PSECMAC: A Novel Self-Organizing Multi-Resolution Associative Memory Architecture

S. D. Teddy, C. Quek, and E. M.-K. Lai, *Senior Member, IEEE*

Abstract

The cerebellum constitutes a vital part of the human brain system that possesses the capability to model highly nonlinear physical dynamics. The CMAC associative memory network is a computational model inspired by the neurophysiological properties of the cerebellum, and it has been widely used for control, optimization and various pattern recognition tasks. However, the CMAC network's highly regularized computing structure often leads to: (1) a suboptimal modeling accuracy; (2) poor memory utilization; and (3) the generalization–accuracy dilemma. Previous attempts to address these shortcomings have limited success and the proposed solutions often introduce a high operational complexity to the CMAC network. This paper presents a novel neurophysiologically-inspired associative memory architecture named PSECMAC that non-uniformly allocates its computing cells to overcome the architectural deficiencies encountered by the CMAC network. The non-uniform memory allocation scheme employed by the proposed PSECMAC network is inspired by the cerebellar experience-driven synaptic plasticity phenomenon observed in the cerebellum, where significantly higher densities of synaptic connections are located in the frequently-accessed regions. In the PSECMAC network, this biological synaptic plasticity phenomenon is emulated by employing a data-driven adaptive memory quantization scheme that defines its computing structure. A neighborhood based activation process is subsequently implemented to facilitate the learning and computation of the PSECMAC structure. The training stability of the PSECMAC network is theoretically assured by the proof of its learning convergence which will be presented in this paper. The performance of the proposed network is subsequently benchmarked against

Manuscript received xxxxx; revised xxxxx; accepted xxxxx

S. D. Teddy is with the Data Mining Department, Institute for Infocomm Research, Singapore, 119613. The work in this paper was performed during her PhD candidature at the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore.

C. Quek is with the Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, 639798 (email: ashcquek@ntu.edu.sg).

E. M.-K. Lai is with the Institute of Information Sciences and Technology, Massey University, Wellington, New Zealand.

the CMAC network and several representative CMAC variants on three real-life applications, namely: pricing of currency futures option, banking failure classification, and modeling of the glucose–insulin dynamics of the human glucose metabolic process. The experimental results have strongly demonstrated the effectiveness of the PSECMAC network in addressing the architectural deficiencies of the CMAC network by achieving significant improvements in the memory utilization, output accuracy as well as the generalization capability of the network.

Index Terms

PSECMAC, CMAC, cerebellum, brain-inspired, multi-resolution, associative memory, learning convergence, neural networks.

I. Introduction

The human brain is the underlying biological structure responsible for human intelligence, in which complex networks of neurons collaborate in a highly non-linear manner to create a massive information computing system. The cerebellum is one brain region in which the neuronal connectivity is sufficiently regular to facilitate a comprehensive understanding of its functional properties. It is located at the bottom rear of the head (the hind-brain) and constitutes a vital part of the brain system that mediates motor movement control and a number of sub-conscious cognitive functions [1], including the learning and memory of procedural and motor skills. The human cerebellum functions as a motor movement calibrator [2] and possesses the capability to model highly complex and nonlinear physical dynamics to facilitate the precise and rapid executions of dexterous movements and fluid motor reflexes [3]. Hence, it is highly desirable to construct a computational model of the human cerebellum in order to capture and to emulate its rapid and nonlinear function learning capability. Such a computational tool has diverse use in applications such as autonomous control and pattern recognition where there are generally no precise mathematical descriptions of the problem's characteristics and the inherent process behavior can only be inferred from measurable physical observations.

The Cerebellar Model Articulation Controller (CMAC) [4] is a neural network inspired by the neurophysiological properties of the human cerebellum and is widely recognized for its localized generalization and rapid algorithmic computations. As a computational model of the human cerebellum, CMAC manifests as an associative memory network [5], [6], and employs error correction signals to drive the network learning and memory formation processes. This allows for simple computation, fast training, local generalization and ease of hardware implementation [2], [7], which subsequently motivates the prevalent use of CMAC-based systems for process control and optimizations [8]–[13], modeling and

control of robotic manipulators [14]–[16], as well as various signal processing and pattern-recognition tasks [17]–[19]. The learning convergence of the CMAC network has also been established in [20]–[22]. However, there are several major architectural limitations associated with the CMAC network, which arise from the rigidness of its computing structure. The CMAC associative memory network employs a highly regularized grid-like computing structure (i.e. equally spaced memory cells along each input dimension) that indirectly enforces the uniform quantization of a problem's input-output (I/O) mapping space. On the other hand, meaningful real-life applications are generally *heteroskedastic*, where the problems are often characterized by highly nonlinear I/O trends and statistically varying data patterns. Such observations implied that specific regions of these problems' I/O associative spaces are more informative (and therefore demand a higher modeling resolution) than others. For such an application, the simplistic approach of adopting uniformly quantized I/O mapping space (as in the memory space of CMAC) to model the problem's input-output data characteristics may not be adequate as it often leads to: (1) a suboptimal system where there is a lack of modeling accuracy at the important regions of the I/O mapping space; (2) poor memory utilization as characterized by a large number of untrained memory (computing) cells (when no memory hashing is employed); and (3) a trade-off between the generalization capability and the modeling fidelity of the network. That is, a small-sized CMAC with fewer memory cells is able to better generalize the characteristics of the training data, but a large-sized CMAC network with fine modeling resolution produces more accurate outputs.

In the existing literature, there has been a number of attempts to address such limitations of the CMAC network. Generally, these efforts can be broadly classified into two major approaches: the multi-resolution discrete CMAC and the fuzzy CMAC variants. The multi-resolution discrete CMAC variants employ computing cells with crisp boundaries and attempt to produce a more efficient mapping of the I/O associative space via the optimization of the network quantization decision functions [23]–[26] or with the use of multi-layered CMAC networks of increasing resolutions [27], [28]. The fuzzy CMAC systems, on the other hand, employ fuzzified cell boundaries of varying sizes to enhance memory efficiency [19], [29]–[34] as well as ensuring network interpretability via the incorporation of formalized fuzzy inference schemes [35]–[39]. These CMAC extensions will be briefly discussed in Section III-C. However, such approaches have limited success in addressing CMAC's architectural shortcomings and often do so at the expense of introducing a high operational complexity. Most of these variants also do not attempt to establish proof of the system's learning convergence, which is often crucial for control and function approximation tasks. Moreover, since human behavioral studies have established that learning stability in the human cerebellum is central to the acquisition and the subsequent execution of smooth and

precise motor movements [40]–[42], any credible computational model of the cerebellum should therefore guarantee a stable learning process. In addition, although various memory hashing techniques [43], [44] have been proposed to improve the memory efficiency of a CMAC network, the use of hash-coding results in memory collisions, increases the computational complexity and distorts the computational interpretability of the resultant CMAC network [45]–[47].

In this paper, we propose a novel neurophysiologically-inspired multi-resolution associative memory network named the *Pseudo Self-Evolving CMAC* (PSECMAC) network that non-uniformly quantizes its memory cells. The proposed PSECMAC associative memory network is inspired by neuroscience and human behavioral studies on the cerebellar learning process, where it has been shown that significantly higher densities of the cerebellar synaptic connections are located at the frequently-accessed regions of the cerebellum that are activated by repeated learning episodes [48]. This cerebellar-based experience-driven synaptic plasticity phenomenon is emulated in the PSECMAC network by employing a data-driven adaptive memory quantization scheme for the derivation of its computing structure. That is, from a machine learning perspective, more memory cells are assigned to model regions of the data space that contain higher densities of the training exemplars. This seeks to avoid the architectural deficiencies in the CMAC model and justifies PSECMAC as a more efficient computing model of the human cerebellum. Unlike the CMAC network which employs a multi-layered computing structure, the proposed PSECMAC network consists of only a single-layer of computing cells. This attempts to enhance the computational comprehensibility of the PSECMAC network and to avoid the high memory requirement of the CMAC network in which extensive overlappings of the computing layers are needed to achieve a smooth output. In the PSECMAC network, the computational principles of the multi-layered CMAC network are retained via a neighborhood activation of its single layer of computing cells, and the operational similarities between the CMAC model and the proposed PSECMAC network is presented in Section III-B.

The rest of the paper is organized as follows. Section II briefly describes the construct of the human cerebellum and highlights the multi-resolution neurophysiological property of the cerebellum that inspired the development of the proposed PSECMAC network. Section III outlines the basic principles of the CMAC associative memory network, and provides a conceptual mapping of the functional similarities between the multi-layered CMAC model and the single-layered implementation of the proposed PSEC-MAC network. This is followed by a comparative survey on existing CMAC variants. Section IV presents the architecture of the PSECMAC associative memory network and the proof for learning convergence of the PSECMAC network is established in Section V. Section VI evaluates the performance of the PSECMAC network on three real-life applications, namely: (1) the pricing of the GBP vs. USD currency

futures options; (2) the classification of US banking failures; and (3) the modeling of the insulin dynamics of the human metabolic process. Section VII concludes this paper.

II. The Human Cerebellum

The human cerebellum is a brain construct that functions primarily as the movement regulator and is important for a number of motor and cognitive functions, including learning and memory [49], [50]. Together with the striatum (part of basal ganglia formation), the cerebellum constitutes the human procedural memory system, which is a facet of the brain's information computing capacity that represents a memory tract for the acquisition of skills and procedures [3]. Due to its structural neuronal organization and anatomic simplicity, the cerebellum is one of the few brain constructs where the patterns of intrinsic connections are known in considerable details [1]. This section presents the underlying anatomical and physiological characteristics that subserve the cerebellar learning and memory formation process.

A. Memory Formation in the Cerebellum – The Anatomy of the Cerebellar Circuit

The most striking feature of the human cerebellum is the near-crystalline structure of its anatomical layout. However, despite its remarkably uniform anatomical structure, the cerebellum is divided into several distinct regions. Each of these regions receives sensory information from different parts of the brain as well as the spinal cord and projects to different motor systems. Such physical connectivity suggests that different regions of the human cerebellum perform similar computational operations but on different sensory inputs [51]. In order to effectively accomplish its motor regulatory functions, the cerebellum is provided with an extensive repertoire of information about the objectives (intentions), actions (motor commands) and outcomes (feedback signals) associated with a physical movement. There are two main sets of extra cerebellar afferents: the *mossy fibers* and the *climbing fibers*, both of which carry sensory inputs from the periphery as well as sets of motor commands-related information from the cerebral cortex [52].

These cerebellar afferent inputs flow into the *granule cell* layer of the cerebellar cortex. The mossy fiber inputs, which carry both sensory afferent and cerebral efferent signals, are relayed by a massive number of granule cells. These granule cells work as expansion encoders by combining different mossy fiber inputs. Subsequently, each of the granule cells extends an ascending axon that rises up to the molecular layer of the cerebellar cortex as *parallel fiber*. These parallel fibers in turn serve as the inputs to the *Purkinje cells* at the cerebellar cortex. The Purkinje cells are the main computational units of the cerebellar cortex. Each Purkinje cell draws its inputs from the parallel fibers and the climbing fiber. The parallel fibers

input to the Purkinje cells provide large vectors of sensory and command-related information, while the climbing fibers are thought to function as training signals that regulate the modifications of Purkinje cells' synaptic weights. The parallel fibers run perpendicularly to the flat fan-like dendritic aborization of the Purkinje cells, enabling the greatest possible number of parallel fibers and Purkinje cells contacts per unit volume. The Purkinje cells perform combinations of the synaptic inputs, and their axons carry the output of the cerebellar cortex downwards into the underlying white matter and subsequently to the *deep cerebellar nuclei*. The outputs of the deep cerebellar nuclei form the overall output of the cerebellum.

In contrast to the massive synaptic connectivity of the parallel fibers to the Purkinje cells, each of the Purkinje neurons receives input from exactly *one* climbing fiber. There exists a *topographical mapping* in the synaptic connections between the cerebellar cortex, the deep cerebellar nuclei and the inferior olivary from which the climbing fibers originate [53], [54]. This topological mapping results in a modular structure or cluster known as *microzones*, which involves approximately 3000 Purkinje cells. The Purkinje cells of a microzone project to one corresponding deep cerebellar nucleus. The olivocerebellar projections to the cerebellar cortex are also arranged in a similar manner: a sub-nuclei of the inferior olive project to a microzone of Purkinje cells sharing the same target nucleus [55]. Therefore, the output of the cerebellar cortex is organized as a series of *discrete modules*, where each of them is provided with a private connection with the inferior olive [54]. Such an arrangement suggests a pattern of neighborhood activation of Purkinje cells belonging to the same microzone for each computation of output in the cerebellum.

Memory formation in the cerebellum is facilitated by the information embedded in its synaptic connections. The cerebellum corresponds to an associative memory system that performs a nonlinear mapping from the mossy fiber inputs to the Purkinje cells' outputs. The granule cell layer acts as an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells form an array of modifiable synaptic weights of the cerebellar computing system. The Purkinje cell array subsequently forms the knowledge base of the cerebellum and the output of the cerebellar memory system is generated by integrating the content of the activated Purkinje cells.

B. Learning in the Cerebellum – The Physiological Aspects of the Cerebellar Circuit

As the movement regulator, the cerebellum evaluates the disparities between the formulated intention and the executed action and subsequently adjusts the operations of the motor centers to affect and regulate the ongoing movement [56], [57]. Neuroscience has established that the cerebellum performs an associative

mapping from the input sensory afferent and cerebral efferent signals to the cerebellar output, which is subsequently transmitted back to the cerebral cortex and spinal cord through the thalamus [2], [58]–[61]. The physiological process of constructing an associative pattern map constitutes the underlying neuronal mechanism of learning in the human cerebellum.

Neuroscience research has established that the human cerebellum adopts an error-correction-driven supervised learning paradigm [51]. This implies that cerebellar learning requires extended trials with repeated exposures to similar sequence of movements in order to achieve a finely calibrated mapping between the intended and actual execution of motor movements. The existence of microzones [53], [54], as well as the established role of the climbing fibers as the teaching signals to the Purkinje cells, suggest that the cerebellar circuitry performs neighborhood-based training of the synaptic weights. That is, the cerebellar input (motor commands and sensory signals) and output (corrective error signals) pairing of a learning episode alters the synaptic weights of a cluster of Purkinje cells that is topographically defined by the corresponding microzones. This neighborhood-based learning mechanism enables a faster convergence of the cerebellar learning process, and underlies the generalization of skill learning in everyday life.

The cerebellar learning mechanism is facilitated by the modifiable synaptic transmissions (cerebellar synaptic plasticity) and the synaptic reorganization capability (cerebellar structural plasticity) of its neuronal connections. Research into the physiology of the human cerebellum has sufficiently demonstrated that the Long Term Depression (LTD) of the Purkinje cells' firing potentials in response to synaptic inputs from the parallel fibers is the underlying cellular mechanism responsible for cerebellar synaptic plasticity [51], [52], [58], [61]–[63]. However, scientific evidence suggests that synaptic depression may not be adequate for forming permanent, long term memories of motor programs [61]. Instead, there are evidences of morphological alterations of the cerebellar cortex following extensive cerebellar learning. These studies on the experience-driven cerebellar structural plasticity phenomenon have demonstrated that complex motor skill learning actually leads to an increase in the number of synapses within the cerebellar cortex [42], [48], [64], [65]. In such studies, rats were given acrobatic training by challenging them to acquire complex motor skills necessary to traverse a series of obstacles. It was discovered that rats with such training developed an increased density of the parallel fibers to Purkinje cells synapses per unit volume. The increased synaptic density was accomplished by increased dendritic aborization and increased dendritic spine densities along the Purkinje cells' spiny branchlets [65]. Such an observation constitutes a plasticity-driven biological manifestation of the multi-resolution nature of the cerebellar circuitry. Similar plasticity characteristics were also observed along the olivocerebellar pathway. The olivocerebellar axons and climbing fibers are capable of remarkable structural plasticity that is regulated

via their interactions with the Purkinje cells [55]. This neurobiological adaptation process underlies the formation of long term procedural memory at the cerebellum.

The experience-driven cerebellar structural plasticity phenomenon suggests that the cerebellum organizes its learned knowledge in an adaptive and non-linear manner, where repeated training (exposures to a particular input-output mapping association tuple) yields an increase in the synaptic connections as well as finer calibrations in the neural circuitry of the Purkinje cells. This results in the biological formation of a more precise knowledge representation scheme. These neurophysiological observations inspired the development of the proposed multi-resolution PSECMAC associative memory network.

