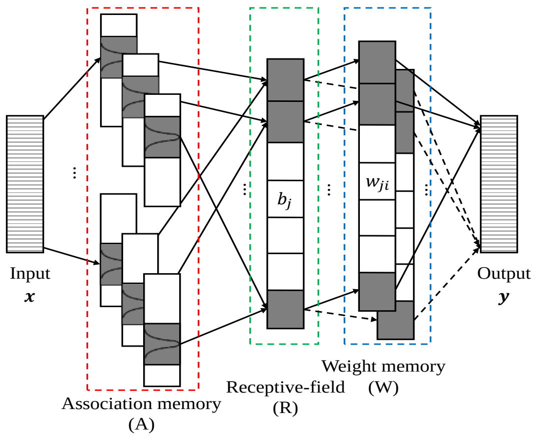



Fig. 3. Architecture of a CMAC.

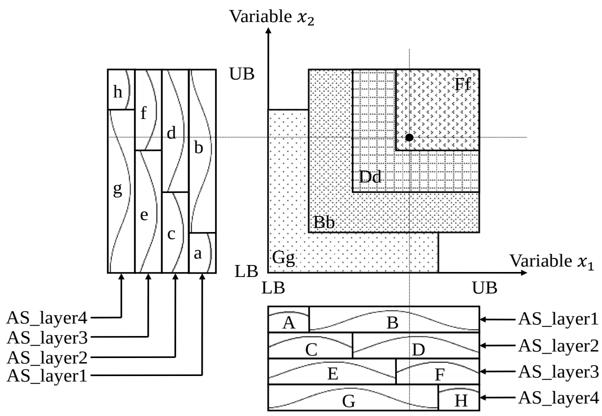



Fig. 4. Architecture of the CMAC with a two-dimensional vector ($N = 2$)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access

3) Receptive field space: In Fig. 4, areas formed by blocks are called receptive fields. The receptive field space has eight areas ($N_R$=8): Aa, Bb, Cc, Dd, Ee, Ff, Gg, and Hh. Given the input $\boldsymbol{x}$, the $j$-th receptive field function is represented as [9, 10]:

$$b_j = \prod_{i=1}^{N} \varphi_{ij} = \exp\left[-\left(\sum_{i=1}^{N} \frac{(x_i-m_{ij})^2}{\sigma_{ij}^2}\right)\right]. \qquad (2)$$

In the following, we express the receptive field functions in the form of vectors, namely, $\boldsymbol{b} = \left[b_1, b_2, \cdots, b_{N_R}\right]^T$. In this study, we set $N_R = N_B$.

4) Weight memory space: This space specifies the adjustable weights of the results of the receptive field space:

$$\boldsymbol{w}_p = \left[w_{1p}, w_{2p}, \cdots, w_{N_R p}\right]^T \quad \text{for} \quad p = 1, 2, \cdots, M, \qquad (3)$$

where $M$ denotes the output vector dimension.

5) Output space: From Fig. 3, the output of the CMAC is [9, 10]:

$$y_p = \boldsymbol{w}_p{}^T \boldsymbol{b} = \sum_{j=1}^{N_R} w_{jp} \exp\left[-\left(\sum_{i=1}^{N} \frac{(x_i-m_{ij})^2}{\sigma_{ij}^2}\right)\right], \qquad (4)$$

where $y_p$ is the $p$-th element of the output vector, $\boldsymbol{y} = [y_1, y_2, \cdots, y_M]^T$. The output of state point is the algebraic sum of outputs of receptive fields (Aa, Bb, Cc, Dd, Ee, Ff, Gg, and Hh) multiplied by the corresponding weights.

*B. Parameters of Adaptive Learning Algorithm*

To estimate the parameters in the association memory, receptive field, and weight memory spaces of the CMAC, we first define an objective function:

$$O(k) = \frac{1}{2}\sum_{t=1}^{M}[e_t(k)]^2 = \frac{1}{2}\sum_{t=1}^{M}[y_t(k) - d_t(k)]^2, \qquad (5)$$

where the error signal $e_t(k)$ indicates the error between the desired response $d_t(k)$ and the filter's output $y_t(k)$, at the $k$-th sample. Based on Eq. (5), the normalized gradient descent method can be used to derive the update rules for the parameters in a CMAC model:

$$m_{ij}(k+1) = m_{ij}(k) + \mu_m \frac{\partial O}{\partial m_{ij}} \quad , \quad \text{where } \frac{\partial O}{\partial m_{ij}} = b_j \frac{2(x_i-m_{ij})}{(\sigma_{ij})^2}\left(\sum_{t=1}^{M} e_t(k) w_{jt}\right); \qquad (6)$$

