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, $(\hat{v}(k) - h(k))$, 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.
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 \( \hat{P}() \), 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 \( x \), and the output signal is \( y^L \). The output of the first layer CMAC \( (y_1^1) \) 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.

**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. 2. Architecture of the deep CMAC (DCMAC).](image)

Fig. 2 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 \( x = [x_1, x_2, \ldots, x_N] \) \( \in \mathbb{R}^N \), 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, \( x = [x_1, x_2] \) \( \in \mathbb{R}^2 \). 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]:

\[
\phi_{ij} = \exp \left[ \frac{-(x_i - m_{ij})^2}{\sigma_{ij}^2} \right] \quad \text{for} \quad j = 1, 2, \ldots, N_b \quad \text{and} \quad i = 1, 2, \ldots, N,
\]

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.

### B. Parameter Learning Algorithm

This section will present the parameter-learning algorithm of CMAC.

1. **Initialization**: Initialize the value of the weight matrix \( \mathbf{W} \) for the CMAC and the excitation function \( \phi_{ij} \) for the AS_layer.

2. **Forward Propagation**: For a given input \( x \), compute the output \( y \) of the CMAC layer by layer.

3. **Backpropagation**: Compute the error \( e \) between the desired output \( d \) and the actual output \( y \), and then update the weight matrix \( \mathbf{W} \) and excitation function \( \phi_{ij} \) using the appropriate learning rule.

4. **Stopping Criteria**: If the error \( e \) is below a certain threshold, the learning process is stopped; otherwise, go to step 2.

This algorithm allows the CMAC to learn the mapping from input to output, and the AS_layer to adaptively estimate the excitation function.

**References**


**Acknowledgments**

This work was supported by the National Natural Science Foundation of China under Grant 61703047 and by the Chinese Academy of Sciences under Grant QYZDJ-SSW-SLH008.
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 \(x\), the \(j\)-th receptive field function is represented as \([9,10]\
\[b_j = \prod_{i=1}^{N_R} \varphi_{ij} = \exp \left[- \left( \sum_{i=1}^{N_R} \frac{(x_i-m_{ij})^2}{\sigma_{ij}^2} \right) \right]. \tag{2}\]

In the following, we express the receptive field functions in the form of vectors, namely, \(b = [b_1, b_2, \cdots, b_{N_R}]^T\). 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:
\[w_p = [w_{1p}, w_{2p}, \cdots, w_{N_Bp}]^T \text{ for } p = 1, 2, \cdots, M, \tag{3}\]
where \(M\) denotes the output vector dimension.

5) Output space: From Fig. 3, the output of the CMAC is \([9,10]\):
\[y_p = w_p^T b = \sum_{j=1}^{N_B} w_{jp} \exp \left[- \left( \sum_{i=1}^{N_R} \frac{(x_i-m_{ij})^2}{\sigma_{ij}^2} \right) \right], \tag{4}\]
where \(y_p\) is the \(p\)-th element of the output vector, \(y = [y_1, y_2, \cdots, y_M]^T\). The output of state point is the algebraic sum of outputs of receptive fields (Aa, Bb, Cc, Dd, Ef, 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:
\[O(k) = \frac{1}{2} \sum_{l=1}^{M} [e_l(k)]^2 = \frac{1}{2} \sum_{l=1}^{M} [y_l(k) - d_l(k)]^2, \tag{5}\]
where the error signal \(e_l(k)\) indicates the error between the desired response \(d_l(k)\) and the filter’s output \(y_l(k)\), 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:
\[m_{ij}(k+1) = m_{ij}(k) + \mu_m \frac{\partial O}{\partial m_{ij}}, \tag{6}\]
where \(\frac{\partial O}{\partial m_{ij}} = b_j \frac{2(x_i-m_{ij})}{(\sigma_{ij})^2} \left( \sum_{l=1}^{M} e_l(k)w_{jl} \right)\);
\[\sigma_{ij}(k+1) = \sigma_{ij}(k) + \mu_\sigma \frac{\partial O}{\partial \sigma_{ij}}, \tag{7}\]
where \(\frac{\partial O}{\partial \sigma_{ij}} = b_j \frac{2(x_i-m_{ij})^2}{(\sigma_{ij})^3} \left( \sum_{l=1}^{M} e_l(k)w_{jl} \right)\);
\[w_{jp}(k+1) = w_{jp}(k) + \mu_w \frac{\partial O}{\partial w_{jp}}, \tag{8}\]
where \(\frac{\partial O}{\partial w_{jp}} = e_p(k)b_j\),
where \(\mu_m\) and \(\mu_\sigma\) are the learning rates for updating the mean and variance in the associative memory functions, 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 \(y^1\) is obtained by
\[y^1_p = \sum_{j=1}^{N_B} w_{jp} \exp \left[- \left( \sum_{i=1}^{N_R} \frac{(x_i-m_{ij})^2}{\sigma_{ij}^2} \right) \right], \tag{9}\]
where \(y^1_p\) is the \(p\)-th element of the output of \(y^1\), and \(N_B^1\) is the number of receptive fields in the first layer. Next, the correlation of the output of the \((l-1)\)-th layer \((y^{l-1})\) and that of the \(l\)-th layer \((y^l)\) can be formulated as
\[y^l_p = \sum_{j=1}^{N_B^1} w_{jp} \exp \left[- \left( \sum_{i=1}^{N_B^1} \frac{(y^{l-1}_i-m_{ij})^2}{\sigma_{ij}^2} \right) \right], \tag{10}\]
where \(N_i^l\) is the input dimension of the \(l\)-th layer (output dimension of the \((l-1)\)-th layer), \(N_B^l\) is the number of receptive fields in the \(l\)-th layer; \(y^l_p\) is the \(p\)-th element of the output of \(y^l\); \(m_{ij}^l\), \(\sigma_{ij}^l\), and \(w_{jp}\) 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 \(y^l = [y^L_1, y^L_2, \cdots, y^L_M]^T \in \mathbb{R}^M\), where \(M^l\) is the feature dimension, the objective function of the DCMAC is
\[O(k) = \frac{1}{2} \sum_{l=1}^{M} [y^L_l(k) - d_l(k)]^2. \tag{11}\]