III. CMAC as a Computational Model of the Human Cerebellum

The Cerebellar Model Articulation Controller (CMAC) associative memory network is a well-established computational model of the human cerebellum [4], [5] that was constructed to explain the information-processing characteristics of its biological counterpart. This section presents the basic computational principles of the CMAC neural network and reviews some of the CMAC variants proposed in the literature.

A. Basics of CMAC Neural Network

The CMAC network functions as a static associative memory that models the non-linear mapping between the mossy fiber inputs and the Purkinje cell outputs. The massive mesh of granule cell encoders in the cerebellum corresponds to an association layer that generates a sparse and extended representation of the mossy fiber inputs. The synaptic connections between the parallel fibers and the dendrites of the Purkinje cells forms an array of modifiable synaptic weights that motivates the grid-like CMAC computing structure. In the human cerebellum, the outputs of the activated Purkinje cells are combined to form the cerebellar output. In CMAC, the network output is computed by aggregating the memory contents of the active computing cells. The CMAC network is essentially a multi-dimensional memory array, in which an input acts as the address decoder to access the respective memory cells containing the adjustable weight parameters that constitutes the corresponding output. CMAC learns the correct output response to each input vector by modifying the contents of the selected memory locations via an error-correction learning scheme. For each input, the difference between the CMAC response and the known target response is computed and the weight values of the selected memory cells in the network are adjusted accordingly.

B. A Single Layer Model of the CMAC Network

In the original implementation of the CMAC network proposed by Albus [66], the network memory cells are divided into layers. The number of layers in a CMAC network is determined by the number

of *quantization functions* defined. That is, one quantization function corresponds to one layer. This also implicitly means that all the input dimensions have the same number of quantization functions. Figure 1(a) depicts an example of a 2-input CMAC network with four quantization functions (and therefore four layers) in each of the input dimensions. The resultant 2-dimensional grid in Figure 1(a) corresponds to the input space of the CMAC network that is used to learn the associative mapping patterns for 256 input–output vector combinations. The quantization functions of the network are defined as follows:

Along dimension S_1:

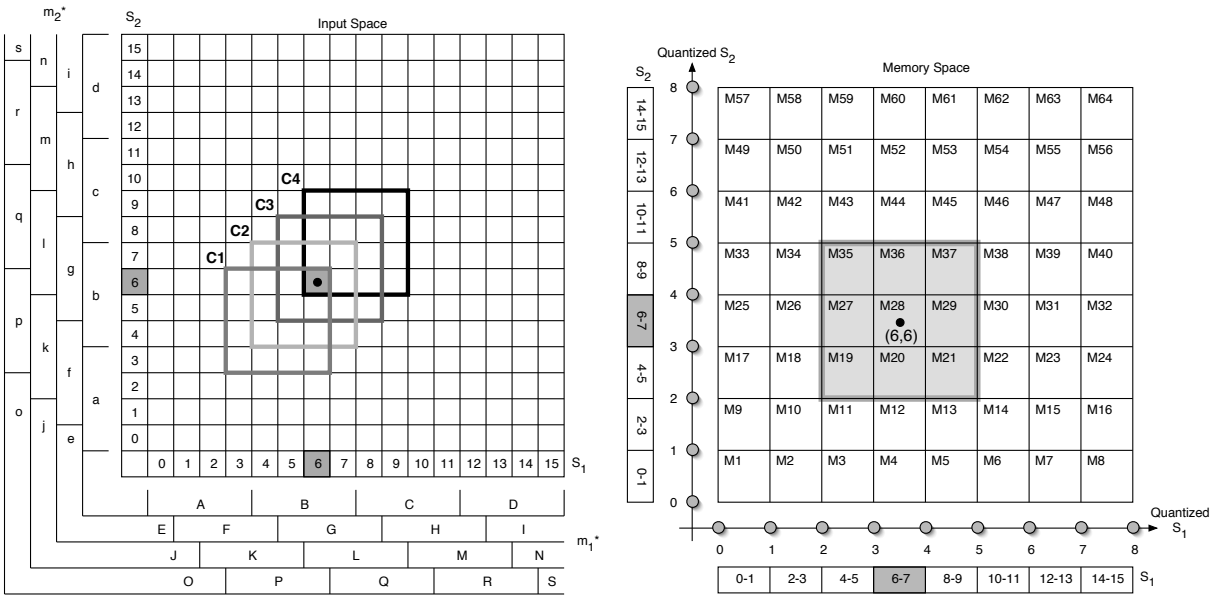
$$\begin{array}{l} {}^{S_1}Q_1 \rightarrow \{A, B, C, D\} \\ {}^{S_1}Q_2 \rightarrow \{E, F, G, H, I\} \\ {}^{S_1}Q_3 \rightarrow \{J, K, L, M, N\} \\ {}^{S_1}Q_4 \rightarrow \{O, P, Q, R, S\} \end{array} \quad (1)$$

Along dimension S_2:

$$\begin{array}{l} {}^{S_2}Q_1 \rightarrow \{a, b, c, d\} \\ {}^{S_2}Q_2 \rightarrow \{e, f, g, h, i\} \\ {}^{S_2}Q_3 \rightarrow \{j, k, l, m, n\} \\ {}^{S_2}Q_4 \rightarrow \{o, p, q, r, s\} \end{array} \quad (2)$$

where ${}^iQ_j$ denotes the quantization function for the j^{th} layer of the i^{th} dimension and $\{\}$ denotes the set of quantization levels. Each of the quantization levels (i.e. A, B, C, etc.) denotes a memory axis in the respective layer. The intersections of the memory axes of a layer constitute the memory locations for that layer. For instance, with respect to Figure 1(a), there are 16 memory cells corresponding to the pair of quantization functions ${}^{S_1}Q_1$ and ${}^{S_2}Q_1$ in layer 1. During the operations of the CMAC network, the input vector is first quantized into the corresponding quantization level at each layer, and this forms the address index used to access the memory location in that layer. Each input vector selects *one* memory cell from a layer. In the CMAC of Figure 1(a), the input vector of $(6, 6)$ selects a total of four memory cells (i.e. **C1** to **C4**). The output of the CMAC network is subsequently computed by the linear combination of the memory contents of the selected cells.

The multi-layered implementation of the CMAC network often renders the network operations difficult to comprehend. Moreover, in such an implementation, extensive layers of overlapping computing cells are required to produce a smooth output. Therefore, we proposed in this paper a generic single layer implementation with neighborhood computations to retain the modeling principles of the original multi-layered CMAC network. Figure 1(b) illustrates such a single-layered model of a 2-input CMAC network consisting of 64 memory cells. The two dimensional computing grid corresponds to the memory space of the CMAC network. In Figure 1(b), each input dimension is *quantized* into eight discrete quantization steps (or levels). Similar to the multi-layered CMAC, the input vector to this network is quantized to the corresponding level for each input dimension to obtain the index address of the winner memory cell. However, in the single-layered CMAC model, each input vector selects a neighborhood of memory cells



(a) An example of a 2D CMAC network (m_1^* refers to the set of quantization functions along the S_1 dimension and m_2^* refers to the set of quantization functions along the S_2 dimension)

(b) A single layer perspective of the 2D CMAC network example shown in Figure 1(a)

Fig. 1. Comparison of multi-layered and single-layered CMAC implementation for 2-dimensional input problem

or neurons that is centered at the winner memory cell.

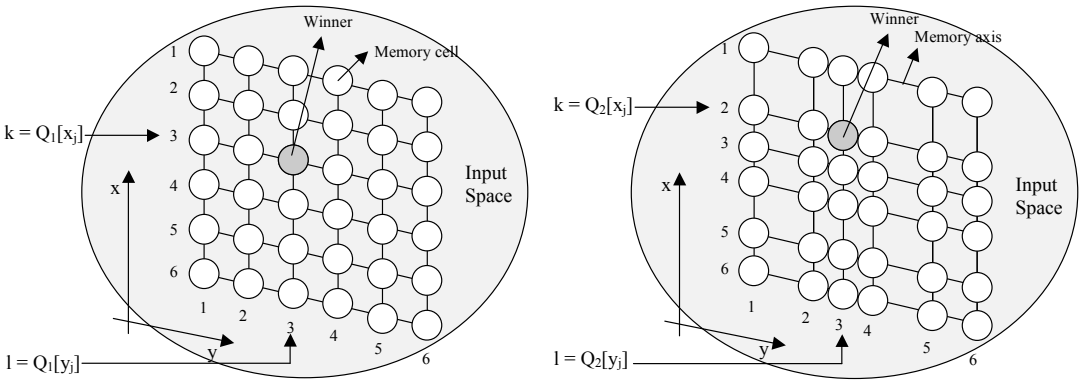
The conceptual similarities between the proposed single-layered model and the original multi-layered implementation of the CMAC network can be examined from their respective modeling principles. The layered cell activations in the original CMAC network contributed to three significant computational objectives: (1) smoothing of the computed output; (2) facilitating a distributed learning paradigm; and (3) activating similar or highly correlated computing cells in the I/O associative space. These three modeling principles are similarly conserved in the single-layered model of the CMAC network via the introduction of a neighborhood-based computational process. The activation of the neighboring cells in the input space of the single-layered CMAC corresponds to the simultaneous activation of the highly correlated cells in its multi-layered counterpart. This contributes to the smoothing of the computed output since the neighborhood-based activation process results in continuity of the network output. In addition, the distributed learning paradigm for each input-output training pair is achieved by a neighborhood update process of the single-layered CMAC model.

C. CMAC Variants

The significance of an efficient memory allocation scheme to the performance of a CMAC based system is evidenced by the number of CMAC variants proposed to address the issue in the literature. Generally, the solutions offered by the CMAC variants are of two main approaches: (1) multi-resolution based discrete quantization; and (2) fuzzy quantization of the input space. The motivation for the former is to derive an efficient mapping of the memory cells onto the input-output associative space to enhance memory usage via increasingly finer output resolutions without inducing high computational complexity of the resultant network. The fuzzified CMAC variants, on the other hand, attempt to address the rigidness of the crisp boundaries and the problem of a constant output resolution imposed by employing discrete quantization functions with the use of fuzzy membership for the quantization process. In addition, the mapping of a formalized reasoning process helps to improve the computational interpretability of the opaque CMAC network. A literature survey presented in [67] presents a brief analysis of the various fuzzified CMAC (FCMAC) architectures proposed in the literature [19], [29]–[39]. In addition, the survey [67] also briefly outlines a summary of some of the crisp CMAC variants reported in the literature [23]–[28].

IV. PSECMAC: A Self-Organizing Multi-Resolution Associative Memory Network

The proposed non-uniformly quantized PSECMAC network is inspired by the findings of neurophysiological studies of the learning phenomenon observed in the human cerebellum, where significantly higher densities of the cerebellar synaptic connections are located at the high-throughput regions of the cerebellum [42], [48], [64], [65] where frequent access facilitates the acquisition and execution of skilled behavioral responses in everyday life (i.e. experience-dependent adaptation). The PSECMAC network is a single-layered self-organizing multi-resolution computational model of the cerebellum that employs a data-driven adaptive memory quantization scheme. The experience-based synaptic adaptation process observed in the human cerebellum is emulated via the use of the Pseudo Self-Evolving Cerebellar (PSEC) [68] clustering technique that determines the data density profile of the training exemplars along each input dimension. The memory quantization step sizes of the PSECMAC network (and hence the placements of its computing cells) are subsequently adapted based on the computed information distribution of the training data. In the PSECMAC network, memory efficiency is enhanced by allocating more memory (computing) cells to the densely data-populated regions of the I/O associative space. This subsequently lowers memory wastage as the number of untrained (unused) computing cells is reduced. Meanwhile, the accuracy of the PSECMAC network's output in the high-throughput I/O subspaces (which conceptually correspond to the often-accessed regions of the cerebellar cortex) is simultaneously enhanced



(a) 2D CMAC Memory Cells (b) 2D PSECMAC Memory Cells

Fig. 2. Comparison of CMAC and PSECMAC memory surface for 2-dimensional input problem

with a higher modeling resolution.

Figure 2 graphically illustrates the fundamental architectural difference in the organization of the memory (computing) cells to define the I/O mapping space between the proposed PSECMAC network and the CMAC model. In CMAC, the memory cells are uniformly distributed over the entire associative (memory) space. The computing cells in the PSECMAC network, on the other hand, are intentionally assigned to create an efficient representation of the data distribution. In this section, the structural and parameter learning phases of the proposed PSECMAC associative memory network are presented.

A. The PSECMAC Network Architecture

The proposed PSECMAC network architecture employs a two-phased learning process, namely: structural learning and parameter tuning. The objective of the structural learning phase is to create the PSECMAC network's associative structure by computing the quantization decision functions for each input dimension. Subsequently, the input to output associative information of the training data samples are learnt by adapting the memory contents of the PSECMAC network in the parameter tuning phase. The initial step in the PSECMAC structural learning phase is to identify the regions of the I/O space with high data densities. Subsequently, more memory cells (i.e. a finer network output granularity) are assigned to these regions to emulate the experience-driven dendritic aborization phenomenon observed in the cerebellar learning process during skill acquisition. Analogous to the repeated exposures of the learning episodes during skill acquisition, these identified regions of the I/O space contain a large amount of training data points that coexisted in close proximity. For simplicity, the PSECMAC memory allocation and non-uniform

quantization process is performed individually for each input dimension and consists of several steps: (1) the identification of the data density clusters; (2) the allocation of the PSECMAC memory cells based on the computed density profile; and (3) the derivation of the respective PSECMAC quantization decision functions.

1) THE COMPUTATION OF DATA DENSITY CLUSTERS

In the PSECMAC network, significant data clusters supporting the inherent organization of the training dataset are first identified via the Pseudo Self-Evolving Cerebellar (PSEC) [68] clustering algorithm through an analysis of the density distribution of the training data points along each input dimension. The PSEC algorithm is a density-based clustering algorithm which synergizes the merits of the incremental learning procedure of the *Learning Vector Quantization* (LVQ) [69] technique with the effectiveness of the density-based partitioning method of the DBSCAN algorithm [70]. This clustering algorithm is inspired by the biological development of the human brain where neural cell death plays an integral part in the refinement process of its neuronal organization [68]. Neurophysiological studies have established that there are two overlapping stages in the development of the human brain [51]. The first stage of this development process encompasses the formation of the basic architecture of the brain system, in which coarse connection patterns emerge as a result of the genesis of the brain cells during prenatal development. Subsequently, in the second stage of the brain's development, the initial architecture is refined and extraneous synaptic connections are pruned throughout an individual's life-span via exposures to various activity-dependent experiences. These two stages of the human brain adaptation process are functionally emulated by the PSEC clustering algorithm.

The proposed PSECMAC network employs a modified PSEC (MPSEC) clustering algorithm to identify the centers of the density clusters along each input dimension of the training data space. A *density cluster* is defined as a cluster identified from the data density profile computed with the MPSEC algorithm. Each density cluster is associated with a *cluster center*, which denotes the point of highest data density in the cluster. The MPSEC algorithm commences with an initial set of regularly-spaced density clusters, with the mid points of these initial density clusters defined as the respective cluster centers. This initial set of clusters is incrementally *evolved* to capture the data density profile along each input dimension to derive a final set of density clusters. The LVQ iterative algorithm is subsequently employed to refine the positions of the cluster centers in this final set of density clusters.

Let J and L denote the total number of input and output dimensions for a given dataset respectively. Assume that a training dataset of $\mathbf{U} = \left\{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \cdots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \cdots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\right\}$ is

used to train the PSECMAC network, where $\mathbf{X}_s = \begin{bmatrix} x_{s,1} & x_{s,2} & \cdots & x_{s,J} \end{bmatrix}^T$ denotes the s^{th} input training vector, and $\hat{\mathbf{Y}}_s = \begin{bmatrix} \hat{y}_{s,1} & \hat{y}_{s,2} & \cdots & \hat{y}_{s,L} \end{bmatrix}^T$ denotes the corresponding expected output target vector of the PSECMAC network. Let τ denotes the clustering iteration in MPSEC and $\mathbf{C}_j^{(\tau)} = \left\{ C_{j,1}^{(\tau)}, C_{j,2}^{(\tau)}, \cdots, C_{j,n}^{(\tau)}, \cdots, C_{j,n_{C,j}^{(\tau)}}^{(\tau)} \right\}$ denotes the set of density clusters along the j^{th} input dimension at the τ^{th} iteration, where $n_{C,j}^{(\tau)}$ is the the corresponding total number of density clusters. Let $\mathbf{C}_j^{(-1)}$ denotes the initial set of density clusters for the MPSEC algorithm and $n_{C,j}^{(-1)}$ be the number of these regularly-spaced density clusters along the j^{th} input dimension. For each input dimension $j \in \{1 \cdots J\}$, the MPSEC clustering algorithm is briefly outlined as follows:

Step 1 *Initialize the clustering parameters.*

An initial number of density clusters $n_{C,j}^{(-1)}$, along with a pseudo potential threshold β, a clustering termination criterion ε and the LVQ learning constant α_c are determined prior to the start of MPSEC clustering iteration.