$$\sigma_{ij}(k+1) = \sigma_{ij}(k) + \mu_\sigma \frac{\partial O}{\partial \sigma_{ij}} \quad , \quad \text{where } \frac{\partial O}{\partial \sigma_{ij}} = b_j \frac{2(x_i-m_{ij})^2}{(\sigma_{ij})^3}\left(\sum_{t=1}^{M} e_t(k) w_{jt}\right); \qquad (7)$$

$$w_{jp}(k+1) = w_{jp}(k) + \mu_w \frac{\partial O}{\partial w_{jp}} \quad , \quad \text{where } \frac{\partial O}{\partial w_{jp}} = e_p(k) b_j \quad , \qquad (8)$$

where $\mu_m$ and $\mu_\sigma$ are the learning rates for updating the mean and variance in the associative memory functions, and , and $\mu_w$ is the learning rate for the adjustable weights.

## *2.3 Proposed DCMAC Model*

This section describes the structure of a DCMAC and the corresponding learning algorithm.

*A. Structure of the DCMAC*

From Eq. (4), the output of the first layer $\boldsymbol{y}^1$ is obtained by

$$y_p^1 = \sum_{j=1}^{N_R^1} w_{jp}^1 \exp\left[-\left(\sum_{i=1}^{N} \frac{\left(x_i-m_{ij}^1\right)^2}{\sigma_{ij}^{1\,2}}\right)\right], \qquad (9)$$

where $y_p^1$ is the $p$-th element of the output of $\boldsymbol{y}^1$, and $N_R^1$ is the number of receptive fields in the first layer. Next, the correlation of the output of the ($l$-1)-th layer ($\boldsymbol{y}^{l-1}$) and that of the $l$-th layer ($\boldsymbol{y}^l$) can be formulated as

$$y_p^l = \sum_{j=1}^{N_R^l} w_{jp}^l \exp\left[-\left(\sum_{i=1}^{N^l} \frac{\left(y_i^{l-1}-m_{ij}^l\right)^2}{\sigma_{ij}^{l\,2}}\right)\right], \; l = 2 \sim L, \qquad (10)$$

where $N^l$ is the input dimension of the $l$-th layer (output dimension of the $(l-1)$-th layer); $N_R^l$ is the number of receptive fields in the $l$-th layer; $y_p^l$ is the $p$-th element of the output of $\boldsymbol{y}^l$; $m_{ij}^l$, $\sigma_{ij}^l$, and $w_{jp}^l$ are the parameters in the $l$-th CMAC; $L$ is the total number of CMAC in a DCMAC.

*1) Backpropagation Algorithm for DCMAC*

Assume that the output vector of a DCMAC is $\boldsymbol{y}^L = [y_1^L, y_2^L, \cdots, y_{M^L}^L]^T \in R^{M^L}$, where $M^L$ is the feature dimension, the objective function of the DCMAC is

$$O(k) = \frac{1}{2}\sum_{t=1}^{M^L}[y_t^L(k) - d_t(k)]^2. \qquad (11)$$

In the following, we present the backpropagation algorithm to estimate the parameters in the DCMAC. Because the update rules for "means and variances" and "weights" are different, they are presented separately.

1) The update algorithm of means and variances:

The update algorithms of the means and variances for the last layer (the $L$-th layer) of the DCMAC are the same as that of CMAC (as shown in Eqs. (6) and (7)). For the penultimate layer (the ($L$-1)-th layer), the parameter updates are based on:

$$\frac{\partial O}{\partial z_{ip}^{L-1}} = \frac{\partial b_p^{L-1}}{\partial z_{ip}^{L-1}} \frac{\partial O}{\partial b_p^{L-1}} \quad , \qquad (12)$$

where $b_p^{L-1}$ is the $p$-th receptive field function for the ($L$-1)-th layer. We define the momentum $\delta_{z_p}^{L-1} = \frac{\partial O}{\partial b_p^{L-1}}$ of the $p$-th receptive field function in the ($L$-1)-th layer. Then, we have

$$\delta_{z_p}^{L-1} = \sum_{j=1}^{N_R^L} \frac{\partial b_j^L}{\partial b_p^{L-1}} \frac{\partial O}{\partial b_j^L} \qquad (13)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access

$$= \sum_{t=1}^{M^{L-1}} \frac{\partial y_t^{L-1}}{\partial b_p^{L-1}} \sum_{j=1}^{N_R^L} \frac{\partial b_j^L}{\partial y_t^{L-1}} \delta_{z_j}^L ,$$

where $b_j^L$ is the $j$-th receptive field function for the $L$-th layer, $y_t^{L-1}$ is the $t$-th element of the $\boldsymbol{y}^{L-1}$, $N_R^L$ is the number of receptive fields in the $L$-th layer, and $M^{L-1}$ is the feature dimension of $\boldsymbol{y}^{L-1}$. Notably, by replacing $z$ with $m$ and $\sigma$ in Eq. (13), we obtain momentums $\delta_{m_p}^{L-1}$ and $\delta_{\sigma_p}^{L-1}$.

