Closed-loop method to improve image PSNR in pyramidal CMAC networks

Hung-Ching Lu*
Department of Electrical Engineering, Tatung University, No. 40, Sec. 3, Chung Shan North Road, Taipei, Taiwan 104
Fax: (886)-2-25941371 E-mail: luhung@ttu.edu.tw
*Corresponding author

Ted Tao
Department of Electrical Engineering, Kuang Wu Institute of Technology, No. 151, I-te Street, Taipei, Taiwan 112
E-mail: tedtao@ms15.hinet.net

Abstract: A closed-loop method to improve image the peak signal to noise ratio (PSNR) in pyramidal cerebellar model arithmetic computer (CMAC) networks is proposed in this paper. We propose a novel coding procedure, which can make the CMAC network learn the feature of the transmitted image with only one-shot training, so some sampled data of the original image can quickly be sent to reconstruct a coarse image. In the meantime, differential codes are transmitted to improve the image quality using the closed-loop method in pyramidal CMAC networks. As a result, the quality of the reconstructed image can be improved at the bottom of the pyramidal CMAC networks. Finally, the experimental results demonstrate that the proposed method can give higher PSNR at a lower bit rate after reconstruction, when it is applied to JPEG compression.

Keywords: CMAC; image compression; pyramid; PSNR.


1 INTRODUCTION

Image compression is a key technology in the development of various multimedia computer services and telecommunication applications, such as teleconferencing, digital broadcast code, and video technology etc. The intention of image compression is to reduce the volume of data without a significant loss of visual quality. These are two more types of image compression techniques:

1 transforms include discrete cosine transform (DCT), fast Fourier transform (FFT), joint photographic expert group (JPEG), and Wavelet transform etc. (Ahmed et al., 1974; Arerbuch et al., 1996; Jiang, 1999)

2 non-transforms include pulse code modulation (PCM), differential pulse code modulation (DPCM), and vector quantisation (VQ) etc. (Cierniak and Rutkowski, 2000; Mas Ribes et al., 2001; Merhav and Bhaskaran, 1997).

The latter techniques always combine with the former techniques to increase the compression ratio, but the peak signal to noise ratio (PSNR) will sometimes decrease. To compensate for the above disadvantage, indirect neural network applications are developed to assist with traditional techniques and show great potential for further improvements on conventional image coding and compression algorithms (Mas Ribes et al., 2001). Improvement can be expected by using neural networks, since their application in this area are well established (Merrill and Port, 1991; Mougeot et al., 1991; Namphol et al., 1996).

J.S. Albus proposed the cerebellar model articulation controller or cerebellar model arithmetic computer (CMAC) in 1975 (Albus, 1975a,b), which can be seen as an associative memory neural network based on a look-up table method. There are several advantages, including local generalisation, good learning capacity, and rapid learning convergence, which have been

Copyright © 2006 Inderscience Enterprises Ltd.
demonstrated (Lin and Chiang, 1997; Wong and Sideris, 1992). They are easy to implement in automatic control (Chen and Chang, 1996; Miller et al., 1990; Tao et al., 2002) and signal processing etc. (Iiguni, 1996; Tao et al., 2003). However, the use of conventional CMAC networks in the above literature still needs several cycles or epochs to converge. If neural networks can learn just with one-shot training, this will save reconstructive time in the image compressive domain. So, we propose a novel coding procedure, which can feedback the error between the desired value and the actual output value in order to improve the reconstructed image quality in the pyramidal closed-loop CMAC networks. Consequently, the PSNR of the reconstructed image becomes higher at the bottom of the pyramid.

2 TWO-DIMENSIONAL CMAC STRUCTURE

We will briefly introduce the structure and learning algorithm of a two-dimensional CMAC in order to learn image data in this section. In general, let \( v = (x, y) \) be the input vector and \( w_{v(i)} \) be the stored weight in the \( j \)-th layer memory cells (or so called hypercubes) which are mapped by input vector in the \( K \)-layer CMAC network (Albus, 1975b; Lin and Chiang, 1997; Tao et al., 2002). Then the output value \( g(x, y) \) can be computed in Equation (1).

\[
g(x, y) = \sum_{j=1}^{K} w_{v(i)}. \tag{1}
\]

Since each pixel addresses exactly \( K \) hypercubes, only those weights are used in Equation (1) for each input in the CMAC network. Because the output value \( g(x, y) \) is generated from the CMAC hypercubes, the data of hypercubes need to be updated by the differential code \( e(x, y) \), that is the difference between the desired value \( d(x, y) \) and output value \( g(x, y) \), in the learning process. The updating algorithm for \( w_{v(i)} \) is defined by the following equations:

\[
w_{v(i)}^{new} = w_{v(i)}^{old} + \frac{u}{K} e(x, y); \quad j = 1, 2, \ldots, K; \tag{2}
\]

\[
e(x, y) = d(x, y) - g(x, y), \tag{3}
\]

where \( u \) is a learning rate in the learning process, and according to Lin and Chiang (1997) and Tao et al. (2002) if \( 0 < u < 2 \) is true, then the above learning algorithm will converge.