Step 2 *Construct the initial set of density clusters.*

The initial set of density clusters $\mathbf{C}_j^{(-1)}$ is subsequently constructed with $n_{C,j}^{(-1)}$ regularly spaced clusters such that $\mathbf{C}_j^{(-1)} = \left\{ C_{j,1}^{(-1)}, C_{j,2}^{(-1)}, \cdots, C_{j,n}^{(-1)}, \cdots, C_{j,n_{C,j}^{(-1)}}^{(-1)} \right\}$. Each density cluster $C_{j,n}^{(\tau)}$ is associated with a cluster center $P_{j,n}^{(\tau)}$ and a density value $V_{j,n}^{(\tau)}$. In the initial set of density clusters $\mathbf{C}_j^{(-1)}$, the cluster center $P_{j,n}^{(-1)}$ is assigned to the center points of the corresponding density cluster $C_{j,n}^{(-1)}$ and the density value $V_{j,n}^{(-1)}$ are initialized to zero. This step emulates the formation of the initial brain system, in which extraneous connection patterns emerge as a result of the overproduction of neurons during the prenatal brain development phase.

Step 3 *Compute the initial cluster density values.*

MPSEC performs structural learning by executing a one-pass learning of the density values $V_{j,n}^{(-1)}$ to obtain a density distribution of the training data along the j^{th} input dimension.

Step 4 *Evolve the initial set of density cluster.*

For each input dimension, the initial set of density clusters $\mathbf{C}_j^{(-1)}$ is evolved to capture the inherent data density profile by identifying all the local maxima in the set of computed density values $V_{j,n}^{(-1)}$. This step emulates the competitive neuronal selection process in human brain development, whereby neurons with high tropic factors are identified as the winner neurons and the remaining extraneous neurons are pruned to create a more refined structure of synaptic connections. The clusters in the initial set of density clusters $\mathbf{C}_j^{(-1)}$ whose density values form prominent convex density peaks in the computed density distribution of Step 3 are included

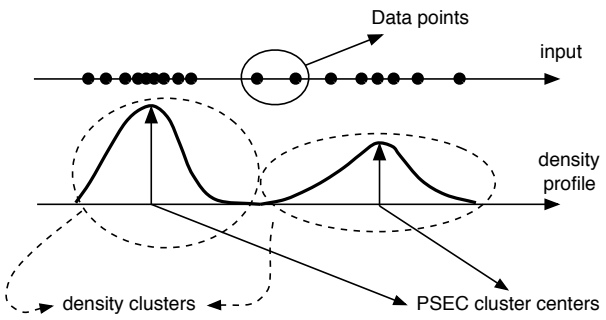


Fig. 3. A sample output of the MPSEC clustering technique

in the new set of density clusters $\mathbf{C}_j^{(0)}$. The rest of the density clusters are removed (pruned). The remaining clusters in the new set of density clusters $\mathbf{C}_j^{(0)}$ are therefore analogous to the surviving neurons with high tropic factors in the brain neuronal selection process.

Step 5 *Incremental learning of cluster centers.*

The cluster centers $P_{j,n}^{(0)}$ of the new set of density clusters $\mathbf{C}_j^{(0)}$ are subsequently refined to derive the accurate positioning of the density-induced cluster centers. The incremental learning of cluster centers are performed iteratively using the LVQ algorithm, resulting on the final set of density clusters $\mathbf{C}_j^{(\tau)}$.

Step 6 *Compute the resultant density profile.*

A one-pass learning of the density values $V_{j,n}^{(\tau)}$ in the final set of density clusters $\mathbf{C}_j^{(\tau)}$ is performed to derive the density values at the final cluster centers $P_{j,n}^{(\tau)}$. Finally, for each density cluster $C_{j,n}^{(\tau)}$ in $\mathbf{C}_j^{(\tau)}$, the left and right boundary $L_{j,n}^{(\tau)}$ and $R_{j,n}^{(\tau)}$ are computed as the center point between the cluster center $P_{j,n}^{(\tau)}$ and the cluster center of its corresponding left and right neighbors.

Figure 3 illustrates the mechanism of the MPSEC clustering algorithm. Essentially, MPSEC computes a set of density-induced clusters, whose centers denote the highest density points in the respective clusters. The boundary between any two neighboring clusters are conveniently assumed to be at the bisection of the two respective cluster centers. The detailed descriptions and mathematical formulations of the MPSEC algorithm is listed in [71].

2) THE PSECMAC MEMORY ALLOCATION PROCESS

In the proposed PSECMAC network, the number of memory cells allocated to a density cluster is proportional to the normalized density value of the corresponding cluster center. Let $\hat{M}_j$ denotes the

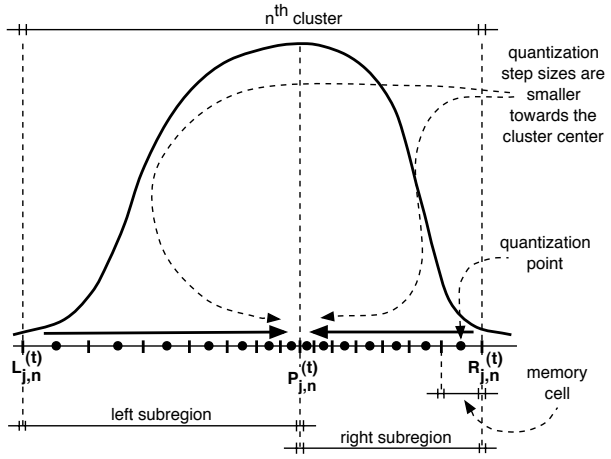


Fig. 4. Adaptive memory cell distribution in cluster $C_{j,n}^{(\tau)}$

total number of predefined memory cells in the j^{th} input dimension. Then for each density cluster $C_{j,n}^{(\tau)}$ in $\mathbf{C}_j^{(\tau)}$, the number of memory cells $M_{j,n}$ allocated for this cluster is computed as:

$$M_{j,n} \quad = \quad \left\lfloor \frac{V_{j,n}^{(\tau)}}{\sum_{n' \in \{1 \cdots n_{C,j}^{(\tau)}\}} V_{j,n'}^{(\tau)}} \right\rfloor \times \hat{M}_j \tag{3}$$

where $M_{j,n}$ is the number of memory cells allocated to cluster $C_{j,n}^{(\tau)}$ in the j^{th} input dimension, $V_{j,n}^{(\tau)}$ is the density value of $C_{j,n}^{(\tau)}$, and $n_{C,j}^{(\tau)}$ denotes the total number of computed density clusters in the j^{th} input dimension.

3) THE PSECMAC QUANTIZATION DECISION FUNCTIONS

For the proposed PSECMAC network, a non-linear assignment scheme is introduced for the computation of the quantization decision functions to vary the quantization step sizes of the memory cells of the identified density clusters. In PSECMAC, the memory cells allocated to an arbitrary cluster $C_{j,n}^{(\tau)}$ is equally distributed to the left and right side of the cluster center (i.e. the left and right subregions). In each of the subregions, the quantization point of each memory cell is logarithmically assigned with respect to the cluster center. The quantization point of a memory cell is defined as the midpoint of the memory cell. The result of this computation is illustrated in Figure 4, which depicts the adaptively quantized memory cells inside a cluster. Computationally, the center of each density-induced cluster constitutes the finest data granularity. As a memory cell moves away from the cluster center, its quantization step size increases in response to a lower density of the observed training data.

In this work, a logarithmic quantization technique (commonly referred to as μ-law quantization [72]) is employed to manage the distribution of the memory cells in a cluster. The degree of non-linearity in the quantization step sizes of the memory cells is governed by a parameter μ. Subsequently, a quantization mapping function $\mathbf{Q}_j[\cdot] \rightarrow \left\{Q_{j,1}, Q_{j,2}, \cdots, Q_{j,\hat{M}_j}\right\}$ is constructed to define the quantization of the memory cells in the j^{th} input dimension of the PSECMAC network, where $Q_{j,\bar{n}}$, $\bar{n} \in \{1 \cdots \hat{M}_j\}$, denotes the $\bar{n}^{th}$ quantization point. The derivation of the quantization function $\mathbf{Q}_j[\cdot]$ is described as follows:

(a) Initialize $\bar{n} = 1$ (i.e. first quantization point) and define the parameter μ for the non-linear distribution of the memory cells.

(b) For $n = 1 \cdots n_{C,j}^{(\tau)}$: Let $M_{j,n}$ (computed from Equation (3)) denote the number of memory cells allocated to a density cluster $C_{j,n}^{(\tau)}$ in the j^{th} input dimension and k be the index to the memory cells in the density cluster $C_{j,n}^{(\tau)}$. For $k = 1 \cdots M_{j,n}$, compute the quantization point for the k^{th} memory cell in $C_{j,n}^{(\tau)}$ such that:

- IF the k^{th} memory cell is in the left subregion (i.e. $k \leq \left\lfloor \frac{M_{j,n}}{2} \right\rfloor$) THEN:

$$step = \frac{P_{j,n}^{(\tau)} - L_{j,n}^{(\tau)}}{\left\lfloor \frac{M_{j,n}}{2} \right\rfloor} \tag{4}$$

$$pt = L_{j,n}^{(\tau)} + (k - 0.5) \cdot step \tag{5}$$

$$Q_{j,\bar{n}} = L_{j,n}^{(\tau)} + \left\lceil \frac{\left(P_{j,n}^{(\tau)} - L_{j,n}^{(\tau)}\right) \cdot \log\left(1 + \frac{\mu \cdot |L_{j,n}^{(\tau)} - pt|}{(P_{j,n}^{(\tau)} - L_{j,n}^{(\tau)})}\right)}{\log(1+\mu)} \right\rceil \tag{6}$$

  where $P_{j,n}^{(\tau)}$ and $L_{j,n}^{(\tau)}$ are the center and the left boundary of the density cluster $C_{j.n}^{(\tau)}$ respectively. Update the index $\bar{n} = \bar{n} + 1$.

- ELSE IF $M_{j,n}$ is odd and the k^{th} memory cell is assigned to the cluster center (i.e. $\left\lfloor \frac{M_{j,n}}{2} \right\rfloor < k < \left\lceil \frac{M_{j,n}}{2} \right\rceil + 1$) THEN:

$$Q_{j,\bar{n}} = P_{j,n}^{(\tau)} \tag{7}$$

  Update the index of the current decision point $\bar{n} = \bar{n} + 1$.

- ELSE IF the k^{th} memory cell is in the right subregion (i.e. $k > \left\lfloor \frac{M_{j,n}}{2} \right\rfloor$) THEN:

$$step = \frac{R_{j,n}^{(\tau)} - P_{j,n}^{(\tau)}}{\left\lfloor \frac{M_{j,n}}{2} \right\rfloor} \tag{8}$$

$$pt = P_{j,n}^{(\tau)} + (k - \left\lfloor \frac{M_{j,n}}{2} \right\rfloor - 0.5) \cdot step \quad (9)$$

$$Q_{j,\bar{n}} = R_{j,n}^{(\tau)} - \left\lceil \frac{\left(R_{j,n}^{(\tau)} - P_{j,n}^{(\tau)}\right) \cdot \log\left(1 + \frac{\mu \cdot |pt - R_{j,n}^{(\tau)}|}{(R_{j,n}^{(\tau)} - P_{j,n}^{(\tau)})}\right)}{\log(1+\mu)} \right\rceil \quad (10)$$

where $R_{j,n}^{(\tau)}$ is the right boundary of the density cluster $C_{j.n}^{(\tau)}$. Update the index $\bar{n} = \bar{n} + 1$.

Note that the second condition (Equation (7)) is met only when $M_{j,n}$ is odd. Otherwise, the number of allocated memory cells to the left and right subregions of cluster $C_{j,n}^{(\tau)}$ are evenly allocated and is equal to $\left\lfloor \frac{M_{j,n}}{2} \right\rfloor$.

After the completion of this placement process, the quantization mapping function $\mathbf{Q}_j[\cdot]$ is defined for the j^{th} input dimension. The computed quantization decision points of each input dimension subsequently form the memory axes of the proposed PSECMAC network and are used to define its overall computing structure. The intersections of these memory axes denote the computing cells of the PSECMAC network and define the I/O associative space. The training of this PSECMAC computing structure is described in the following two sections.

B. The PSECMAC Computational Process

The PSECMAC network employs a *Weighted Gaussian Neighborhood Output* (WGNO) computation process, where a set of neighborhood-bounded computing cells is activated to derive the network's output response to the given input stimulus. In this computation process, each of the neighborhood cells has a weighted degree of activation that is inversely proportional to the distance of the cell from the input stimulus point. The objective of the WGNO scheme is to minimize the influences of the input quantization errors on the computed network output. In addition, it introduces a "smoothing" effect on the PSECMAC output and enhances the generalization capability of the PSECMAC network.

Let $\mathbf{Y}_s$ denotes the computed PSECMAC network output for an input stimulus $\mathbf{X}_s = [x_{s,1}, x_{s,2}, \cdots x_{s,J}]$ to the PSECMAC network. The WGNO computation process is defined as follows:

Step 1: *Determining the region of activation.*

The size of the activated PSECMAC neighborhood with respect to input $\mathbf{X}_s$ is defined by $N \in [0 \cdots 1]$, a user-specified parameter that governs the relative size of the neighborhood of activated PSECMAC cells to the overall memory space. The neighborhood activation boundaries are defined on per-dimension basis such that $N = 0.2$ denotes an activation boundary of 20% relative to the range of the respective input dimension. A neighborhood constant of $N = 0.2$ therefore signifies

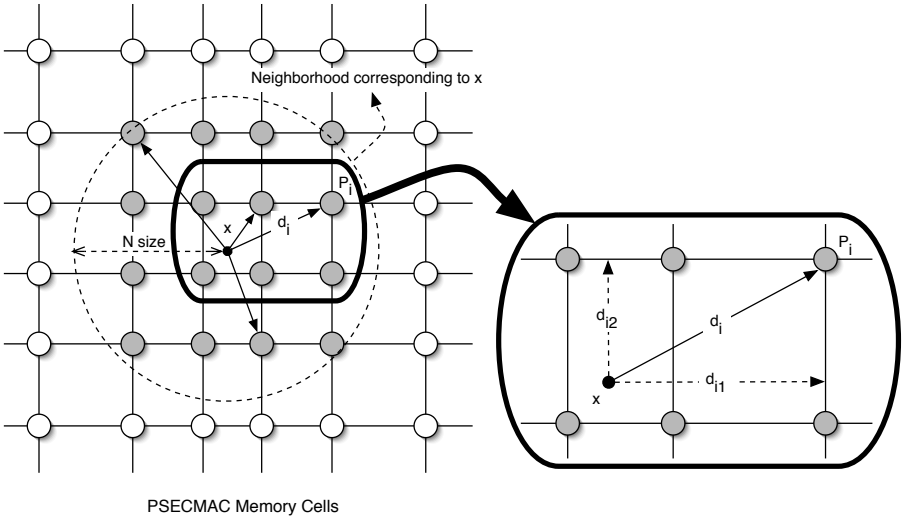


Fig. 5. An example of a 2D PSECMAC neighborhood

an neighborhood activation of $(0.2)^J \times 100\%$ relative to the entire input space, where J denotes the total number of input dimension. For the input stimulus $\mathbf{X}_s$, its activation neighborhood is defined as:

$$lb_{s,j} = x_{s,j} - 0.5 \cdot N \cdot range_j \tag{11}$$

$$rb_{s,j} = x_{s,j} + 0.5 \cdot N \cdot range_j \tag{12}$$

$$j \in \{1, 2, \cdots, J\}$$

where $lb_{s,j}$ denotes the left activation boundary, $rb_{s,j}$ denotes the right activation boundary, and $range_j$ is the domain for the j^{th} input dimension. Subsequently, the memory cells encapsulated within the neighborhood defined by the computed boundaries are activated in response to the input stimulus $\mathbf{X}_s$. A PSECMAC activation neighborhood is illustrated as Figure 5. The size of the neighborhood affects the accuracy of the computed PSECMAC output. The larger the neighborhood size, the more generalized is the output of the PSECMAC network. Conversely, a smaller neighborhood size results in a more accurate output computation. Therefore, a larger neighborhood size is suitable for a dataset that is sparse in the input space as this increases the generalization ability of the PSECMAC network. A smaller neighborhood size, on the other hand, is suitable for a compact dataset so as to produce more accurate results.