Similarly, we can derive the momentum, $\delta_{z_p}^{L-2}$, for the $p$-th receptive field function in the ($L$-2)-th layer by:

$$\begin{aligned} \delta_{z_p}^{L-2} &= \frac{\partial O}{\partial b_p^{L-2}} \\ &= \sum_{j=1}^{N_R^{L-1}} \frac{\partial b_j^{L-1}}{\partial b_p^{L-2}} \frac{\partial O}{\partial b_j^{L-1}} \\ &= \sum_{t=1}^{M^{L-2}} \frac{\partial y_t^{L-2}}{\partial b_p^{L-2}} \sum_{j=1}^{N_R^{L-1}} \frac{\partial b_j^{L-1}}{\partial y_t^{L-2}} \delta_{z_j}^{L-1}, \end{aligned} \tag{14}$$

where $b_j^{L-1}$ is the $j$-th receptive field function for the ($L$-1)-th layer, $y_t^{L-2}$ is the $t$-th element of the $\boldsymbol{y}^{L-2}$, $N_R^{L-1}$ is the number of receptive fields in the $(L-1)$-th layer, and $M^{L-2}$ is the feature dimension of $\boldsymbol{y}^{L-2}$.

Based on the normalized gradient descent method, the learning algorithm of $m_{ip}^l$ (the $i$-th mean parameter in the $p$-th receptive field in the $l$-th layer) is defined as

$$m_{ip}^l(k+1) = m_{ip}^l(k) + \mu_{m^l} \frac{\partial b_p^l}{\partial m_{ip}^l} \delta_{m_p}^l; \tag{15}$$

similarly, the learning algorithm of $\sigma_{ij}^l$ (the $i$-th variance parameter in the $j$-th receptive field in the $l$-th layer) is defined as

$$\sigma_{ip}^l(k+1) = \sigma_{ip}^l(k) + \mu_{\sigma^l} \frac{\partial b_p^l}{\partial \sigma_{ip}^l} \delta_{\sigma_p}^l , \tag{16}$$

where $\mu_{m^l}$ in Eq. (15) and $\mu_{\sigma^l}$ in Eq. (16) are the learning rates for the mean and variance updates, respectively.

2) The update algorithm of weights:

The update rule of the weight in the last layer (the $L$-th layer) of the DCMAC is the same as that for the CMAC (as shown in Eq. (8)). For the penultimate layer (the ($L$-1)-th layer), the parameter update is:

$$\frac{\partial O}{\partial w_{jp}^{L-1}} = \frac{\partial y_p^{L-1}}{\partial w_{jp}^{L-1}} \frac{\partial O}{\partial y_p^{L-1}} . \tag{17}$$

where $y_p^{L-1}$ is the $p$-th element of the $\boldsymbol{y}^{L-1}$. Then, we define the momentum for the ($L$-1)-th layer $\delta_{w_p}^{L-1} = \frac{\partial O}{\partial y_p^{L-1}}$:

$$\begin{aligned} \delta_{w_p}^{L-1} &= \sum_{j=1}^{N_R^L} \frac{\partial b_j^L}{\partial y_p^{L-1}} \sum_{t=1}^{M^L} \frac{\partial y_t^L}{\partial b_j^L} \frac{\partial O}{\partial y_t^L} \\ &= \sum_{j=1}^{N_R^L} \frac{\partial b_j^L}{\partial y_p^{L-1}} \sum_{t=1}^{M^L} \frac{\partial y_t^L}{\partial b_j^L} \delta_{w_t}^L , \end{aligned} \tag{18}$$

Similarly, the momentum for the ($L$-2)-th layer can be computed by:

$$\begin{aligned} \delta_{w_p}^{L-2} &= \sum_{j=1}^{N_R^{L-1}} \frac{\partial b_j^{L-1}}{\partial y_p^{L-2}} \sum_{t=1}^{M^{L-1}} \frac{\partial y_t^{L-1}}{\partial b_j^{L-1}} \frac{\partial O}{\partial y_t^{L-1}} \\ &= \sum_{j=1}^{N_R^{L-1}} \frac{\partial b_j^{L-1}}{\partial y_p^{L-2}} \sum_{t=1}^{M^{L-1}} \frac{\partial y_t^{L-1}}{\partial b_j^{L-1}} \delta_{w_t}^{L-1} . \end{aligned} \tag{19}$$

where $y_p^{L-2}$ is the $p$-th element of the $\boldsymbol{y}^{L-2}$.