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 \((L-1)\)-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:
\[\frac{\partial O}{\partial m_{ij}} = \frac{\partial b_{ij}^{L-1}}{\partial m_{ij}} \frac{\partial O}{\partial b_{ij}^{L-1}}, \tag{12}\]
where \(b_{ij}^{L-1}\) is the \(p\)-th receptive field function for the \((L-1)\)-th layer. We define the momentum \(\delta_{bj}^{L-1} = \frac{\partial O}{\partial b_{ij}^{L-1}}\) of the \(p\)-th receptive field function in the \((L-1)\)-th layer. Then, we have
\[\delta_{bj}^{L-1} = \sum_{j=1}^{N_B^L} \frac{\partial b_{ij}^{L}}{\partial b_{ij}^{L-1}} \frac{\partial O}{\partial b_{ij}^{L}}. \tag{13}\]
\[
\delta^{(L-1)} = \sum_{j=1}^{M^{(L-1)}} \frac{\partial y^{(L-1)}}{\partial b^{(L-1)}} \delta^{(L)}
\]
where \(b^{(L-1)}\) is the \(j\)-th receptive field function for the \(L\)-th layer, \(y^{(L-1)}\) is the \(t\)-th element of the \(y^{(L-1)}\), \(N^{(L-1)}\) is the number of receptive fields in the \(L\)-th layer, and \(M^{(L-1)}\) is the feature dimension of \(y^{(L-1)}\). Notably, by replacing \(z\) with \(m\) and \(\sigma\) in Eq. (13), we obtain momentums \(\delta^{(L)}\) and \(\delta^{(L-1)}\).

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

\[
\delta^{(L-2)} = \frac{\partial y^{(L-2)}}{\partial p} \left[ \right]
\]

\[
= \sum_{t=1}^{t=1} \frac{\partial y^{(L-2)}}{\partial b^{(L-2)}} \sum_{j=1}^{j=1} \frac{\partial y^{(L-2)}}{\partial t^{(L-2)}} \delta^{(L-1)}
\]

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

Based on the normalized gradient descent method, the learning algorithm of \(m^{(L)}_p\) (the \(i\)-th mean parameter in the \(p\)-th receptive field in the \((L)\)-th layer) is defined as

\[
m^{(L)}_p(k + 1) = m^{(L)}_p(k) + \mu m^{(L)} \frac{\partial b^{(L)}}{\partial m^{(L)}} \delta^{(L)}
\]

Similarly, the learning algorithm of \(\sigma^{(L)}_i\) (the \(i\)-th variance parameter in the \(j\)-th receptive field in the \((L)\)-th layer) is defined as

\[
\sigma^{(L)}_i(k + 1) = \sigma^{(L)}_i(k) + \mu \sigma \frac{\partial b^{(L)}}{\partial \sigma^{(L)}} \delta^{(L)}
\]

where \(\mu_m\) in Eq. (15) and \(\mu_o\) in Eq. (16) are the learning rates for the mean and variance updates, respectively.