Step 2: *Computing the Gaussian weighting function.*

A Gaussian weighting factor g_k is associated with each activated PSECMAC cell to determine its contribution towards the computation of the network output. The Gaussian weighting factor is defined as:

$$g_k = (1 - d_k)e^{-d_k^2/2\gamma^2} \tag{13}$$

where γ is the Gaussian width constant and d_k denotes the normalized Euclidean distance from the k^{th} activated cell to the input stimulus $\mathbf{X}_s$ (see Figure 5). Let $\mathbf{K}_s$ be the set of activated PSECMAC cells in the computed neighborhood. Subsequently, d_k is defined as

$$d_k = \frac{\|\mathbf{Q}_k - \mathbf{X}_s\|}{\max_{k' \in \mathbf{K}_s} \|\mathbf{Q}_{k'} - \mathbf{X}_s\|} \tag{14}$$

where $\mathbf{Q}_k = [Q_{1,k}, Q_{2,k}, \cdots, Q_{J,k}]$ denotes the quantization point of cell k in the memory space.

Step 3: *Retrieving the PSECMAC Output.*

The PSECMAC network output $\mathbf{Y}_s$ is computed as a weighted linear combination of the memory contents of the activated cells such that:

$$\mathbf{Y}_s = \frac{\sum_{k \in \mathbf{K}_s} (g_\mathbf{k} \cdot \mathbf{W}(k))}{\sum_{k \in \mathbf{K}_s} g_k} \tag{15}$$

where $\mathbf{K}_s$ denotes the set of neighborhood-activated PSECMAC cells, and $\mathbf{W}(k)$ is the stored weight value(s) of the activated PSECMAC cell with index k.

C. The PSECMAC Learning Process

This section describes the parameter tuning phase of the proposed PSECMAC network. Parameter tuning is performed for the PSECMAC network to learn the mapping of the input-output associative patterns from the training data tuples. To emulate the neighborhood learning phenomenon of the human cerebellum [51], [54], the PSECMAC network adopts a *Weighted Gaussian Neighborhood Update* (WGNU) process. WGNU combines the Widrow-Hoff training algorithm [73] with the Gaussian weighting function defined in Equation (13). The objective of this neighborhood update scheme is to distribute the effect of learning to increase the generalization capability of the PSECMAC network.

For an arbitrary input-output training data tuple $(\mathbf{X}_s, \hat{\mathbf{Y}}_s)$, the PSECMAC learning process is mathematically described as follows:

1) Compute the PSECMAC output $\mathbf{Y}_s^{(i)}$ at the i^{th} training iteration:

$$\mathbf{Y}_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} \left(g_k \cdot \mathbf{W}^{(i)}(k)\right)}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \tag{16}$$

where $\mathbf{K}_s$ is the set of activated computing cells corresponding to the input $\mathbf{X}_s$, g_k is the Gaussian weighting factor of the k^{th} activated memory (computing) cell, $\mathbf{W}^{(i)}(k)$ denotes the memory content of the k^{th} activated memory cell at the i^{th} iteration, and $\mathbf{Y}_s^{(i)}$ is the output of the PSECMAC network to the input $\mathbf{X}_s$ at the i^{th} iteration.

2) Compute the network output error at the i^{th} iteration:

$$\mathbf{Err}_s^{(i)} = \hat{\mathbf{Y}}_s - \mathbf{Y}_s^{(i)} \tag{17}$$

where $\mathbf{Err}_s^{(i)}$ denotes the output error of the PSECMAC network to the input $\mathbf{X}_s$ at the i^{th} iteration, and $\hat{\mathbf{Y}}_s$ is the desired (target) output of the PSECMAC network in response to the input $\mathbf{X}_s$.

3) Update the stored network weights:

$$\mathbf{W}^{(i+1)}(k) = \mathbf{W}^{(i)}(k) + \mathbf{\Delta W}^{(i)}(k) \quad k \in \mathbf{K}_s \tag{18}$$

$$\mathbf{\Delta W}^{(i)}(k) = \alpha \underbrace{\frac{g_k}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \mathbf{Err}_s^{(i)}}_{\text{local error for cell } k} \quad k \in \mathbf{K}_s \tag{19}$$

where α is the learning constant, and $\mathbf{W}^{(i)}(k)$ denotes the content (weight) of the k^{th} activated cell in the neighborhood $\mathbf{K}_s$ in PSECMAC in response to the input stimulus $\mathbf{X}_s$ at the i^{th} training iteration.

The PSECMAC memory learning phase commences with the computation of the network output corresponding to the input stimulus $\mathbf{X}_s$. A learning error is computed based on the derived PSECMAC output and the target response. This error is subsequently distributed to all the activated computing (memory) cells based on the Gaussian weighting factors. The local errors are then used to update the memory contents of the activated cells. As a computational model of the human cerebellum, the PSECMAC network exhibits properties that are highly similar to the neurobiological and neurophysiological aspects of its biological counterpart. The correspondence between the characteristics of the human cerebellum and the proposed PSECMAC network are listed in [74].

V. The PSECMAC Learning Convergence

This section presents the mathematical proof of the learning convergence of the proposed PSECMAC network. This proof is modeled closely after the theoretical proof of convergence of the basic CMAC network presented in [21]. Figure 6 depicts an example of the memory surface $\mathbf{Z}$ of a 2-input PSECMAC network. $\mathbf{Z}(\mathbf{q1}, \mathbf{q2})$ denotes the network cell with the address index $(\mathbf{q1}, \mathbf{q2})$. With respect to Figure 6, the

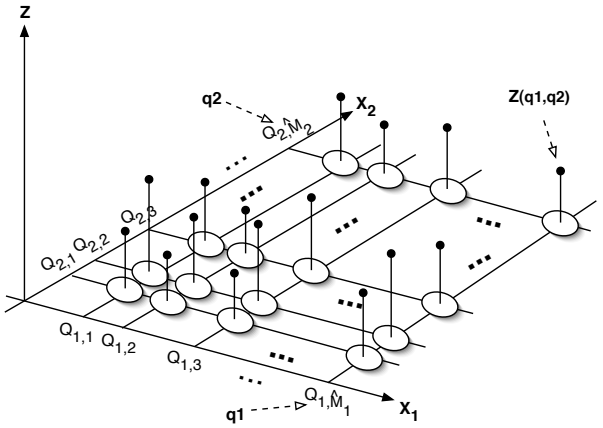


Fig. 6. An example of the 2-input PSECMAC Memory Content

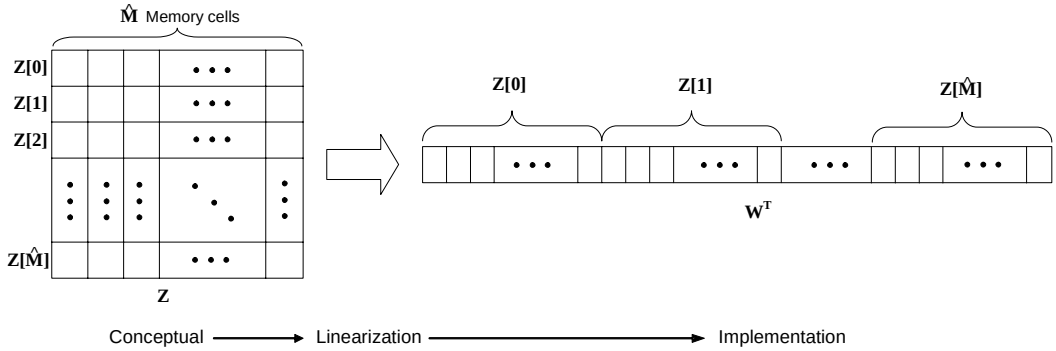


Fig. 7. 2D PSECMAC $\mathbf{Y} \rightarrow \mathbf{W}$ mapping

quantization points along the X_1 dimension are $\{Q_{1,1}, Q_{1,2}, Q_{1,3}, \cdots Q_{1,\hat{M}_1}\}$ and along the X_2 dimension are $\{Q_{2,1}, Q_{2,2}, Q_{2,3}, \cdots Q_{2,\hat{M}_2}\}$ respectively. However, for simplicity, equal memory size per dimension is assumed in the proof of convergence, i.e. $\hat{M}_1 = \hat{M}_2 = \cdots = \hat{M}$, where $\hat{M}$ is the memory size per dimension.

A. Mathematical Perspective of The PSECMAC Network

The PSECMAC network employs the WGNO and WGNU computations for the network retrieval and update operations respectively and the activated neighborhoods of the two processes are centered at the corresponding multi-dimensional input vector to the network (See Figure 5). The conceptual memory surface $\mathbf{Z}$ of a multi-input PSECMAC network can be expressed as a one-dimensional weight array $\mathbf{W}$. Figure 7 illustrates the *linearization* of the conceptual memory surface $\mathbf{Z}$ to the physically implemented one-dimensional weight array $\mathbf{W}$ for a two-dimensional PSECMAC example. With respect to the PSEC-

MAC network, the computed output for the s^{th} input sample at the i^{th} learning iteration is defined as:

$$\mathbf{Y}_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} \left(g_k \cdot \mathbf{W}^{(i)}(k) \right)}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \tag{20}$$

where $\mathbf{Y}_s^{(i)}$ is the computed output of the multi-dimensional PSECMAC to the s^{th} input training vector at the i^{th} iteration, $\mathbf{K}_s$ is the set of activated computing cells corresponding to the s^{th} input training vector, g_k is the Gaussian weighting factor of the k^{th} activated memory (computing) cell, and $\mathbf{W}^{(i)}(k)$ denotes the content of the k^{th} activated memory cell at the i^{th} iteration.

For simplicity, only scalar output is considered here. That is, the PSECMAC network presented here for the proof of learning convergence is a multi-input single-output system. However, this proof can be easily extended to a multi-input multi-output PSECMAC system without any loss of generality. The computed output of the single output PSECMAC system is expressed as:

$$y_s^{(i)} = \frac{\sum_{k \in \mathbf{K}_s} \left(g_k \cdot \mathbf{W}^{(i)}(k) \right)}{\sum_{k' \in \mathbf{K}_s} g_{k'}} \tag{21}$$

where $y_s^{(i)}$ denotes the computed output of the single-output PSECMAC network with respect to the s^{th} input training sample at the i^{th} training iteration.

Assume a training dataset $\mathbf{U}$ of S tuples, i.e. $\mathbf{U} = \{(\mathbf{X}_1, \hat{\mathbf{Y}}_1), (\mathbf{X}_2, \hat{\mathbf{Y}}_2), \cdots, (\mathbf{X}_s, \hat{\mathbf{Y}}_s), \cdots, (\mathbf{X}_S, \hat{\mathbf{Y}}_S)\}$. Let the total number of memory cells in the PSECMAC network be $\hat{M}^J$ where J is the total number of input dimensions and the column vector $\mathbf{A}_s$ denotes the activation mask of the PSECMAC memory cells with respect to the s^{th} input training sample. That is,

$$\mathbf{A}_s^T = \underbrace{[a_{s,1} \quad a_{s,2} \quad \cdots \quad a_{s,\hat{M}^J}]}_{1 \times \hat{M}^J \text{ array}} \tag{22}$$

$$a_{s,j} = \frac{g_j}{\sum_{k \in \mathbf{K}_s} g_k}, \qquad j \in \{1 \cdots \hat{M}^J\} \tag{23}$$

$$g_j = 0, \qquad \text{if } j \notin \mathbf{K}_s \tag{24}$$

where $\mathbf{K}_s$ denotes the set of activated memory cells in the neighborhood selected by the s^{th} input training sample. Hence, the mask $\mathbf{A}_s^T$ identifies the activated memory cells corresponding to the neighborhood selected by the s^{th} input training sample and subsequently weights these cells using the Gaussian neighborhood function defined in Equation (13). The scalar output of the PSECMAC network can thus

be formulated as a vector multiplication such that

$$y_s = \underbrace{[a_{s,1} \quad a_{s,2} \quad \cdots \quad a_{s,\hat{M}^J}]}_{1\times\hat{M}^J \text{ array}} \mathbf{W}_s = \mathbf{A}_s^T \mathbf{W}_s \tag{25}$$

where $\mathbf{W}_s$ is the memory content of the entire PSECMAC network structure at the time when the s^{th} input training sample is presented.

The memory update operation of the PSECMAC network for the s^{th} input training sample is subsequently defined as:

$$\begin{aligned} \mathbf{W}_{s+1}^{(i)} &= \mathbf{W}_s^{(i)} + \underbrace{\Delta\mathbf{W}_s^{(i)}}_{\alpha\times\text{local error}} \\ &= \mathbf{W}_s^{(i)} + \alpha \underbrace{\mathbf{A}_s \overbrace{\{\hat{y}_s - \mathbf{A}_s^T \mathbf{W}_s^{(i)}\}}^{\text{learning error}}}_{\text{local error}} \end{aligned} \tag{26}$$

where $\mathbf{W}_{s+1}^{(i)}$ denotes the memory content of the entire PSECMAC network structure when the $(s+1)^{th}$ training sample is presented in the i^{th} training iteration, α is the learning constant, $\mathbf{A}_s$ is the activation mask of the PSECMAC memory cells, and $\hat{y}_s$ denotes the desired (expected) PSECMAC output for the s^{th} training sample.

The difference of the PSECMAC memory contents between two successive iterations for the s^{th} input training sample (denoted as $\mathbf{Dw}_s^{(i)}$) is therefore defined as:

$$\begin{aligned} \mathbf{Dw}_s^{(i)} &= \mathbf{W}_s^{(i+1)} - \mathbf{W}_s^{(i)} \\ &= \underbrace{\mathbf{W}_{s-1}^{(i+1)} + \Delta\mathbf{W}_{s-1}^{(i+1)}}_{\mathbf{W}_s^{(i+1)}} - \underbrace{(\mathbf{W}_{s-1}^{(i)} + \Delta\mathbf{W}_{s-1}^{(i)})}_{\mathbf{W}_s^{(i)}} \\ &= \underbrace{\mathbf{Dw}_{s-1}^{(i)}}_{\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)}} + \underbrace{\alpha\mathbf{A}_{s-1}\{\hat{y}_{s-1} - \mathbf{A}_{s-1}^T \mathbf{W}_{s-1}^{(i+1)}\}}_{\Delta\mathbf{W}_{s-1}^{(i+1)}} - \underbrace{\alpha\mathbf{A}_{s-1}\{\hat{y}_{s-1} - \mathbf{A}_{s-1}^T \mathbf{W}_{s-1}^{(i)}\}}_{\Delta\mathbf{W}_{s-1}^{(i)}} \\ &= \mathbf{Dw}_{s-1}^{(i)} - \alpha\mathbf{A}_{s-1}\mathbf{A}_{s-1}^T \underbrace{(\mathbf{W}_{s-1}^{(i+1)} - \mathbf{W}_{s-1}^{(i)})}_{\mathbf{Dw}_{s-1}^{(i)}} \\ &= (\mathbf{I} - \alpha \underbrace{\mathbf{A}_{s-1}\mathbf{A}_{s-1}^T}_{\text{outer product}})\mathbf{Dw}_{s-1}^{(i)} \end{aligned} \tag{27}$$

Note that the activation mask $\mathbf{A}_s$ is constant for an arbitrary given input training sample s across different training iterations. This is because the PSECMAC network structure is fixed after the structural learning phase.

Following Equation (27), let

$$\mathbf{E}_s \equiv (\mathbf{I} - \alpha \mathbf{A}_s \mathbf{A}_s^T) \quad \text{where } \mathbf{E}_s \text{ is } \hat{M}^J \times \hat{M}^J \text{ matrix} \tag{28}$$

and

$$\mathbf{Dw}^{(i)} \equiv \left[\begin{array}{cccc} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{array} \right] \tag{29}$$

where S denotes the total number of input training samples.

The learning convergence of the PSECMAC network is established via the convergence of the network memory contents as training approaches infinity [21], [75]. In this case, the sufficient and necessary condition for the PSECMAC learning process to convergence can be expressed as:

$$\lim_{i \to \infty} \mathbf{Dw}_s^{(i)} = [0], \quad \forall s \in \{1 \cdots S\}, \quad \text{or in matrix notation} \tag{30}$$

$$\lim_{i \to \infty} \mathbf{Dw}^{(i)} = [0]_{\hat{M}^J \times S} \tag{31}$$

where $[0]$ is the null matrix.