Suppose there are \( N \) discrete units to be distinguished for each dimension in the CMAC network. Then there are \([ceil((N + j - 1)/K)]^2 \) hypercubes in the \( j \)-th layer CMAC network where the function \( ceil(x) \) rounds the elements of \( x \) to the nearest integer towards infinity. Thus the total number of hypercubes in a \( K \)-layer CMAC network is described in the following equation:

\[
N_m = \sum_{j=1}^{K} [ceil((N + j - 1)/K)]^2. \tag{4}
\]

It is seen that there are only \( N_m \) hypercubes needed to distinguish \( N^2 \) pixels. The significant feature of the CMAC network is that the learning algorithm changes the output values for the neighbouring inputs. Similar inputs lead to similar output even for untrained inputs because each hypercube covers \( K \) inputs. This property is called generalisation (Thompson and Kwon, 1995), which is of great use in the CMAC based coding. Moreover, we can control the degree of generalisation by changing the size of \( K \). The larger \( K \) is then the wider the generalisation region is.

3 THE CODING PROCESS AND ONE-SHOT TRAINING IN THE CMAC NETWORK

In Section 2, the updating algorithm must go through the input vector to address hypercubes whose weights will be updated. Here, we propose an addressing function to code those addressed hypercubes. In our approach, the addressing vector \( v_i \) is used to simultaneously generate the indices of \( K \)'s addressed hypercubes. Let us define the associative input vector \( v_i = (x_i, y_i) \) to describe the mapping \( D \rightarrow M \) and the inverse mapping \( M \rightarrow G \): where \( D \) is the domain of the input signal \( d(x_i, y_i) \); \( v_i \) is the addressing vector mapping the input signal \( d(x_i, y_i) \) into \( K \)'s hypercubes; \( w_{v(i)} \in M \) is the weight of the \( j \)-th layer hypercube mapped by \( v_i \); \( G \) is the output domain; and the actual output \( g_i(x, y) \) equals the summation of \( K \)'s weights mapped by vector \( v_i \). Mapping the input signal to the associative memory, \( D \rightarrow M \), can be regarded as the encoding process, and the inverse mapping from the mapped memory to the actual output value, \( M \rightarrow G \), can be regarded as the decoding process during the reconstruction process. In our approach, the addressing vector \( v_i \) is used to simultaneously generate the indices of \( K \)'s addressed hypercubes. Take a two-dimensional CMAC network as an example, and there are \( N \) discrete units to be distinguished in each dimension. If there are \( K \) layers in a CMAC network, it needs only \( N_m \) hypercubes to distinguish \( N^2 \) pixels as described in Equation (4). Now, consider a signal \( d(x_i, y_i) \) representing the quantised value of the pixel \( (x_i, y_i) \), and define the addressing function \( v_i(j) \), which is produced by the pixel in the \( j \)-th layer for \( j = 1, 2, \ldots, K \). Then the addressing function \( v_i(j) \) for \( j = 1, 2, \ldots, K \), can be generated from the following function:
\[ v_j = a(x_i) + (a(y_j) - 1) \times \left[ \text{ceil}(N + j - 1)/K \right] \\
+ (j - 1) \times \left[ \text{ceil}(N + j - 1)/K \right]^2. \]

With this addressing function \( v_j \), \( N^2 \) states can be mapped into \( N_m \) hypercubes. When a signal \( d(x_i, y_j) \) is quantised, the addressed hypercubes can be obtained directly with the above addressing function. Thus, the required data extraction or data updating can be performed with those hypercubes directly, both in the encoding process and the decoding process.

After proposing the coding processes, we will discuss training methods in the CMAC network. In general cases, the learned information will be more accurate by repeatedly training. But repeatedly training costs more time than one-shot training. Here, we propose a one-shot training method to take the place of repeated training method. Through the least trained samples, the one-shot training method can train all weights in the CMAC network, and its error will be smaller than repeated training method. For the convenience of explaining one-shot training, we only consider a one-dimensional mathematic function (Thompson and Kwon, 1995), as described in Equation (6), where the constraint is that the mathematical function must remain smooth within the resolution of the CMAC.

\[
d(x) = 2 \left[ \sin(x) - \frac{1}{2} \sin(2x) + \frac{1}{3} \sin(3x) \right] \\
\times \left[ \cos(x) - \frac{1}{2} \cos(2x) + \frac{1}{3} \cos(3x) \right]
\]

for \(-180^\circ \leq x \leq 180^\circ\).