According to the normalized gradient descent method, the learning algorithm of $w_{jp}^l$ (weight for the $j$-th receptive field and the $p$-th output in the $l$-th layer) is defined as

$$w_{jp}^l(k+1) = w_{jp}^l(k) + \mu_{w^l} \frac{\partial y_p^l}{\partial w_{jp}^l} \delta_{w_p}^l, \tag{20}$$

where $\mu_{w^l}$ is the learning rate for the weights.

## III. EXPERIMENTS

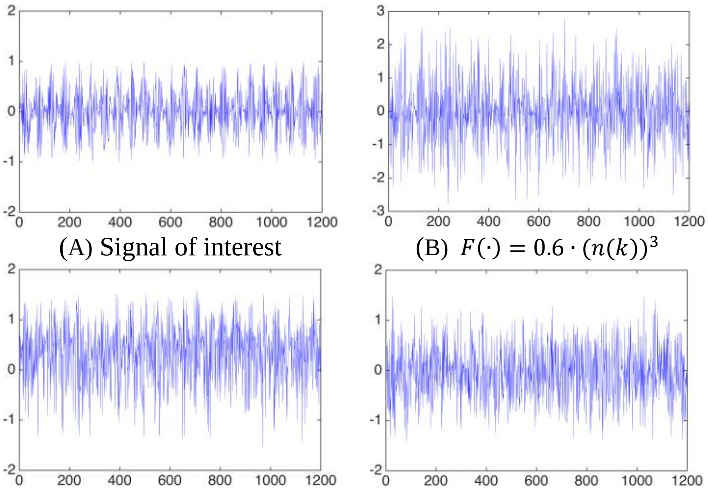
### *3.1 Experimental Setup*

In the experiment, we consider the signal of interest $s(k) = \sin(0.06k)$ multiplied by a white noise signal, normalized within $[-1, 1]$, as shown in Fig. 5 (A). The noise signal, $n(k)$, is generated by white noise, normalized within $[-1.5, 1.5]$. A total of 1200 training samples are used in this experiment. The noise signal $n(k)$ will go through a nonlinear channel generating the damage noise $z(k)$. The relation between $n(k)$ and $z(k)$ is $z(k) = F(n(k))$, where $F(\cdot)$ represents the function of the nonlinear channel. In this experiment, we used 12 different functions, $\{ 0.6 \cdot (n(k))^{2i-1}; 0.6 \cdot \cos((n(k))^{2i-1});\ 0.6 \cdot \sin((n(k))^{2i-1}), i=1, 2, 3, 4 \}$ to generate four different damage noise signals $z(k)$. The noisy signals $v(k)$ associated with four different $z(k)$ signals, with three representative channel functions, namely, $F = 0.6 \cdot (n(k))^3$, $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$, and $F(\cdot) = 0.6 \cdot \sin((n(k))^3)$ are shown in Figs. 5 (B), (C), and (D), respectively.

We followed reference [8] to set up the parameters of the DCMAC, as characterized below:

1) Number of layers ($AS\_layer$): 4
2) Number of blocks ($N_B$)=8: $Ceil(5\ \ (N_e)/ 4\ \ (AS\_layer)) \times 4\ \ (AS\_layer) = 8$.
3) Number of receptive fields ($N_R$)= 8.
4) Associative memory functions: $\varphi_{ij} = exp\left[-\left(x_i - m_{ij}\right)^2/\sigma_{ij}^2\right], i = 1;\ \ j = 1, \cdots, N_R$.