Substituting Equation (28) into Equation (27), one obtains

$$\mathbf{Dw}_s^{(i)} = \mathbf{E}_{s-1} \mathbf{Dw}_{s-1}^{(i)} \tag{32}$$

The PSECMAC network is trained iteratively on a set of S training samples. When $s = 1$, from Equation (32)

$$\mathbf{Dw}_1^{(i)} = \mathbf{E}_0 \mathbf{Dw}_0^{(i)} \tag{33}$$

such that

$$\mathbf{Dw}_0^{(i)} = \mathbf{Dw}_S^{(i-1)} \tag{34}$$

(i.e. by recycling the training samples in their existing order for each new training iteration). Following the definition of Equation (34),

$$\mathbf{E}_0 = \mathbf{E}_S \tag{35}$$

$$\Rightarrow \quad \mathbf{A}_0 = \mathbf{A}_S \text{ (From Equation (28))}$$

From the results of Equations (32 – 35), $\mathbf{Dw}^{(i)}$ (See Equation (29)) can be expanded and re-expressed

as

$$\begin{aligned}
\mathbf{Dw}^{(i)} &= \left[\begin{array}{cccc} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{array} \right] \\
&= \left[\begin{array}{cccc} \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \underbrace{\mathbf{E}_1 \mathbf{Dw}_1^{(i)}}_{\mathbf{Dw}_2^{(i)}} & \cdots & \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i)}}_{\mathbf{Dw}_S^{(i)}} \end{array} \right] \\
&= \left[\begin{array}{cccc} \mathbf{E}_S \underbrace{\mathbf{E}_{S-1} \mathbf{Dw}_{S-1}^{(i-1)}}_{\mathbf{Dw}_S^{(i-1)}} & \mathbf{E}_1 \underbrace{\mathbf{E}_S \mathbf{Dw}_S^{(i-1)}}_{\mathbf{Dw}_1^{(i)}} & \cdots & \mathbf{E}_{S-1} \underbrace{\mathbf{E}_{S-2} \mathbf{Dw}_{S-2}^{(i)}}_{\mathbf{Dw}_{S-1}^{(i)}} \end{array} \right]
\end{aligned} \tag{36}$$

Decompose the $\mathbf{Dw}$ terms on the right hand side repeatedly to obtain the following:

$$\begin{aligned}
\mathbf{Dw}^{(i)} &= \left[\begin{array}{ccc} (\mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_1) \mathbf{Dw}_1^{(i-1)} & (\mathbf{E}_1 \mathbf{E}_S \cdots \mathbf{E}_2) \mathbf{Dw}_2^{(i-1)} & \cdots \\ & (\mathbf{E}_{S-1} \mathbf{E}_{S-2} \cdots \mathbf{E}_S) \mathbf{Dw}_S^{(i-1)} \end{array} \right]
\end{aligned} \tag{37}$$

Following Equation (37), let

$$\mathbf{G}_s \equiv \mathbf{E}_{s-1} \mathbf{E}_{s-2} \cdots \mathbf{E}_1 \mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_s, \quad s \in \{1 \cdots S\} \tag{38}$$

where $\mathbf{G}_s$ is a matrix multiplication of S terms. With the definition of $\mathbf{G}_s$, Equation (37) can be re-expressed as

$$\begin{aligned}
\mathbf{Dw}^{(i)} &= \left[\begin{array}{cccc} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{array} \right] \\
&= \left[\begin{array}{ccc} \underbrace{(\mathbf{E}_S \mathbf{E}_{S-1} \cdots \mathbf{E}_1)}_{\mathbf{G}_1} \mathbf{Dw}_1^{(i-1)} & \underbrace{(\mathbf{E}_1 \mathbf{E}_S \cdots \mathbf{E}_2)}_{\mathbf{G}_2} \mathbf{Dw}_2^{(i-1)} & \cdots \\ & \underbrace{(\mathbf{E}_{S-1} \mathbf{E}_{S-2} \cdots \mathbf{E}_S)}_{\mathbf{G}_S} \mathbf{Dw}_S^{(i-1)} \end{array} \right] \\
&= \left[\begin{array}{cccc} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \cdots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{array} \right]
\end{aligned} \tag{39}$$

It can be observed that

$$\mathbf{Dw}_s^{(i)} = \mathbf{G}_s \mathbf{Dw}_s^{(i-1)} \tag{40}$$

Consequently, it follows that

$$\begin{aligned}
\mathbf{Dw}^{(i)} &= \left[\begin{array}{cccc} \mathbf{Dw}_1^{(i)} & \mathbf{Dw}_2^{(i)} & \cdots & \mathbf{Dw}_S^{(i)} \end{array} \right] \\
&= \left[\begin{array}{cccc} \mathbf{G}_1 \mathbf{Dw}_1^{(i-1)} & \mathbf{G}_2 \mathbf{Dw}_2^{(i-1)} & \cdots & \mathbf{G}_S \mathbf{Dw}_S^{(i-1)} \end{array} \right]
\end{aligned}$$

$$= \left[\begin{array}{cccc} \mathbf{G}_1 \underbrace{\mathbf{G}_1 \mathbf{Dw}_1^{(i-2)}}_{\mathbf{Dw}_1^{(i-1)}} & \mathbf{G}_2 \underbrace{\mathbf{G}_2 \mathbf{Dw}_2^{(i-2)}}_{\mathbf{Dw}_2^{(i-1)}} & \cdots & \mathbf{G}_S \underbrace{\mathbf{G}_S \mathbf{Dw}_S^{(i-2)}}_{\mathbf{Dw}_S^{(i-1)}} \end{array} \right]$$

$$= \left[\begin{array}{cccc} (\mathbf{G}_1)^2 \mathbf{Dw}_1^{(i-2)} & (\mathbf{G}_2)^2 \mathbf{Dw}_2^{(i-2)} & \cdots & (\mathbf{G}_S)^2 \mathbf{Dw}_S^{(i-2)} \end{array} \right] \quad (41)$$

Decompose the $\mathbf{Dw}$ terms on the right hand side repeatedly to obtain the following:

$$\mathbf{Dw}^{(i)} = \left[\begin{array}{cccc} (\mathbf{G}_1)^i \mathbf{Dw}_1^{(0)} & (\mathbf{G}_2)^i \mathbf{Dw}_2^{(0)} & \cdots & (\mathbf{G}_S)^i \mathbf{Dw}_S^{(0)} \end{array} \right] \quad (42)$$

where $\mathbf{Dw}_s^{(0)}$ denotes the change in the memory contents of the PSECMAC network for the first training iteration when the s^{th} training sample is presented. With respect to Equation (42), the memory difference matrix $\mathbf{Dw}^{(i)}$ must approach a null matrix as training tends to infinity (i.e. $i \rightarrow \infty$) in order to establish the learning convergence of the proposed PSECMAC network. Hence, the PSECMAC network learning process converges if and only if

$$(\mathbf{G}_s)^i \mathbf{Dw}_s^{(0)} = [0], \quad \forall s \in \{1 \cdots S\} \quad (43)$$

By definition, the difference vector $\mathbf{Dw_s}^{(0)}$ can be expressed as

$$\begin{aligned} \mathbf{Dw}_s^{(0)} &= \mathbf{W}_s^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_{s-1}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)}}_{\mathbf{W}_s^{(1)}} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-2}^{(1)}}_{\mathbf{W}_{s-1}^{(1)}} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \end{aligned} \quad (44)$$

Decompose the $\mathbf{W}_s^{(i)}$ terms on the right hand side repeatedly to obtain:

$$\begin{aligned} \mathbf{Dw}_s^{(0)} &= \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_1^{(1)} + \Delta \mathbf{W}_2^{(1)} + \cdots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_S^{(0)} + \Delta \mathbf{W}_S^{(0)}}_{\mathbf{W}_1^{(1)}} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-2}^{(1)} + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \underbrace{\mathbf{W}_s^{(0)} + \Delta \mathbf{W}_s^{(0)}}_{\mathbf{W}_{s+1}^{(0)}} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} - \mathbf{W}_s^{(0)} \\ &= \Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)} \end{aligned} \quad (45)$$

From Equation (26), the PSECMAC memory update due to the s^{th} input training sample at the i^{th}

iteration ($\Delta \mathbf{W}_s^{(i)}$) is computed as:

$$\Delta \mathbf{W}_s^{(i)} = \alpha \underbrace{\mathbf{A}_s \overbrace{\{\hat{y_s} - \mathbf{A}_s^T \mathbf{W}_s^{(i)}\}}^{\text{learning error}}}_{\text{local error}} \tag{46}$$

where $\{\hat{y_s} - \mathbf{A}_s^T \mathbf{W}_s^{(i)}\}$ is a scalar value and is the learning (training) error for the s^{th} input training sample at the i^{th} iteration. Let $u_s^{(i)} = (\hat{y_s} - \mathbf{A}_s^T \mathbf{W}_s^{(i)})$ and it follows from Equation (46) that

$$\Delta \mathbf{W}_s^{(i)} = \alpha \mathbf{A}_s u_s^{(i)} \tag{47}$$

From Equations (42), (45) and (47), the following results.

$$\begin{aligned}
(\mathbf{G}_s)^i \mathbf{D}\mathbf{w}_s^{(0)} &= (\mathbf{G}_s)^i \{\Delta \mathbf{W}_s^{(0)} + \Delta \mathbf{W}_{s+1}^{(0)} + \cdots + \Delta \mathbf{W}_S^{(0)} + \Delta \mathbf{W}_1^{(1)} + \cdots + \Delta \mathbf{W}_{s-1}^{(1)}\} \\
&= (\mathbf{G}_s)^i \{\alpha \mathbf{A}_s u_s^{(0)} + \alpha \mathbf{A}_{s+1} u_{s+1}^{(0)} + \cdots + \alpha \mathbf{A}_{s-1} u_{s-1}^{(1)}\} \\
&= \alpha (\mathbf{G}_s)^i \{\mathbf{A}_s u_s^{(0)} + \mathbf{A}_{s+1} u_{s+1}^{(0)} + \cdots + \mathbf{A}_{s-1} u_{s-1}^{(1)}\}
\end{aligned} \tag{48}$$

B. Learning Convergence of The PSECMAC Network

Therefore, if it can be shown that $\lim_{i \to \infty} (\mathbf{G}_s)^i \mathbf{A}_\upsilon = [0]$ for all $\upsilon \in \{1 \cdots S\}$, $(\mathbf{G}_s)^i \mathbf{D}\mathbf{w}_s^{(0)}$ in Equation (48) will evaluate as null. Subsequently, from Equation (42), the matrix $\mathbf{D}\mathbf{w}^{(i)} = [0]$ follows. Based on the definition of $\mathbf{D}\mathbf{w}^{(i)}$ in Equation (29), the vector $\mathbf{D}\mathbf{w}_s^{(i)} = [0]$, $s \in \{1 \cdots S\}$. Thus, from Equations (30) and (31), the PSECMAC learning process converges. It has been proven in [76] that when the learning constant α is such that $0 < \alpha \leq 2$, $\lim_{i \to \infty} (\mathbf{G}_s)^i \mathbf{A}_\upsilon = [0]$ for all $\upsilon \in \{1 \cdots S\}$. Hence, the following theorem is established.

Theorem 1: The training process of the proposed PSECMAC network converges if the learning constant α is such that $0 < \alpha \leq 2$.

Proof: The mathematical proof presented in [76] and [75] shows that for all $\upsilon \in \{1 \cdots S\}$, when $0 < \alpha \leq 2$, $\lim_{i \to \infty} (\mathbf{G}_s)^i \mathbf{A}_\upsilon = [0]$. From the arguments above, the training process of the PSECMAC network converges. ■

VI. Experimental Results and Analysis

This section presents the experiments that have been conducted to evaluate the performance of the proposed PSECMAC associative memory network, namely: (1) the pricing of GBP vs. USD currency futures option; (2) US banking failure classification; and (3) the modeling of the plasma insulin dynamics of the human metabolic process. The performances of the PSECMAC network are dutifully

evaluated against the basic CMAC network and two representative CMAC variants, namely: (1) the Multi-Resolution CMAC (MR-CMAC) [27], and (2) the Fuzzy CMAC with Yager Inference Scheme (FCMAC-Yager) [37]. Other benchmarking architectures studied in this paper include the Generic Self-Organizing Fuzzy Neural Network (GensoFNN) [77], the Rough Set-Based Pseudo-Outer-Product Fuzzy Neural Network (RSPOP) [78], as well as the classical machine learning models such as the Radial Basis Function (RBF) network and a decision table model named Inducers of Decision Table Majority (IDTM) [79], both of which are implemented in the WEKA software package [80].

A. Pricing of GBP vs. USD Currency Futures Option

Options, as a derivative security, provide a means to manage financial risks and they are playing an increasingly important role in modern financial markets [81]. The buyer of an option enters into a contract with the right, but not the obligation, to purchase or sell an underlying physical or financial asset at a later date at a price agreed upon today. The price of an option is determined by a set of pricing factors such as time to expiry and the intrinsic value of the option. A vital aspect of option trading is to derive and be aware of the theoretical fair value of an option. This process is called *option pricing*. The conventional approach to option pricing is to construct parametric models that are based on the assumptions of continuous-time finance theory [82]. The pioneering models are the *Black-Scholes formula* [83] and the *Binomial Pricing Model* [84]. However, these models presume complex and rigid statistical formulations from which the prices are deduced [85]. Nonparametric methods of option pricing based on neural networks [86]–[88], genetic algorithms [89], kernel regression [90], and neuro-fuzzy formulations [91], on the other hand, are model-free approaches. The pricing model, which represents a nonlinear functional mapping between the input factors and the theoretical option price, is derived from vast quantities of historical data.

This study investigates the use of the proposed PSECMAC network for the pricing of American style options on currency futures. In this experiment, the PSECMAC network is used to construct a pricing model to predict the correct valuations for American call options on the British pound (GBP) and US dollar (USD) exchange rate futures contract. The option pricing formula is represented as a function of the following inputs: S_0, X, T, and σ_{30}; where S_0 is the current GBP vs. USD exchange rate futures contract value; X is the strike price of the option; T is the time to maturity of the option in years; and σ_{30} denotes the historical price volatility of the futures contracts for the past 30 trading days. Subsequently, the notion of *moneyness* (or intrinsic value) of a futures option is introduced, which is defined as the difference between the current futures contract value S_0 and the strike price X (i.e. $S_0 - X$). Thus, the

TABLE I
SIMULATION SET-UPS BASED ON PERMUTATIONS OF THE THREE SUB-GROUPS A, B AND C TO DEFINE THE TRAINING AND TESTING SETS OF THE PROPOSED PSECMAC OPTION PRICING MODEL

Evaluation Model	Configuration	Simulation	Training set	Testing set
Model 1	$1/3$ training and $2/3$ testing	I	Sub-group A	Sub-groups B and C
		II	Sub-group B	Sub-groups A and C
		III	Sub-group C	Sub-groups A and B
Model 2	$2/3$ training and $1/3$ testing	IV	Sub-groups A and B	Sub-group C
		V	Sub-groups A and C	Sub-group B
		VI	Sub-groups B and C	Sub-group A

option pricing function f for the valuation of the American call options on the GBP vs. USD futures to be approximated by the PSECMAC network is defined as:

$$C_0 = f(S_0 - X, T, \sigma_{30}) \tag{49}$$

where C_0 is current call option price; and $(S_0 - X)$ reflects the moneyness of the option.

The data used in this study consists of the daily closing quotes of the GBP versus USD currency futures and the daily closing bid and ask prices of the American style options on such futures on the Chicago Mercantile Exchange (CME) [92] during the period of Sept 2002 to Aug 2003. In total, 792 data samples are available in the selected futures option dataset, which contains the historic pricing data for five different strike prices: $\$158$, $\$160$, $\$162$, $\$166$ and $\$168$, with 159, 158, 173, 137 and 165 data samples respectively. The 792 data samples are subsequently partitioned into three evenly distributed sub-groups denoted as A, B and C, where each subgroup contains 264 data tuples. A total of six different cross-validation (CV) sets are constructed based on the permutations of these sub-groups as outlined in Table I. The six CV sets are organized into two different evaluation models, namely Model 1 and Model 2. In Model 1, the training set is constructed using data samples from only one sub-group while the data from the remaining two sub-groups constitute the testing set. The objective of this evaluation model is to assess the generalization ability of the trained pricing system. In contrast, Model 2 employs the data samples from two sub-groups for training and aims to investigate the performances of the benchmarked pricing systems as more training samples are provided.