Now let the variable \( x \) be quantised to 361 discrete units (i.e. each degree as a unit), the number of layers be four, and \( g(x) \) be the learned actual output in the CMAC networks. If we choose every two degrees as a trained sample, there will be 181 samples to be trained. After one-shot training the mean square error (MSE) for all discrete degrees including untrained samples equals 0.1636 as shown in Figure 1(a), and it becomes 0.0185

![Figure 1](image)
after twice training in Figure 1(b). If we only choose every 4 degrees as a trained sample, then the total number of trained samples equals 91 which is almost half of Figure 1(a) and (b). The result of one-shot training is shown in Figure 1(c) and its MSE = $3.895 \times 10^{-5}$ for all discrete degrees including untrained samples which is far smaller than 181 trained samples. Compare these results for ten cycles as shown in Figure 1(d). We notice that the MSE of 91 trained samples is smaller than that of 181 trained samples in the first four cycles although the latter is a little smaller after five cycles. It violates the training rule where errors should decrease with an increasing number of trained samples, because later updating when it shares the same weights with the previous trained samples may corrupt the previous learned information. If we choose every two degrees as a trained sample in four-layer CMAC, then weights of $x = 1$ and $x = 3$ are shared by weights of $x = 3$ and $x = 5$ and so on. When weights of $x = 3$ and $x = 5$ are updated, they may corrupt information in $x = 1$ and $x = 3$. Though the training process lasts for several cycles, this situation may actually be smoothed out. However, when one-shot training is required, there may not be enough time for smoothing out the corrupted data. If we choose trained samples such that every weight is updated by only one time in each learning cycle, then the weights in a CMAC network will not be corrupted. Here we choose every 4 degrees as a trained sampled distance in a four-layer CMAC network, and the error of its learned result is quite small as shown in Figure 1(c) with MSE = $3.895 \times 10^{-5}$. Figure 1(d) reveals that CMAC networks will stop learning (i.e. the MSE will not decrease with increasing trained cycles), when the number of sampled distance is the same as the number of layers in the CMAC network. In order to prevent the phenomenon of stop learning, we propose closed-loop pyramid CMAC networks where the information can still be updated in the lower level's of the CMAC without corrupting previously learned information, which can lessen the disadvantage. This will be discussed in detail in Section 4.

### 4 CLOSED-LOOP METHOD IN PYRAMIDAL CMAC NETWORKS

In this section, we propose a closed-loop method in pyramid CMAC networks, whose block diagram is illustrated in Figure 2. The volume of transmitted data is

![Figure 2](image-url)
like a pyramid: in other words, the volume of transmitted data on the top of the pyramidal CMAC network is the least, so a coarse image can be reconstructed quickly. The pyramidal architecture contains $m$ levels, and each level is composed of four processes: downsampling, compression (CR), decomposition (DC), and upsampling. Note that an independent CMAC network for different levels does the upsampling process, so it ensures that later information can still be updated but not corrupt the learned information updated in higher levels of the CMAC networks. The original image data are used as initial differential codes $e_0$ in the first level, then the differential codes are selected every $2^m$ units in each dimension (downsampled data are $1/4^m$ of whole image data), and then the sampled differential codes, through a compression process, become compressed data $\tilde{e}_0$, which will be transmitted to the receiver. When the receiver accepts the compressed data $\tilde{e}_0$, which should be decomposed, and the decomposed data $e_0$ are upsampeled by the first level CMAC network (CMAC(1)) they then become the reconstructed data $g_1$. The original data, $e_0$, minus reconstructed data, $g_1$, are simultaneously sent to the next level as the second level’s differential codes $e_1$. These four closed-loop processes repeat $m$ times in the $m$ level pyramidal CMAC networks, but the downsampling and upsampling ratio are different in different levels. The downsampling ratio equals $1/4^{m-i+1}$, and the upsampling ratio equals $4^{m-i+1}$ in the $i$-th level, but it does not need to downsample and upsample in the last level because all the differential codes should be transmitted.