Note that $Ceil(\cdot)$ represents the unconditional carry of the remainder. Signal range detection is required to set the UB and LB necessary to include all the signals. In this study, [UB LB]=[3 -3] gives the best performance. Please note that the main goal this study is to investigate whether DCMAC can yield better ANC results than a single-layer CMAC. Therefore, we report the results using [3 -3] for both CMAC and DCMAC in the following discussions. The initial means of the Gaussian function ($m_{ij}$) are set in the middle of each block ($N_B$) and the initial variances of the Gaussian function ($\sigma_{ij}$) are determined by the size of each block ($N_B$). With [UB LB]=[3 -3], we initialize the mean parameters as: $m_{i1} = -2.4$, $m_{i2} = -1.8$, $m_{i3} = -1.2$, $m_{i4} = -0.6$, $m_{i5} = 0.6$, $m_{i6} = 1.2$, $m_{i7} = 1.8$, $m_{i8} = 2.4$, so that the eight blocks can cover [UB LB] more evenly. Meanwhile, we set $\sigma_{ij} = 0.6$ for $j$=1,..8, and the initial weights ($w_{jt}$) zeros. Based on our experiments, the parameters initialized differently only affect the performance at the first few epochs and converge to similar values quickly. The learning rates are chosen as $\mu_s = \mu_z = \mu_w = \mu_m = \mu_\sigma = 0.001$ (this learning rates achieve better results in our preliminary investigation). The parameters within all layers of the DCMAC are the same. In this study, we examine the performance of DCMACs formed by three, five, and seven layers of CMACs, which are denoted as DCMAC(3), DCMAC(5), and DCMAC(7), respectively. The input dimension was set as $N$=1; the output dimensions for CMAC and DCMACs were set as $M = 1$ and $M^L = 1$, respectively.



(A) Signal of interest (B) $F(\cdot) = 0.6 \cdot (n(k))^3$

(C)$F(\cdot) = 0.6 \cdot \cos((n(k))^3)$ (D)$F(\cdot) = 0.6 \cdot \sin((n(k))^3)$

Fig. 5. (A) Signal of interest $s(k)$. (B-D) Noisy signal $v(k)$ with three channel functions.

### 3.2 Experimental Results

This section compares DCMAC with different architectures based on two performance metrics, the MSE and the convergence speed. Fig. 6 shows the converged MSE under a CMAC and a DCMAC under the three different structures testing on the channel function $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$. The three structures include (AS_layer = 2, $N_e = 5$), (AS_layer = 4, $N_e = 5$), and (AS_layer = 4, $N_e = 9$), the three groups of results being demonstrated from the left to right in Fig. 6. To compare the performance of the proposed DCMAC, we have conducted experiments using two popular adaptive filter methods, namely LMS [1] and the Volterra filter [5, 6]. For a fair comparison, the learning epochs are set the same for LMS, Volterra, CMAC, and DCMAC, where there are 1200 data samples in each epoch. The parameters of LMS and the Volterra filter are tested and the best results are reported in Fig. 6. Please note that the results of LMS and the Volterra filter are the same across the three groups of results.

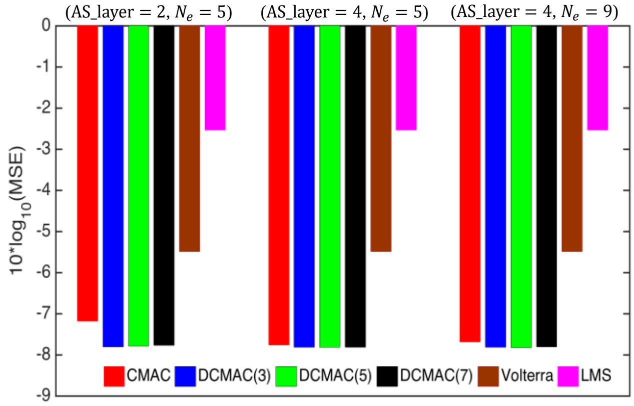


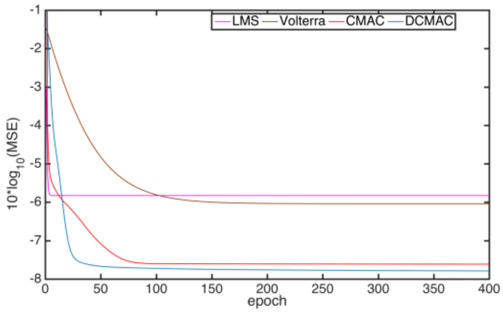

Fig. 6. MSE of LMS, Volterra, CMAC, and DCMAC with channel function $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$.

From Fig. 6, we see that DCMAC outperforms not only conventional Volterra and LMS, but also CMAC under the three setups. The results confirm the advantage of increasing the depth of CMAC to attain better ANC performance. We observed the same trends across 12 different channel functions, and thus only the result of $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$ is presented as a representative.