A PSECMAC network with a memory size of 12 cells per dimension is constructed for the option pricing problem. A neighborhood size (N) of 0.2 and a Gaussian width constant (γ) of 0.5 have been empirically determined. Table II lists the *recall* (in-sample testing) and *generalization* (out-of-sample testing) performances of the PSECMAC option pricing model for the various CV sets. *RMSE*

TABLE II
PERFORMANCES OF THE PROPOSED PSECMAC OPTION PRICING MODEL

Evaluation Model	Simulation	Recall		Generalization	
		RMSE	**PearCorr**	**RMSE**	**Correlation**
Model 1	I	0.1299	0.9956	0.2386	0.9858
	II	0.1376	0.9954	0.2727	0.9816
	III	0.1178	0.9964	0.2638	0.9847
	Average	0.1284	0.9958	0.2584	0.9840
Model 2	IV	0.1382	0.9952	0.2103	0.9889
	V	0.1404	0.9949	0.2210	0.9885
	VI	0.1353	0.9954	0.2007	0.9902
	Average	0.1380	0.9952	0.2107	0.9892

denotes the root-mean-square-error between the set of predicted and desired option prices; and *PearCorr* is the Pearson correlation coefficient, a statistical measure reflecting the goodness-of-fit between the predicted and desired pricing functions. The performances of the proposed PSECMAC option pricing model are encouraging, with an average RMSE of approximately 0.13 and 0.26 for the recall and generalization assessments of Model 1 respectively. An average correlation of 0.98 is achieved by the PSECMAC model for the generalization evaluation, indicating a less than 2% performance degradation as the evaluation emphasis shifts from the in-sample testing (recall) to the out-of-sample evaluation (generalization) capability of the PSECMAC pricing system. From Table II, one can also observe that a larger training dataset improves the generalization performance of the PSECMAC option pricing model. The experimental results of Model 2 showed a 18% improvement ($(0.2584-0.2107)/0.2584$) in the RMSE value over that of Model 1 for out-of-sample testing. This increase in the accuracy of the PSECMAC option pricing model can be attributed to the improvement in the network's ability to efficiently capture the price dynamics and the valuation principles of the futures options with respect to the underlying pricing factors as the number of training instances increases. A larger training dataset results in a more comprehensive training of the entire PSECMAC associative memory surface and this increases the generalization ability of the network to previously unseen test samples.

Subsequently, the entire set of option pricing simulations is repeated using the other benchmarked architectures. To ensure a fair comparison, the size of the CMAC network is defined as 12 cells in each input dimension, while the MR-CMAC structure that is evaluated consists of two layers, each with 6 and 12 memory cells in each input dimension respectively. The parameters for the FCMAC-Yager, RSPOP-CRI, GenSoFNN-TVR [93] and IDTM systems have been empirically optimized for best performances, while the RBF network is initialized to contain 100 hidden layer nodes. Based on the optimal setting,

TABLE III

BENCHMARKING RESULTS FOR VARIOUS OPTION PRICING MODEL

Evaluation		**Recall**				**Generalization**			
Model	**System**	**ARMSE**	**StdDev**	**APC**	**PI$_1$**	**ARMSE**	**StdDev**	**APC**	**PI$_1$**
Model 1	CMAC	0.0531	0.0124	0.9992	94.88	0.2896	0.0229	0.9792	75.93
	MR-CMAC	0.0526	0.0107	0.9993	94.93	0.3838	0.0618	0.9639	69.66
	PSECMAC	**0.1284**	**0.0099**	**0.9958**	**88.24**	**0.2584**	**0.0177**	**0.9840**	**78.19**
	FCMAC-Yager	0.1924	0.0275	0.9911	83.12	0.5221	0.1929	0.9236	60.68
	GenSoFNN-TVR	0.1759	0.0153	0.9943	84.56	0.2764	0.0507	0.9839	77.08
	RSPOP-CRI	0.2562	0.0257	0.9849	78.40	0.4204	0.0630	0.9578	67.43
	RBF	0.1767	0.0389	0.9920	84.30	0.3438	0.1085	0.9701	72.19
	IDTM	0.2388	0.0407	0.9853	79.54	0.3964	0.0512	0.9596	68.72
Model 2	CMAC	0.0678	0.0032	0.9988	93.54	0.2579	0.0102	0.9834	78.18
	MR-CMAC	0.0710	0.0022	0.9987	93.25	0.3179	0.0409	0.9756	74.03
	PSECMAC	**0.1380**	**0.0026**	**0.9952**	**87.45**	**0.2107**	**0.0102**	**0.9892**	**81.70**
	FCMAC-Yager	0.2365	0.0261	0.9879	79.89	0.2829	0.0110	0.9831	76.63
	GenSoFNN-TVR	0.1857	0.0103	0.9948	83.90	0.2387	0.0198	0.9908	79.99
	RSPOP-CRI	0.2938	0.0131	0.9787	75.65	0.3461	0.0621	0.9695	72.02
	RBF	0.2389	0.0323	0.9854	79.54	0.3076	0.0470	0.9758	74.63
	IDTM	0.2721	0.0124	0.9812	77.13	0.3208	0.0092	0.9740	73.74

the FCMAC-Yager network on average employs 12 memory cells for each input dimension. Table III summarizes the average RMSE (ARMSE), standard deviation (StdDev) and average Pearson correlation (APC) findings for the evaluation Model 1 and Model 2 across the different architectures. A *Performance Index* (PI$_1$) is used to combine the ARMSE and APC measures as described in Equation (50).

$$\text{PI}_1 \quad = \quad \frac{APC}{(1 + ARMSE)} \times 100 \tag{50}$$

such that a higher PI$_1$ value corresponds to a better pricing performance.

From Table III, one can observe that the proposed PSECMAC network achieved the best generalization performance index (PI$_1$) as compared to the rest of the pricing systems for both evaluation models. The PSECMAC network achieved a generalization PI$_1$ of 78.19 for Model 1 and 81.70 for evaluation Model 2 respectively. The PSECMAC network has also comprehensively outperformed the basic CMAC network and the benchmarked CMAC variants based on the generalization results, thereby demonstrating clearly the effectiveness of the judicious memory cells allocation process of its self-organizing structure. Specifically, the multi-resolution structure of the PSECMAC network yields, on average, a 3.7% improvement in PI$_1$ value over the uniformly-quantized CMAC of the same network size. However, the recall performances of the PSECMAC network were slightly lower than those of the CMAC and MR-CMAC networks. This

is because the static uniform memory quantization of the CMAC and MR-CMAC networks results in structures that are highly optimized for the training set. The PSECMAC memory allocation procedure, on the other hand, is geared more towards obtaining an efficient characterization of the problem's input-output mappings. This is achieved by allocating the memory cells in a non-linear manner based on the distribution of the training data. The effective non-linear memory quantization of PSECMAC allows for a better description of the problem's characteristic surface to address a new/unseen testing data. Thus, PSECMAC is able to achieve an improved generalization performance despite a trivial degradation in the recall performance.

Table III also showed that the generalization performances of the benchmarked CMAC variants were inferior to those of the CMAC as well as the PSECMAC networks. The MR-CMAC network [27] is essentially a multi-layered multi-resolution CMAC system that employs a hierarchy of CMACs with increasing modeling resolutions to improve the generalization and accuracy of the system. Unlike the proposed PSECMAC network that allocates the memory cells selectively to enhance the network's generalization capability and the accuracy of the computed output, the MR-CMAC network employs a coarsely partitioned CMAC in the base layer to generalize the characteristics of the training data. MR-CMAC's output accuracy is then gradually refined with the increasingly finer modeling resolutions of the higher-layer CMACs. Such a hierarchical associative memory model, however, may not be suitable for applications with a high output volatility. The large performance degradation by the MR-CMAC network as shown from the recall to the generalization pricing assessment in Table III is a clear indication of the network's inability to adequately generalize the characteristics of the option dataset using the coarsely partitioned base layer and to subsequently learn the specificity of the pricing data using a second layer of computing cells. The FCMAC-Yager network, on the other hand, suffers from a poor output accuracy for both the recall and the generalization assessments due to the fuzzification of the inputs. FCMAC-Yager is a Mamdani fuzzy rule based system and employs trapezoidal-shaped membership functions that often lead to a low output accuracy due to the granularity of the membership functions. This is an inherent limitation associated with the use of the Mamdani rule system form of knowledge representation.

The proposed PSECMAC network also achieved more accurate pricing decisions as compared to the benchmarked neuro-fuzzy systems (i.e. GenSoFNN, RSPOP) and classical machine learning techniques of RBF and IDTM. The evaluation results of Model 1 have shown that the PSECMAC network is able to efficiently generalize the characteristics of the training data despite a small training set. The performances of the proposed PSECMAC network compared favorably to those of the GenSoFNN-TVR, RSPOP-CRI, RBF, and IDTM pricing models for the generalization assessment of Model 1. The simulation results of

TABLE IV
COMPARISON OF THE CELL OCCUPANCY RATE OF THE PSECMAC AND CMAC OPTION PRICING MODEL

	CMAC			PSECMAC			
Simulation	**Total Cells**	**Trained Cells**	**COR**	**Total Cells**	**Trained Cells**	**COR**	**Improvement**
I	1728	1064	61.57%	1728	1170	67.71%	6.14%
II	1728	1082	62.62%	1728	1178	68.17%	5.55%
III	1728	1088	62.96%	1728	1175	68.00%	5.04%
IV	1728	1126	65.16%	1728	1297	75.06%	9.91%
V	1728	1125	65.10%	1728	1227	71.01%	5.91%
VI	1728	1116	64.58%	1728	1276	73.84%	9.26%
Average			**63.67%**			**70.63%**	**6.96%**

Model 2 for out-of-sample testing further verified the effectiveness of the PSECMAC memory allocation procedure in addressing the generalization–accuracy dilemma. The consistency in the PSECMAC pricing performances is also evidenced by the StdDev values listed in Table III. The proposed PSECMAC option pricing model achieved the lowest standard deviation for evaluation Model 1, with the StdDev values of 0.0099 and 0.0177 for recall and generalization respectively. For Model 2, PSECMAC yields StdDev values of 0.0026 (recall) and 0.0102 (generalization), which are highly comparable to the best-achieved StdDev values of 0.0022 (MR-CMAC) and 0.0092 (IDTM) for the recall and generalization assessments respectively. In summary, Table III have adequately demonstrated the performance of the proposed PSECMAC network as an accurate option pricing model.

Subsequently, the effective cell utilization rates of the PSECMAC network for all the six CV sets were computed and compared against those of the CMAC network. The results (denoted as the Cell Occupancy Rate) are tabulated as Table IV. The Cell Occupancy Rate (COR) is defined as the proportion of the trained memory cells to the total network size. From Table IV, one can conclude that the PSECMAC network consistently achieves a higher COR value than the CMAC network. An average improvement of 6.9% was achieved for COR and this clearly demonstrates the effectiveness of the non-linear memory allocation scheme of the proposed PSECMAC network in reducing memory wastage. A higher COR value implies a more comprehensive training of the CMAC/PSECMAC input-output associative space and indirectly translates to an improved generalization ability of the network. Such a notion is reinforced by the higher generalization accuracy achieved by the PSECMAC network over the CMAC network.

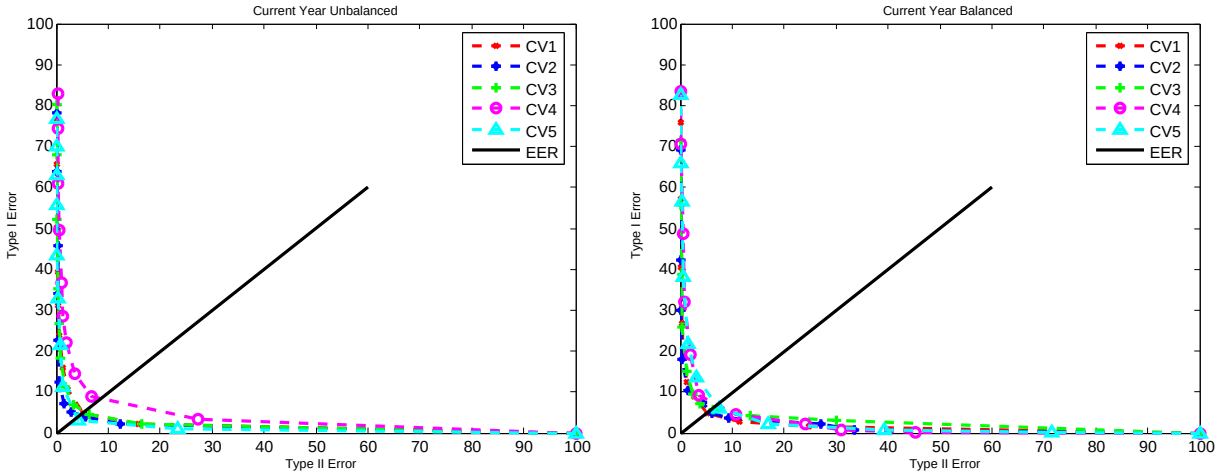
B. Banking Failure Classification

Bank failure prevention is an important issue for the regulators of the banking industries. The collapse and failure of a bank could trigger an adverse financial repercussion and generate negative impacts such

as a massive bail out cost for the failing bank and loss of confidence from the investors and depositors. Technically, banks do not fail overnight, and very often, bank failures are due to prolonged periods of *financial distress*. Banking regulators can therefore establish the traits of financial distress that characterize bank failures to identify a potential failing bank. Some commonly used statistical methodologies are *multivariate discriminant analysis* [94], *logit analysis* [95] and Cox's *proportional hazards model* [96]. However, the use of such classical approaches generally results in a decision model that is neither adaptive nor computationally interpretable to the banking regulators [97]. The PSECMAC network, on the other hand, is an associative memory network that employs a data-driven approach for its structural and network learning process. The single-layered PSECMAC network structure facilitates the automatic identification and subsequent human interpretation of the traits of financial distress characterizing a bank failure. This motivates the use of the proposed PSECMAC network to identify problem banks using the *financial covariates* extracted from the financial statements reported by the banks.

In this experiment, the monitored banks are classified as failed or survived (non-failing) banks based on their financial performances. Financial variables (covariates) used to characterize the banks' operational quality were extracted from the Call Reports available from the Federal Reserve Bank of Chicago [98]. There are a total of nine variables and they are listed in [99]. *Regulatory closure* is the defining event of the failure of a bank. The observation period of the survived (non-failing) banks spans from January 1980 to December 2000 inclusively. For consistency, the financial reports for the failed and survived banks have the same balance sheet dates. The original dataset is pre-processed to filter out the last available financial statement for each bank in the observation period. For the failed banks, this refers to the last records prior to failure, while the last records for the surviving banks are those submitted in year 2000 (last year of the observation period). After removing banks with missing data in their records, the final dataset consists of 548 failed banks (with failure dates spreading across the entire observation period) and 2555 banks that survived the observation period. Thus, the failed banks constituted approximately 17.7% of the dataset while the survived banks account for the remaining 82.3%.

Based on the nine selected covariates, bank failure classification using the data extracted from the last financial statements was performed. The entire banking dataset is partitioned into five mutually exclusive cross-validation (CV) groups denoted as CV1–5. Each group consists of one training and one test set that are randomly generated from the set of selected surviving and failed banks. The banking dataset is initially segregated into two pools: failed and survived (non-failing) banks. For each cross-validation group, 20% of both pools are randomly selected to form the training set while the remaining 80% of the data constitute the test set. Hence, the number of survived banks far exceeds the failed banks ("unbalanced" training



(a) Unbalanced Training Scenario

(b) Balanced Training Scenario

Fig. 8. ROC curves of the PSECMAC banking failure classification

scenario). A single output is used to differentiate between failed and survived banks. Failed banks are denoted with output ”1” while survived banks are identified by output ”0”.

The proposed PSECMAC network and the benchmarked architectures which include CMAC, MR-CMAC, FCMAC-Yager and GenSoFNN-CRI [99] were used to perform the bank failure classification. The systems were constructed using the training set and the performances of the trained models are evaluated using the testing set. The simulation is performed for all the five CV groups. The classification threshold (to discern between failed and survived (non-failing) banks based on the nine input financial covariates) is varied to obtain the receiver-operating-characteristic (ROC) curves of each evaluated system. The *Equal Error Rate* (EER) values extracted from the ROC curves are subsequently employed as the performance measure of the respective models. Type I error is defined as the error of mis-classifying a failed bank as a survived (non-failing) one whereas Type II error is the mis-classification of a non-failing bank as a failed entity. EER denotes the point where Type I equals Type II error.