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:

\[
\frac{\partial o}{\partial w_1} = \frac{\partial y^{(L-1)}}{\partial w_1} \frac{\partial o}{\partial y^{(L-1)}}
\]

where \(y^{(L-1)}\) is the \(p\)-th element of the \(y^{(L-1)}\). Then, we define the momentum for the \((L-1)\)-th layer \(\delta^{(L-1)}\) as:

\[
\delta^{(L-1)} = \sum_{j=1}^{N^{(L-1)}} \frac{\partial b^{(L-1)}}{\partial y^{(L-1)}} \sum_{t=1}^{t=1} \frac{\partial y^{(L-1)}}{\partial t^{(L-1)}} \delta^{(L)}
\]

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

\[
\delta^{(L-2)} = \sum_{j=1}^{N^{(L-2)}} \frac{\partial b^{(L-2)}}{\partial y^{(L-2)}} \sum_{t=1}^{t=1} \frac{\partial y^{(L-2)}}{\partial t^{(L-2)}} \delta^{(L-1)}
\]

\[
= \sum_{j=1}^{N^{(L-2)}} \frac{\partial b^{(L-2)}}{\partial y^{(L-2)}} \sum_{t=1}^{t=1} \frac{\partial y^{(L-2)}}{\partial t^{(L-2)}} \delta^{(L-1)}
\]

The noise signal \(n(k)\) will go through a non-linear channel generating the damage noise \(z(k)\). The relation between \(n(k)\) and \(z(k)\) is \(z(k) = F(n(k))\), where \(F()\) represents the function of the non-linear channel. In this experiment, we used 12 different functions, \(0.6 \cdot (n(k))^{2i-1} ; 0.6 \cdot \cos((n(k))^{2i-1}) ; \sin((n(k))^{2i-1}) \cdot 1, 2, 3, 4\) 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, \(F = 0.6 \cdot (n(k))^3\), \(F = 0.6 \cdot \cos((n(k))^3)\), and \(F = 0.6 \cdot \sin((n(k))^3)\) 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 (\(N_R=8\): \(Cell(5 \times (N_e)/4\) (\(AS_{layer}\)) \times 4 \(AS_{layer}\) = 8.
3. Number of receptive fields (\(N_R=8\).
4. Associative memory functions: \(\varphi_{ij}\) = \(exp[-(x_i - m_{ij})^2 / \sigma_{ij}^2]\), \(i = 1, 2, \ldots, N_R\).

Note that \(Cell()\) 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 \(LB=B=3-1\) 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 \(3-1\) 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: \(m_{i1} = -2.4\), \(m_{i2} = -1.8\),
\(m_{i3} = -1.2\), \(m_{i4} = 0.6\), \(m_{i5} = 1.2\), \(m_{i6} = 1.8\), \(m_{i7} = 2.4\), so that the eight blocks can cover \([UB\ LB]\)
more evenly. Meanwhile, we set \(\sigma_j = 0.6\) for \(j=1...8\), and
the initial weights \((w_{ij})\) 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 \(\mu_1 = \mu_2 = \mu_w = \mu_m = \mu_p = 0.001\)
(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 \(M^k = 1\), respectively.

This section compares DCMAC with different architectures
based on two performance metrics, the MSE and the conver-
gence 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 re-
results being demonstrated from the left to right in Fig. 6. To
compare the performance of the proposed DCMAC, we have con-
ducted experiments using two popular adaptive filter methods,
namely LMS [1] and the Volterra filter [5, 6]. For a fair com-
parison, 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.

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 ob-
served 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 reduc-
tion 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 con-
verged 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 satu-
rated around 400 epochs. In a real-world application, a de-
velopment 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.
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

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

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 VAIRANCE OF 10 log_{10}(MSE) SCORES FOR LMS, VOLTERRA, CMAC, AND DCMAC OVER 12 CHANNEL FUNCTION

<table>
<thead>
<tr>
<th></th>
<th>LMS</th>
<th>Volterra</th>
<th>CMAC</th>
<th>DCMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>-4.35</td>
<td>-5.05</td>
<td>-7.01</td>
<td>-7.59</td>
</tr>
<tr>
<td>Variance</td>
<td>11.95</td>
<td>11.57</td>
<td>1.08</td>
<td>0.19</td>
</tr>
</tbody>
</table>

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 the training 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 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


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.
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.