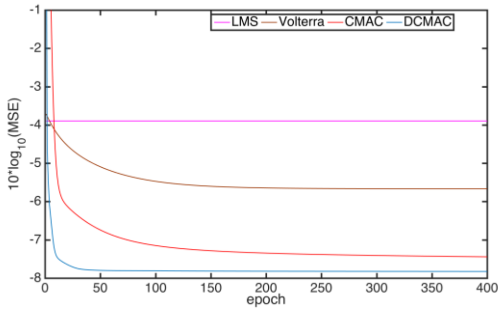
Fig. 7 shows the convergence speed and the MSE reduction rate versus the number of epochs, for different algorithms. Speed is also an important performance metric in an adaptive filter. For ease of comparison, Fig. 7 only shows the results of three-layer DCMAC (denoted as DCMAC in Fig. 7) since the trends of DCMAC performances are consistent across different layer numbers (as can be seen in Fig. 6). For CMAC and DCMAC, we adopted AS_layer = 4, $N_e = 5$. Fig. 7 shows the results of three channel functions, namely, $F(\cdot) = 0.6 \cdot (n(k))^3$, $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$, and $F(\cdot) = 0.6 \cdot \sin((n(k))^3)$. The results in Fig. 7 first show that LMS and Volterra yield better performance than CMAC and DCMAC when the number of epoch is few. On the other hand, when the number of epoch becomes large, both DCMAC and CMAC yield lower MSE scores compared to that from LMS and Volterra, over all the three testing channels. Moreover, DCMAC consistently outperforms CMAC with a lower converged MSE scores. The results also show that the performance gain of the DCMAC becomes increasingly more significant as the nonlinearity of the channels increases. Finally, we note that the performances of both DCMAC and CMAC became saturated around 400 epochs. In a real-world application, a development set of data can be used to determine the saturation point, so that the adaptation can be switched off.

Simulation results of a CMAC and that of a DCMAC, both for 400 epochs of training, are shown in Figs. 8 (A) and (B), respectively. The results show that the proposed DCMAC can achieve better filtering performance than that from the CMAC for this noise cancellation system.

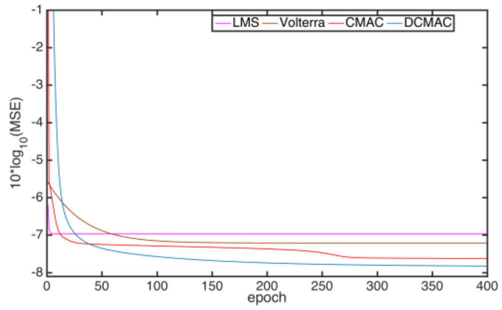
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access




(A) $F(\cdot) = 0.6 \cdot (n(k))^3$

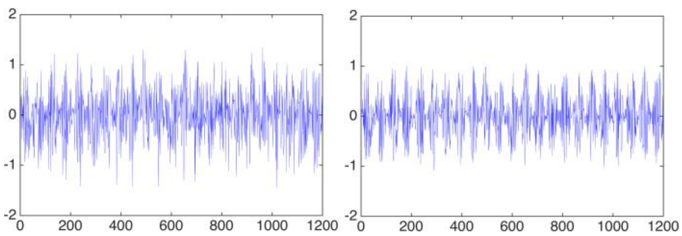



(B) $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$




(C) $F(\cdot) = 0.6 \cdot \sin((n(k))^3)$

Fig. 7. MSE of LMS, Volterra, CMAC, and DCMAC with three types of channel functions. More results are presented in http://wimoc70639.simplesite.com/419530354



(A) CMAC (B) DCMAC

Fig. 8. Recovered signal using (A) CMAC and (B) DCMAC, where $F(\cdot) = 0.6 \cdot \cos((n(k))^3)$.

Table I lists the mean and variance of MSE scores for LMS, Volterra, CMAC, and DCMAC across 12 channel functions. The MSE for each method at a channel function was obtained with 1000 epochs of training. From the results, both CMAC and DCMAC give lower MSE than LMS and Volterra. In addition to the results in Table I, we adopted the dependent t-Test for the hypothesis test on the 12 sets of results. The t-Test results revealed that DCMAC outperforms CMAC with $P$-values = 0.005.