The best classification performance for the PSECMAC network was empirically determined with a network size of 3 cells per dimension, a neighborhood parameter (N) of 0.3 and a Gaussian width constant (γ) of 0.3. The CMAC network, on the other hand, obtained an optimal EER measure with 4 memory cells in each dimension, whereas the MR-CMAC network achieves optimal classification rates using two layers of CMAC with 2 and 4 cells in each dimension respectively. The parameters for the remaining benchmarked systems (i.e. FCMAC-Yager and GenSoFNN-CRI) have all been empirically

TABLE V
EER READINGS EXTRACTED FROM THE ROC CURVES - UNBALANCED TRAINING SCENARIO

	Equal Error Rate (EER) [%]					Mean EER	StdDev EER
System	**CV1**	**CV2**	**CV3**	**CV4**	**CV5**	**[%]**	**[%]**
CMAC	4.386	6.552	5.661	12.16	6.552	7.0622	2.9847
MR-CMAC	5.208	6.32	5.473	12.38	6.876	7.2514	2.9432
PSECMAC	**5.097**	**4.453**	**5.328**	**8.445**	**3.91**	**5.4466**	**1.7660**
FCMAC-Yager	8.93	7.13	7.77	8.23	5.08	7.43	1.4678
GenSoFNN-CRI	4.4	10.99	5.49	15.6	16.48	10.59	5.5740

optimized. The FCMAC-Yager system on average uses 4 cells in each dimension. Figure 8(a) depicts the ROC plots of the PSECMAC network banking failure classification system. The EER measures of the various benchmarked systems are listed as Table V. It is evident that the classification performance of the proposed PSECMAC network surpasses all the benchmarked architectures. The PSECMAC network obtained a significantly lower EER value of approximately 5.4% as compared to the 7% EER achieved by the CMAC network and the 10.5% EER of GenSoFNN-CRI. The experimental results have clearly demonstrated the superior modeling accuracy of the proposed PSECMAC network that is due to its non-uniform memory allocation procedure. That is, when compared to CMAC and MR-CMAC, PSECMAC has achieved a superior classification rate even when based on a smaller network size.

In addition, from Table V, one can observe that the classification performances of the CMAC-based systems outperformed those achieved by the benchmarked GenSoFNN-CRI network. A plausible explanation is the use of the ”unbalanced” training scenario where the proportion of survived to failed banks in the training set is highly skewed. GenSoFNN-CRI is a connectionist neural fuzzy system whereby each presentation of the training data activates and trains the entire computing structure. Thus, the learning principle of the GenSoFNN-CRI to generalize or fit the behaviors/characteristics of such unbalanced training set would likely contribute to its poor performances to discern the two overlapping but contrasting concepts (i.e. failed and survived banks) as in the banking dataset. The PSECMAC, CMAC, MR-CMAC and FCMAC-Yager, on the other hand, consist of an ensemble of locally active models constructed from different segments of the training data. In such a local learning model, a skewed dataset has a limited effect on the learned data characteristic mappings as compared to the GenSoFNN network that learns globally. Hence, a local learning model such as PSECMAC can effectively segregate the information expressed by the two contrasting groups of banks even though the ratio of survived to failed banks is highly skewed. The results in Table V have demonstrated that the performance of PSECMAC, which employs localized learning, is less likely to be perturbed by a statistically skewed dataset.

TABLE VI
EER READINGS EXTRACTED FROM THE ROC CURVES - BALANCED TRAINING SCENARIO

	Equal Error Rate (EER) [%]					Mean EER	StdDev EER
System	**CV1**	**CV2**	**CV3**	**CV4**	**CV5**	**[%]**	**[%]**
CMAC	7.214	8.153	5.48	6.388	9.951	7.4372	1.7185
MR-CMAC	8.474	9.212	10.00	10.10	11.3	7.8172	1.0586
PSECMAC	**4.995**	**5.574**	**5.898**	**6.932**	**6.90**	**6.0594**	**0.8460**
FCMAC-Yager	8.77	8.78	7.5	8.19	8.42	8.33	0.5274
GenSoFNN-CRI	7.08	9.38	5.63	13.96	4.79	8.17	3.6760

The set of simulations on the benchmarked systems is subsequently repeated with a ”balanced” training scenario. The training sets of the five CV groups are modified by randomly pruning away redundant survived banks until the number of survived banks equals the failed banks. The testing sets of the CV groups remain unchanged. The ROC curves of the proposed PSECMAC network for the ”balanced” training scenario are depicted as Figure 8(b). The EER measures of the various benchmarked systems for the ”balanced” training scenario are subsequently listed as Table VI. As observed, the mean EER value of the GenSoFNN-CRI network for the ”balanced” training scenario improved by 2.4% in comparison with the ”unbalanced” training scenario (note that the testing sets are the same for both scenarios). In addition, the simulation results in Tables V and VI also demonstrated that the PSECMAC and the CMAC-based systems yielded slightly higher EER values for the ”balanced” than the ”unbalanced” training scenarios. This minor degradation of classification performance (0.52% on average) could be attributed to the fact that the training sets in the ”balanced” training scenario consist of a smaller number of training samples from which these localized learning networks can learn from the actual data distribution, and thus slightly poorer classification performances resulted. However, from the standard deviation (StdDev) of the EER measurements listed in Tables V and VI, one can observe that the performances of the proposed PSECMAC network are consistently good.

Lastly, the effective cell utilization rates (COR) of the proposed PSECMAC and CMAC banking failure classification systems across the five CV groups (”unbalanced”) are analyzed. The COR values are tabulated as Table VII. One can observe that the PSECMAC based bank failure classifier achieved a remarkably high COR value of 58% (average) as compared to the CMAC based classifier (4.2% average). This significant improvement of 53.8% in memory efficiency can be attributed to the smaller number of memory cells required by the PSECMAC network (3 cells per dimension) as compared to the CMAC network (4 cells per dimension) in achieving the respective optimal classification results. Thus, the COR results of this experiment has clearly demonstrated the effectiveness of the memory allocation scheme

TABLE VII
COMPARISON OF THE CELL OCCUPANCY RATE OF THE PSECMAC AND CMAC BANKING FAILURE CLASSIFICATION

	CMAC			PSECMAC			
Simulation	**Total Cells**	**Trained Cells**	**COR**	**Total Cells**	**Trained Cells**	**COR**	**Improvement**
CV1	262144	19368	7.39%	19683	11074	56.26%	48.87%
CV2	262144	4752	1.81%	19683	6142	31.20%	29.39%
CV3	262144	16125	6.15%	19683	18768	95.35%	89.20%
CV4	262144	9765	3.73%	19683	14512	73.73%	70.00%
CV5	262144	4478	1.71%	19683	6580	33.43%	31.72%
Average			**4.16%**			**57.99%**	**53.83%**

employed by the proposed PSECMAC network to resolve the high memory wastage encountered by the CMAC model.

C. Modeling the Plasma Insulin Dynamics of the Human Glucose Metabolic Process

Diabetes is a metabolic disorder where the body is no longer able to properly regulate the use and storage of glucose in the blood, leading to prolonged periods of high (hyperglycemia) or low (hypoglycemia) plasma glucose concentration. Currently, the standard medical treatment of diabetes primarily involves insulin medication coupled with strict dietary control. The fundamental objective of the insulin therapy for diabetes treatment is essentially to artificially recreate and replicate the healthy insulin profiles in a diabetic patient in response to metabolic disturbances such as food intakes and exercises, so as to regulate the diabetic blood glucose level within the homeostatic range of 60–110 mg/dl [100]. The human metabolic process consists of highly complex and intertwined relationships between the plasma glucose (originating from the food ingestion) and insulin (produced by the pancreatic β-cells) that ensure the steady supply of energy substrates to maintain bodily equilibrium. This motivates the use of the multi-resolution mapping characteristics of the PSECMAC network to model the dynamics of the insulin response to food intakes in a healthy subject. Such a non-linear healthy insulin model could subsequently be used to formulate an appropriate insulin therapy schedule for the treatment of diabetes.

Due to the lack of real-life patient data and the logistical difficulties and ethical issues involving the collection of such data, a well-known web-based simulator known as *GlucoSim* [101] from the Illinois Institute of Technology is employed in this study to simulate a person subject to generate the metabolic data that is needed for the construction of the healthy insulin response model. For this purpose, a human profile for the simulated healthy subject is created and described in Table VIII. The simulated person, Subject A, is a typical middle-aged Asian male. His body mass index (BMI) is 23.0, which is within

Attribute Name	Attribute Value
Sex	Male
Age	40 years old
Race	Asian
Weight	67 kg (147.71 lbs)
Height	1.70 m (5 ft 7 in)
BMI	23 (Recommended for Asian)
Lifestyle	Typical office worker with moderate physical activities such as walking briskly, leisure cycling and swimming.

TABLE VIII
THE PROFILE OF THE SIMULATED HEALTHY SUBJECT A

the recommended range for Asian. Based on the profile of Subject A, his recommended daily allowance (RDA) of carbohydrate intake from meals is obtained from the website of the Health Promotion Board of Singapore [102]. According to his sex, age, weight and lifestyle, the recommended daily carbohydrate intake for Subject A is approximately 346.9g.

The GlucoSim simulator requires 10 different inputs, which consists of the body weight, the simulation period, and both the time and carbohydrate content of each of the assumed daily four meals, namely: breakfast, lunch, afternoon snack, and dinner respectively. The carbohydrate contents and the timings of the daily meals are varied from day-to-day during the data collection phase. This is to account for the inter- and intra-day variability of the eating habits of Subject A and to ensure that the PSECMAC insulin model is not being trained on a cyclical dataset but elicits the inherent relationships between food intakes and the insulin response of a healthy person. Since the human glucose metabolic process depends on the current (and internal) body states as well as exogenous inputs (or disturbances) such as food intakes, it is hypothesized that the plasma insulin concentration level at any given time t is a non-linear function of prior food intakes and the historical traces of the insulin and blood glucose levels. To properly account for the effects of prior food ingestion to the observed blood insulin level at time t, a historical sliding window of six hours with respect to t was adopted and a soft-windowing strategy is employed to partition the six hours into three conceptual segments, namely: *Recent* (i.e. previous 1 hour), *Intermediate Past* (i.e. previous 1 to 3 hours) and *Long Ago* (i.e. previous 3 to 6 hours), resulting in only three food history inputs. Based on these windows, three normalized weighting functions are introduced to compute the carbohydrate content of the meal(s) taken within the recent, intermediate past or long ago periods. Thus, inclusive of the previously measured blood glucose and insulin levels, there are a total of five inputs to the modeling task.

TABLE IX

COMPARISON OF THE MODELING PERFORMANCES OF THE VARIOUS INSULIN MODELS

System	Recall			Generalization		
	RMSE [microU/ml]	**PearCorr**	**PI$_2$**	**RMSE** [microU/ml]	**PearCorr**	**PI$_2$**
CMAC	4.5513	0.9948	17.92	11.8540	0.9695	7.54
MR-CMAC	4.4298	0.9951	18.33	11.6335	0.9722	7.70
PSECMAC	**4.4441**	**0.9951**	**18.28**	**9.7172**	**0.9795**	**9.14**
FCMAC-Yager	13.6142	0.9752	6.67	14.7771	0.9703	6.15
GenSoFNN-CRI	13.5270	0.9884	6.80	14.7830	0.9842	6.24
RSPOP-CRI	9.5183	0.9797	9.31	11.8500	0.9709	7.56
RBF	10.7197	0.9709	8.28	12.1232	0.9678	7.37
IDTM	5.2705	0.9930	15.84	17.3263	0.9336	5.09

Based on the daily inputs presented to GlucoSim, a total of 100 days of metabolic data for Subject A was collected and a sampling interval of 15 minutes is adopted throughout the data collection phase. The collected dataset is subsequently partitioned into two non-overlapping groups: 20 days of data for training and the remaining 80 days for testing and evaluation of the insulin model. Simulations to model the dynamics of the insulin response of Subject A using the PSECMAC network were performed and the results were benchmarked against those of the basic CMAC, MR-CMAC, FCMAC-Yager, GenSoFNN-CRI, RSPOP-CRI, RBF as well as the IDTM. Both the CMAC and PSECMAC networks are constructed with a memory size of 6 cells per dimension, and a neighborhood constant (N) of 0.1 and a Gaussian width constant (γ) of 0.3 are adopted for the PSECMAC network. On the other hand, the MR-CMAC network is evaluated with two layers of computing cells, each containing 3 and 6 memory cells per dimension respectively. The RBF network contains 100 hidden layer nodes, while the parameters of the FCMAC-Yager, GenSoFNN-CRI, RSPOP-CRI and IDTM systems have all been empirically optimized. Table IX details the recall (training) and generalization (testing) performances of the various benchmarked systems. The two performance indicators employed in the study to quantify the modeling quality of the systems are: the *root mean-squared error* (RMSE) and the *Pearson correlation coefficient* (PearCorr) between the actual (observed) and the predicted (computed) plasma insulin levels. These RMSE and PearCorr measures were subsequently employed to compute the *Performance Index* (denoted as PI$_2$) described by Equation (51).

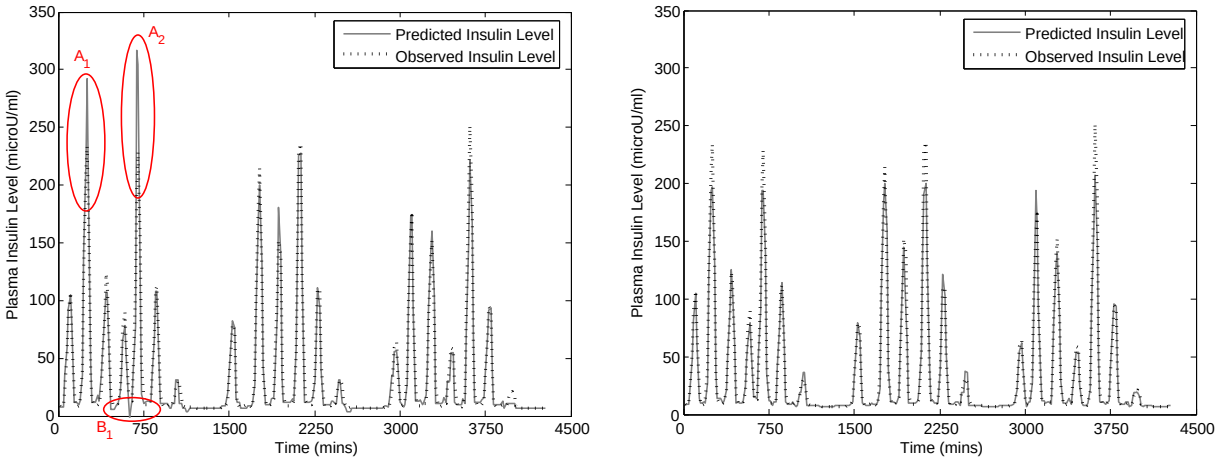
$$\text{PI}_2 \;=\; \frac{\textit{PearCorr}}{(1+\text{RMSE})} \times 100 \tag{51}$$

Hence, a higher PI$_2$ value reflects a better insulin modeling performance.

From the simulation results tabulated in Table IX, one can observe that the best recall performance was achieved by the MR-CMAC network, with an RMSE of about 4.43 microU/ml and PearCorr of 99.5% between the computed and the actual plasma insulin level. The proposed PSECMAC insulin model, on the other hand, achieved a slightly higher RMSE value of about 4.44 microU/ml. This results in a slightly poorer recall performance ($\text{PI}_2 = 18.28$) as compared to that of the MR-CMAC-based insulin model ($\text{PI}_2 = 18.33$). As described previously, the MR-CMAC network employs a layered structure of computing (memory) cells with different resolutions to capture both the general characteristics as well as the minute details of the training data but at the expense of high memory requirement. However, the PSECMAC insulin model has achieved a highly comparable recall performance to the MR-CMAC model despite using a much smaller network size. This clearly demonstrated the effectiveness of the adaptive memory allocation scheme of the proposed PSECMAC network in the characterization of the the human metabolic process.

Table IX also showed that the PSECMAC insulin model achieved the best generalization performance amongst all the evaluated systems. The multi-resolution structure of the PSECMAC network resulted in the lowest RMSE of 9.72 microU/ml and a comparatively high correlation of 97.9% between the computed insulin levels and the actual healthy insulin responses, which led to the highest generalization PI_2 value of 9.14 amongst the benchmarked systems. The MR-CMAC network, which has the best performance for the recall evaluation, yielded a PI_2 value of 7.70 for the generalization assessment. This further reinforces the notion that the trained MR-CMAC insulin model is highly optimized only for the training dataset. The generalization capability of the PSECMAC model, on the other hand, demonstrated that the network is able to efficiently extract the inherent relationships from the training data. As shown in Table IX, the PSECMAC insulin model also comprehensively outperformed the basic CMAC, FCMAC-Yager and other benchmarked models. A generalization performance gain of 21.1% in the PI_2 value was achieved by the adaptive memory allocation process of the PSECMAC model over the uniformly-quantized basic CMAC model.