Let CMAC$(i)$ be a CMAC network in the $i$-th level of the pyramidal CMAC network, so hypercubes are also divided into $m$ levels, and each level contains an independent network (CMAC$(i)$) whose weights of hypercubes should not interfere between different levels. Let the $i$-th level CMAC network (CMAC$(i)$) include $K_i$ layers, then the number of hypercubes $N_i$ used in $i$-th level can be computed by the following equation:

$$N_i = \sum_{j=1}^{K_i} \left[\text{ceil}((N + j - 1)/K_i)\right]^2;$$

$$K_i = 2^{m-i+1}; \quad i = 1, 2, \cdots, m-1,$$

where $N$ is the number of pixels in each dimension.

In order to explain the encoding and decoding processes in a closed-loop method in pyramidal CMAC networks, we describe these two processes in the following two subsections.

### 4.1 The encoding process

After proposing the diagram of closed-loop pyramidal CMAC networks, we need to encode in the compression process. In view of the diagram, the input data from $e_{i-1}$ to $\tilde{e}_{i-1}$ is considered as the selecting and quantising process in the $i$-th level. When we select the pixels for compressing, the distance between every selected point is $K_i = 2^{m-i+1}$ points along each dimension in the $i$-th level. If there are $N \times N = 2^n \times 2^n$ pixels in the original image, there will be $2^{2(n-m+i-1)}$ samples trained in the $i$-th level, which are $1/K_i^2 = 1/4^{m-i+1}$ of whole image data. In the $m$-th level, all pixels are sampled for compressing, and all pixels can be trained again in additional level in order to improve PSNR.

In the encoding process the $i$-th level CMAC$(i)$ has $K_i$ layers, and the sampled pixels are each $K_i$ points along each dimension in this level. It is impossible to interfere with each other for CMAC learning because every hypercube is selected just once; thus it can update weights of CMAC in the same level without corrupting the learned information. And the hypercubes of different level CMAC networks are independent, so the learned information updated in the same level will not interfere with each other.

### 4.2 The decoding process

The decoding process is described in the following equations:

$$e_0(x, y) = d(x, y);$$

$$e_i(x, y) = e_{i-1}(x, y) - g_i(x, y), \quad \text{for } i = 1, 2, \cdots, m;$$

$$g_i(x, y) = \sum_{j=1}^{K_i} w_{v(i, j)} K_i = 2^{m-i+1}, \quad \text{for } j = 1, 2, \cdots, K_i;$$

$$w_{v(i, j)} = w_{v(i, j)}^0 + \frac{\mu}{K_i} \tilde{e}_i(x_i, y_i), \quad \text{for } i = 1, 2, \cdots, m-1. \quad (11)$$

The number of decomposed differential codes $\tilde{e}_i(x_i, y_i)$ is $1/1/K_i^2$ of original image data should be encoded in $i$-th level of CMAC networks. Though one division exists in Equation (11), but the computation time can be regardless when the number $K_i$ is power of 2 when it is performed with a shift operation. From Equations (9) to (11), it costs $2/K_i$ additions per pixel in the $i$-th level because it costs $2K_i$ additions at $1/K_i^2$ ratio of original data in the compression and decomposition processes.
From Equations (12) and (13), it needs only one addition per pixel to improve the PSNR of the reconstructed image in each level. Now the error after $m$ level reconstruction process is discussed, and it can be computed by using the following equation:

$$e_0(x, y) - h_m(x, y) = e_0(x, y) - \sum_{i=1}^{m} g_i(x, y) = e_{m-1}(x, y).$$  \hspace{1cm} (14)

The decomposed codes $\hat{e}_{m-1}(x_i, y_i)$ will approach the downsampled part of differential codes $e_{m-1}(x, y)$, when a good transform such as DCT, Wavelet and JPEG etc., is used. The decomposed differential codes $\hat{e}_{m-1}(x_i, y_i)$ will converge if the learning rate $u$ satisfies condition $0 < u < 2$ in Equation (11), which has been proved (Lin and Chiang, 1997; Tao et al., 2002), so the error $e_{m-1}(x, y)$ will converge too. Thus, it can be concluded that the lower the level of pyramidal CMAC networks is, the smaller the errors $e_{m-1}(x, y)$ will be. If the value of differential codes is smaller, then the compressed data volume will also be less (Mougeot et al., 1991). Thus, we can use smaller differential codes to get a higher compression rate in the bottom of the pyramidal CMAC networks.