TABLE I.
MEAN AND VAIRAINCE OF 10 $\log_{10}$(MSE) SCORES FOR LMS, VOLTERRA, CMAC, AND DCMAC OVER 12 CHANNEL FUCNTIONS

| | LMS | Volterra | CMAC | DCMAC |
|---|---|---|---|---|
| Mean | −4.35 | −5.05 | −7.01 | −7.59 |
| Variance | 11.95 | 11.57 | 1.08 | 0.19 |

## IV. Conclusion

The contribution of the present study was two-fold: First, inspired by the recent success of deep learning algorithms, we extended the CMAC structure into a deep one, termed deep CMAC (DCMAC). Second, a backpropagation algorithm was derived to estimate the DCMAC parameters. Due to the five-space structure, the backpropagation for DCAMC is different from that used in the related artificial neural networks. The parameter updates involved in DCMAC training include two parts (1) The update algorithm of means and variances; (2) The update algorithm of weights. Experimental results of the ANC tasks showed that the proposed DCMAC can achieve better noise cancellation performance when compared with that from the conventional single-layer CMAC. In future, we will investigate the capabilities of the DCMAC on other signal processing tasks, such as echo cancellation and single-microphone noise reduction. Meanwhile, advanced deep learning algorithms used in deep neural networks, such as dropout and sparsity constraints, will be incorporated in the DCMAC framework. We will also compare the proposed deep models with other types of deep models in the ANC task. Finally, similar to related deep learning researches, identifying a way to optimally specify the number of layers and suitably initialize parameters in DCMAC per the amount of training data are important future works.

## Acknowledgement

The authors would like to thank the financial support provided by Ministry of Science and Technology, Taiwan (MOST 106-2221-E-001-017-MY2).

## References

[1] B. Widrow, et al., "Adaptive noise cancelling: Principles and applications," *Proceedings of the IEEE*, vol. 63 (12), pp. 1692-1716, 1975.

[2] S. Haykin, *Adaptive Filter Theory*, *fourth edition*, Prentice-Hall, 2002.

[3] E. A. Wan and R. van der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proc. AS-SPCC*, pp. 153-158, 2000.

[4] F. Daum, "Nonlinear filters: beyond the Kalman filter," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20 (8), pp. 57-69, 2005.

[5] L. Tan and J. Jiang, "Adaptive Volterra filters for active control of nonlinear noise processes," *IEEE Transactions on Signal Processing*, vol. 49 (8), pp. 1667-1676, 2001.

[6] V. John Mathews, "Adaptive Volterra filters using orthogonal structures," *IEEE Signal Processing Letters*, vol. 3 (12), pp. 307-309 1996.
[7] G. Horvath and T. Szabo, "CMAC neural network with improved generalization property for system modeling," in *Proc. IMTC*, vol. 2, pp. 1603-1608, 2002.
[8] C. M. Lin, L. Y. Chen, and D. S. Yeung, "Adaptive filter design using recurrent cerebellar model articulation controller," *IEEE Trans. on Neural Networks*, vol. 21 (7), pp. 1149-1157, 2010.
[9] C. M. Lin and Y. F. Peng, "Adaptive CMAC-based supervisory control for uncertain nonlinear systems," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34 (2), pp. 1248-1260, 2004.
[10] C. P. Hung, "Integral variable structure control of nonlinear system using a CMAC neural network learning approach," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34 (1), pp. 702-709, 2004.
[11] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control*, vol. 97 (3), pp. 220–227, 1975.
[12] P. E. M. Almeida and M. G. Simoes, "Parametric CMAC networks: Fundamentals and applications of a fast convergence neural structure," *IEEE Trans. Ind. Applicat.*, vol. 39 (5), pp. 1551–1557, 2003.
[13] C. M. Lin, L. Y. Chen, and C. H. Chen, "RCMAC hybrid control for MIMO uncertain nonlinear systems using sliding-mode technology," *IEEE Trans. Neural Netw.*, vol. 18 (3), pp. 708–720, 2007.
[14] S. Commuri and F. L. Lewis, "CMAC neural networks for control of nonlinear dynamical systems: Structure, stability and passivity," *Automatica*, vol. 33 (4), pp. 635-641, 1997.
[15] Y. H. Kim and F. L. Lewis, "Optimal design of CMAC neural-network controller for robot manipulators," *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30 (1), pp. 22-31, 2000.
[16] J. Y. Wu, "MIMO CMAC neural network classifier for solving classification problems," *Applied Soft Computing*, vol. 11 (2), pp. 2326-2333, 2011.
[17] Z. R. Yu, T. C. Yang, and J. G. Juang, "Application of CMAC and FPGA to a twin rotor MIMO system," in *Proceedings ICIEA*, pp. 264-269, 2010.
[18] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35 (8), pp.1915-1929, 2013.
[19] H. Lee, C. Ekanadham, and A. Y. Ng, "Sparse deep belief net model for visual area V2," *in Proc. NIPS*, 2007.
[20] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371-3408, 2010.
[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29 (6), pp. 82-97, 2012.
[22] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436-444, 2015.
[23] S. M. Siniscalchi, T. Svendsen, and C. H. Lee, "An artificial neural network approach to automatic speech processing," *Neurocomputing*, vol. 140, pp. 326-338, 2014.
[24] P. Vincent, H., Larochelle, I., Lajoie, Y., Bengio, and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, pp. 3371-3408, 2010.
[25] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, "Unsupervised feature learning by deep sparse coding," in *SDM*, pp. 902-910, 2014.
[26] A. Cichocki and R. Zdunek, "Multilayer nonnegative matrix factorization," *Electronics Letters*, vol. 42 (16), pp. 947-948, 2006.
[27] S. Liang and R. Srikant, "Why deep neural networks?" *arXiv preprint arXiv:1610.04161*.
[28] J. Ba and R. Caruana, "Do deep nets really need to be deep?" In *Procs NIPS*, pp. 2654-2662, 2014.
[29] C. T. Lin and C. F. Juang, "An adaptive neural fuzzy filter and its applications," *IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 27 (4), pp. 635-656, 1997.