5 EXPERIMENTAL RESULTS

The experiment results are described as follows. An image is downsampled and compressed through JPEG and then is coded into the proposed pyramid CMAC. After transmission, the received image is decoded through CMAC and then JPEG to reconstruct the image. Now, we take a $512 \times 512$ size standard ‘Lena’ as an original image whose data volume equals 258 KB and grey levels equal 256. In order to measure the quality of the reconstructed image, the peak signal to noise ratio (PSNR) is defined in Equation (15), where the grey level of image ranges from 0 to 255 and MSE is the abbreviation of mean square error for all pixels including untrained pixels.

$$\text{PSNR} = 10 \times \log \left( \frac{255}{\text{MSE}} \right) \text{ dB.}$$  \hspace{1cm} (15)

The JPEG method is applied to the proposed pyramidal CMAC network as compression (CR) and decomposition (DC) processes. The output of the JPEG compression supposedly ensures the smooth requirement as mentioned in Section 3. The proposed method not only can reduce the bit rate during transmission, but also can improve PSNR in the lower level of the pyramid. The level of pyramid CMAC networks is set to be 3 ($m = 0 \sim 2$) and learning rate $u$ is set to be 1 in the following experiments.

In the first level the original image whose data volume is 258 KB is used as initial differential codes ($e_0$), and the downsampled differential codes, whose size is 1/16 of original image through JPEG compression (CR = 1/3.225), become $\hat{e}_0$ whose data volume equals 5 KB and bit rate $H = \frac{8}{16 \times 3.225} = 0.115$ (bits/pixel). Then the compressed data through JPEG decomposition and reconstruction processes CMAC(1) become a coarse image as shown in Figure 3 ($h_1 = g_1$) with PSNR = 26.93 dB which is calculated by Equation (15).

In the second level the original image data minus the first level’s reconstructed data ($e_0 - g_1$) are used as second level’s differential codes ($e_1$), and the downsampled differential codes whose size is 1/4 of original image through JPEG compression become $\hat{e}_1$ whose data volume is 11 KB and bit rate $H = 0.3411$ (bits/pixel). Then the compressed data through JPEG decomposition and reconstruction processes CMAC(2) become the reconstructed differential codes $g_2$ which must add to the first level’s reconstructed image to become this level’s output image ($h_2 = h_1 + g_2$) whose PSNR equals 31.38 dB as shown in Figure 4.
obvious that the quality of the reconstructed image is improved, but the cost is increased bit rate in the second level.

In the third level the previous level’s differential codes minus the reconstructed data \((e_1 - g_2)\) are used as this level’s differential codes \((e_2)\), which through JPEG compression become \(\tilde{e}_2\) whose data volume is 21 KB and bit rate \(H = 0.6512\) (bits/pixel). Finally the compressed data through the JPEG decomposition process becomes reconstructed differential codes \(g_3\), which must add to the second level’s reconstructed image become this level’s output image \((h_3 = h_2 + g_3)\) whose PSNR equals 35.60 dB in Figure 5. In this level all the differential codes are transmitted, so they need neither the downsampling process nor CMAC reconstruction. Because the value of the differential codes become smaller in this level, the bit rate \(H = 0.6512\) (bits/pixel) becomes smaller too. In addition, if the quality of the reconstructed image is not satisfied, then an additional level can be utilised to improve PSNR again. Thus, the PSNR of the reconstructed image can be improved to 38.6 dB as shown in Figure 6, and its bit rate only increases to 0.9922 (bits/pixel).

6 CONCLUSIONS

In this paper, a closed-loop method to improve image PSNR in pyramidal CMAC networks is proposed in the compressive domain, and JPEG is applied to the presented method. The coarsest reconstructed image will be quickly produced, and the image quality can be improved by closed-loop pyramidal CMAC networks. Meanwhile, a novel coding method proposed in pyramidal CMAC networks makes one-shot training possible and requires less sample data. Due to the generalisation ability of CMAC networks, the sizes of all reconstructed images are equal to the original image size. Because of the above advantages, the proposed method not only can decrease the bit rate, but also can improve PSNR in the compressive domain.

ACKNOWLEDGEMENT

This work was supported by the National Science Council of the Republic of China under the contract of NSC-91-2213-E-036-010.

REFERENCES


Biographical notes:

Hung-Ching Lu was born in Taipei, Taiwan, in 1959. He received his BS degree in Electrical Engineering from the Tatung Institute of Technology in 1984, and received his MS and PhD degrees in Electrical Engineering from the Tatung Institute of Technology, Taipei, Taiwan in 1986 and 1989, respectively. He is currently a professor in the Department of Electrical Engineering at the Tatung University. His research interests include fuzzy control, robust control, neural networks, CMAC, and their applications for industrial design.

Ted Tao was born in Taipei, Taiwan, in 1959. He received his BS degree in Industrial Education from National Taiwan Normal University in 1984, and received his MS and PhD degrees in Electrical Engineering at Tatung University in 1990 and 2004, respectively. He is currently an associate professor in the Department of Electrical Engineering at Northern Taiwan Institute of Science and Technology. His research interests include neural network, CMAC, learning techniques, system control, and image processing.