**Yu Tsao** (M'09) received the B.S. and M.S. degrees in electrical engineering from National Taiwan University in 1999 and 2001, respectively, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology in 2008. From 2009 to 2011, He was a Researcher with the National Institute of Information and Communications Technology, Japan, where he was involved in research and product development in automatic speech recognition for multilingual speech-to-speech translation. He is currently an Associate Research Fellow with the Research Center for Information Technology Innovation, Academia Sinica, Taipei, Taiwan. His research interests include speech and speaker recognition, acoustic and language modeling, audio-coding, and bio-signal processing. He received the Academia Sinica Career Development Award in 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2827699, IEEE Access

**Hao-Chun Chu** received the B.S. degree in control engineering from the Southern Taiwan University of Science and Technology, Tainan, Taiwan, in 2014, and the M.S. degree in control engineering from Yuan Ze University, Taoyuan, Taiwan, in 2016. His research interests include cerebellar model articulation controller, deep learning, controller neural network, signal processing. He is currently a Software Engineer with the Advanced Driver Assistance Systems Team, oToBrite Electronics, Inc.

**Shih-Hau Fang** is a Full Professor in the Department of Electrical Engineering, Yuan Ze University (YZU), and MOST Joint Research Center for AI Technology and All Vista Healthcare, Taiwan. He received a B.S. from National Chiao Tung University in 1999, an M.S. and a Ph.D. from National Taiwan University, Taiwan, in 2001 and 2009, respectively, all in communication engineering. From 2001 to 2007, he was a software architect at Internet Services Division at Chung-Hwa Telecom Company Ltd. and joined YZU in 2009. Prof. Fang received the YZU Young Scholar Research Award in 2012 and the Project for Excellent Junior Research Investigators, Ministry of Science and Technology in 2013. His team won the third place of IEEE BigMM HTC Challenge in 2016, and the third place of IPIN in 2017. He is currently technical advisor to HyXen Technology Company Ltd. and serves as an Associate Editor for IEICE Trans. on Information and Systems. Prof. Fang's research interests include indoor positioning, mobile computing, machine learning, data science and signal processing. He is a senior member of IEEE.

**Junghsi Lee** received the B.S. degree in control engineering from the National Chiao-Tung University, Taiwan, in 1982, and the M.S. and Ph.D. degrees in electrical engineering from the University of Utah in 1987 and 1992, respectively. From 1992 to 1996 he was a researcher with the Industrial Technology Research Institute, Hsinchu, Taiwan. Since 1997 he has been with the Department of Electrical Engineering, Yuan-Ze University, Chung-Li, Taiwan, where he is now an Associate Professor. His research interests include digital signal processing, nonlinear adaptive filtering algorithms and their applications.

**Chih-Min Lin** (Fellow10) was born in Taiwan, in 1959. He received the B.S. and M.S. degrees from Department of Control Engineering and the Ph.D. degree from Institute of Electronics Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 1981, 1983 and 1986, respectively. He is currently a Chair Professor and the Vice President of Yuan Ze University, Chung-Li, Taiwan. His current research interests include fuzzy neural network, cerebellar model articulation controller, intelligent control systems and signal processing. He has published more than 180 journal papers. Dr. Lin was the recipient of an Honor Research Fellow at the University of Auckland, Auckland, New Zealand, from 1997 to 1998. He also serves as an Associate Editor of *IEEE Transactions on Cybernetics* and *IEEE Transactions on Fuzzy Systems*. He is an IEEE Fellow and IET Fellow.