To further analyze the modeling performance of the PSECMAC network, the first three-days of the insulin predictive results of the CMAC and PSECMAC insulin models for the generalization evaluation are depicted in Figure 9. Two large overshoots are recorded in the computed CMAC insulin profile (as highlighted by A_1 and A_2) in Figure 9(a). In addition, the static uniform quantization of the basic CMAC network has also resulted in regions of untrained network cells and the effect of such untrained network cells is highlighted as B_1 (in Figure 9(a)). The non-uniform memory allocation scheme of the PSECMAC network, on the other hand, is able to efficiently allocate the available memory cells according



(a) Generalization performance of CMAC insulin model

(b) Generalization performance of PSECMAC insulin model

Fig. 9. Modeling results of the CMAC and PSECMAC networks for the insulin response of Subject A

to the information distribution of the training data, and thus reduces the number of untrained network cells to facilitate a consistent modeling performance as shown in Figure 9(b). This observation is further reinforced by the analysis of the effective cell utilization rate (COR) of the PSECMAC and CMAC insulin models. The CMAC insulin model trained on the 20 days of metabolic data has 462 trained cells out of a total of 7776 network cells. This translates to a COR of 5.94%. The PSECMAC insulin model, meanwhile, achieved a substantially higher COR value of 14.57% with a total of 1133 trained cells. The insulin modeling results as shown in Table IX and Figure 9 have therefore clearly demonstrated the performance superiority of the PSECMAC over the basic CMAC network and once again highlighted the effectiveness of the proposed multi-resolution structure of the PSECMAC network.

VII. Conclusions

This paper presents a novel neurophysiologically-inspired multi-resolution cerebellar associative memory model named PSECMAC that is used to address the architectural deficiencies of the CMAC network. Motivated by the experience-driven synaptic plasticity phenomenon observed in the learning and adaptation process of the human cerebellum, the PSECMAC network employs a data-driven memory quantization scheme for the derivation of its computing structure. In the proposed PSECMAC network, the biological cerebellar synaptic adaptation process is emulated by allocating more memory cells to the data-intensive regions of the input space that correspond to the frequently-accessed areas of the cerebellum. This

translates to a finer output resolution in the task-critical (important) regions of the PSECMAC input-output (I/O) associative space, which is analogous to the smooth and dexterous execution of well-trained motor skills observed in human behavioral studies. The structural formation and the subsequent learning processes of the proposed PSECMAC network are presented in the paper together with the theoretical proof of the learning convergence of the system.

The performance of the PSECMAC network was subsequently evaluated using three real-life applications, namely: the pricing of the GBP vs. USD currency futures options, the classification of US banking failures, and the modeling of the plasma insulin dynamics of the human metabolic process. PSECMAC was benchmarked against the CMAC network and two representative CMAC variants (MR-CMAC and FCMAC-Yager) as well as the GenSoFNN and RSPOP neuro-fuzzy systems and classical machine learning techniques such as RBF and IDTM. The computed simulation results have adequately demonstrated the effectiveness of the proposed PSECMAC network architecture in capturing the complex input-output relationships of the three applications while addressing the architectural limitations of the CMAC network. The PSECMAC network also outperformed all the benchmarked systems and has achieved significant structural improvements over CMAC and its two well-established variants. It is evident that the judicious memory allocation scheme of the PSECMAC network results in more comprehensive training of its memory space as reflected by the significantly higher rate of memory utilization when compared to the CMAC network.

Although the proposed PSECMAC network has achieved a significantly higher memory utilization rate, the non-linear memory quantization process of the PSEMAC network is currently performed separately for each input. This attempts to reduce the computational complexity arising from the application of the novel non-uniform quantization scheme to define the associative memory structure of the PSECMAC network, and has been shown by the simulation results to work reasonably well for low dimensional problems (i.e. up to 9-dimensional input for the banking failure classification problem). However, such a memory allocation scheme may lead to significant memory wastage when scaled to high dimensional problems. Also, as compared to the basic CMAC, PSECMAC incurs a one-time extra computational load during its structural learning phase due to the use of the MPSEC algorithm and the derivation of the non-uniform quantization decision functions to define the computing structure of the network. The operational complexity as well as the computational load of the parameter tuning phase of the PSECMAC network, however, are similar to those of the basic CMAC network. Extensive experimentations and theoretical analyses need to be undertaken to study these issues and research efforts have currently been directed at resolving the limitations of the PSECMAC network. In addition, as the memory quantization (allocation)

scheme employed in the PSECMAC network is based on the computed density profiles of the training data, the proposed network is currently more suited for offline applications.

Future enhancements to the PSECMAC architecture include providing the support for online learning, and the investigation into the usage of the PSECMAC network for other financial applications such as stock trading and portfolio management as well as biomedical engineering deployments such as intelligent ventilator control in intensive care. Currently, these research endeavours are actively underway at the Centre of Computational Intelligence (C2i) [103] located at the School of Computer Engineering in Nanyang Technological University, Singapore. The C2i lab undertakes intense research in the study and development of advanced brain-inspired learning memory architectures [77], [78], [104], [105] for the modeling of complex, dynamic and non-linear systems. These techniques have been successfully applied to numerous novel applications such as signature forgery detection [106], fingerprint verification [107], computational finance [99], [108], [109], as well as in the biomedical engineering domain [93].

References

[1] F. A. Middleton and P. L. Strick, "The cerebellum: an overview," *Trends in Cognitive Sciences*, vol. 27, no. 9, pp. 305–306, 1998.

[2] J. S. Albus, "Marr and Albus theories of the cerebellum: Two early models of associative memory," *Proceedings of IEEE Compcon*, 1989.

[3] H. Eichenbaum, *The Cognitive Neuroscience of Memory: An Introduction*. Oxford University Press, 2002.

[4] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *J. Dynamic Syst., Measurement, Contr., Trans. ASME*, pp. 220–227, 1975.

[5] ——, "Data storage in cerebellar model articullation controller (CMAC)," *J. Dynamic Syst., Measurement, Contr., Trans. ASME*, pp. 228–233, 1975.

[6] F. H. Glanz, W. T. Miller, and L. G. Kraft, "An overview of the CMAC neural network," in *Proc. IEEE Conf. Neural Networks Ocean Eng.*, Washington, D.C., 1991, pp. 301–308.

[7] W. T. Miller, F. H. Glanz, and L. G. Kraft, "CMAC: An associative neural network alternative to backpropagation," *Proceedings of the IEEE, Special Issue on Neural Networks*, vol. 78, no. 10, pp. 1561–1567, 1990.

[8] T. Yamamoto and M. Kaneda, "Intelligent controller using CMACs with self-organized structure and its application for a process system," *IEICE Trans. Fundamentals*, vol. E82-A, no. 5, pp. 856–860, 1999.

[9] S. Ku, G. A. Larsen, and S. Cetinkunt, "Fast servo control for ultra-presision machining at extremely low feed rates," Mechatronics, pp. 381–393, 1998.

[10] G. A. Larsen, S. Cetinkunt, and A. Donmez, "CMAC neural network control for high precision motion control in the presence of large friction," *J. Dynamic Syst., Measurement, Contr.*, vol. 117, pp. 415–420, 1995.

[11] S. Cetinkunt and A. Donmez, "CMAC learning controller for servo control of high precision machine tools," in *Proc. American Contr. Conf.*, San Fransico, CA, 1993, pp. 1976–1980.

[12] L. G. Kraft, W. T. Miller, and D. Dietz, "Development and application of cmac neural network-based control," *In D.A.*

White and D.A. Sofge (Eds.), Handbook of Intelligent Control. Neural, Fuzzy, and Adaptive Approaches, pp. 215–232, 1992.

[13] C. S. Lin and K. Hyongsuk, "CMAC-based adaptive critic self-learning control," *IEEE Trans. Neural Networks*, vol. 2, pp. 530–533, 1991.

[14] S. Commuri, S. Jagannathan, and F. L. Lewis, "CMAC neural network control of robot manipulators," *J. Robot Syst.*, vol. 14, no. 6, pp. 465–482, 1997.

[15] H. Kano and K. Takayama, "Learning control of robotic manipulators based on neurological model CMAC," in *Proc. 11th Triennial World Congr. Int. Federation of Automat. Contr.*, Tallinn, USSR, 1990, pp. 249–254.

[16] W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *International Journal of Robotics Research*, vol. 6, no. 2, pp. 84–98, 1987.

[17] A. Wahab, E. C. Tan, and H. Abut, "HCMAC amplitude spectral subtraction for noise cancellation," *Intl. Conf. Neural Inform. Processing*, 2001.

[18] K. L. Huang, S. C. Hsieh, and H. C. Fu, "Cascade-CMAC neural network applications on the color scanner to printer calibration," *Intl. Conf. Neural Networks*, vol. 1, pp. 10–15, 1997.

[19] J. Ker, C. Hsu, Y. Kuo, and B. Liu, "A fuzzy CMAC model for color reproduction," *Fuzzy Sets and Systems*, vol. 91, no. 1, pp. 53–68, 1997.

[20] C. He, L. Xu, and Y. Zhang, "Learning convergence of CMAC algorithm," *Neural Processing Letters*, vol. 14, no. 1, pp. 61–74, 2001.

[21] C. S. Lin and C. T. Chiang, "Learning convergence of CMAC technique," *IEEE Trans. Neural Networks*, vol. 8, no. 6, pp. 1281–1292, 1997.

[22] Y. Wong and A. Sideris, "Learning convergence in the cerebellar model articulation controller," *IEEE Trans. Neural Networks*, vol. 3, no. 1, 1992.

[23] H. M. Lee, C. M. Chen, and Y. F. Lu, "A self-organizing HCMAC neural network classifier," *IEEE Transactions on Neural Networks*, vol. 14, no. 1, pp. 15–27, 2003.

[24] M. F. Yeh and H. C. Lu, "On-line adaptive quantization input space in CMAC neural network," *IEEE Intl. Conf. Syst., Man, Cybern.*, vol. 4, 2002.

[25] H. Kim and C. S. Lin, "Use of adaptive resolution for better CMAC learning," *Intl. Joint Conf. Neural Networks*, vol. 117, pp. 517–522, 1992.

[26] X. Gao, C. Wang, X. M. Gao, and S. J. Ovaska, "A new CMAC neural network model with adaptive quantization input layer," *3rd. Intl. Conf. Singal Processing*, vol. 2, pp. 1417–1420, 1996.

[27] J. Moody, "Fast-learning in multi-resolution hierarchies," in *Adv. Neural Infor. Processing Syst.* Morgan Kauffman Publishers, 1989, vol. 14, no. 1, pp. 29–38.

[28] A. Menozzi and M. Chow, "On the training of a multi-resolution CMAC neural network," *23rd. Intl. Conf. Ind. Electron. Contr. Instrum.*, vol. 3, pp. 1130–1135, 1997.

[29] J. Ozawa, I. Hayashi, and N. Wakami, "Formulation of CMAC-fuzzy system," *IEEE International Conference on Fuzzy Systems*, pp. 1179–1186, 1992.

[30] J. H. Nie and D. A. Linkens, "A fuzzified CMAC self-learning controller," *Second IEEE International Conference on Fuzzy Systems,*, vol. 1, pp. 500–505, 1993.

[31] K. Zhang and F. Qian, "Fuzzy CMAC and its application," *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, pp. 944–947, 2000.

[32] C. C. Jou, “A fuzzy cerebellar model articulation controller,” *IEEE International conference on Fuzzy Systems*, pp. 1171–1178, 1992.

[33] D. Kim, “A design of CMAC-based fuzzy logic controller with fast learning and accurate approximation,” *Fuzzy Sets and Systems*, vol. 125, no. 1, pp. 93–104, 2002.

[34] S. H. Lane, D. A. Handelman, and J. J. Gelfand, “Theory and development of higher-order CMAC neural networks,” *IEEE Control System Magazine*, vol. 12, no. 2, pp. 23–30, 1992.

[35] H. Jiang, C. Quek, and G. S. Ng, “FCMAC-EWS: A bank failure early warning system based on novel localized pattern learning, and semantically associative fuzzy neural network,” *Expert Systems with Applications, in press*, 2007.

[36] C. Ting and C. Quek, “TSK-FCMAC: A novel fuzzy CMAC based on the zero-ordered TSK fuzzy inference scheme,” *under preparation*, 2007.

[37] J. Sim, W. L. Tung, and C. Quek, “FCMAC-Yager: A novel Yager inference scheme based fuzzy CMAC,” *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1394–1410, 2006.

[38] M. N. Nguyen, D. Shi, and C. Quek, “FCMAC-BYY: Fuzzy cmac using bayesian ying-yang learning,” *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, no. 5, pp. 1180–1190, 2006.

[39] C. Quek and Z. Guo, “FCM-AARS: A fuzzy cerebellar model network based on the approximate analogical reasoning schema,” *Pattern Recognition, under review*, 2007.

[40] J. F. Medina and M. D. Mauk, “Simulations of cerebellar motor learning: Computational analysis of plasticity at the mossy fiber to deep nucleus synapse,” *Journal of Neuroscience*, vol. 19, no. 16, pp. 7140–7151, 1999.

[41] P. M. Steele and M. D. Mauk, “Inhibitory control of LTP and LTD: Stability of synapse strength,” *Journal of Neurophysiology*, vol. 81, no. 4, pp. 1559–1566, 1999.

[42] J. A. Kleim, M. A. Pipitone, C. Czerlanis, and W. T. Greenough, “Structural stability within the lateral cerebellar nucleus of the rat following complex motor learning,” *Neurobiology of Learning and Memory*, vol. 69, pp. 290–306, 1998.

[43] J. S. Albus, “Mechanisms of planning and problem solving in the brain,” *Mathematical Biosciences*, vol. 45, pp. 247–293, 1979.

[44] Z. Q. Wang, J. L. Schiano, and M. Ginsberg, “Hash-coding in CMAC neural networks,” *Proc. IEEE International Conference on Neural Networks*, vol. 3, pp. 1698–1703, 1996.

[45] Z. Luo, Z. Zhao, and C. Zhu, “The unfavourable effects of hash coding on CMAC convergence and compensatory measure,” *Proc. IEEE International Conference on Intelligent Processing Systems*, pp. 419–422, 1997.

[46] Y. P. Hsu, K. S. Hwang, C. Y. Pao, and J. S. Wang;, “A new CMAC neural network architecture and its ASIC realization,” *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC2000)*, pp. 481–484, 2000.

[47] Y. P. Hsu, K. S. Hwang, and J. S. Wang;, “An associative architecture of CMAC for mobile robot motion control,” *Journal of Information Science And Engineering*, vol. 18, pp. 145–161, 2002.

[48] K. D. Federmeier, J. A. Kleim, and W. T. Greenough, “Learning-induces multiple synapse formation in rat cerebellar cortex,” *Neuroscience Letters*, vol. 332, pp. 180–184, 2002.

[49] W. T. Thach, “What is the role of the cerebellum in motor learning and cognition?” *Trends in Cognitive Sciences*, vol. 27, no. 9, pp. 331–337, 1998.

[50] F. A. Middleton and P. L. Strick, “Cerebellar output: motor and cognitive channels,” *Trends in Cognitive Sciences*, vol. 27, no. 9, pp. 348–354, 1998.

[51] E. R. Kandel, J. H. Schwartz, and T. M. Jessell, *Principles of Neural Science, 4th Edition*. McGraw-Hill, Health Professions Division., 2000.

[52] T. Tyrrell and D. Willshaw, “Cerebellar cortex: Its simulation and the relevance of Marr’s theory,” *Philosophical Transactions: Biological Sciences*, vol. 336, no. 1277, pp. 239–257, 1992.

[53] K. Doya, “What are the computations of the cerebellum, the basal ganglia and the cerebral cortex?” *Neural Networks*, vol. 12, pp. 961–974, 1999.

[54] J. Voogd and M. Glickstein, “The anatomy of the cerebellum,” *Trends in Cognitive Sciences*, vol. 2, no. 9, pp. 307–313, 1998.

[55] P. Strata and F. Rossi, “Plasticity of the olivocerebellar pathway,” *Trends in Neuroscience*, vol. 21, pp. 407–413, 1998.

[56] M. S. Salman, “The cerebellum: it’s about time! but timing is not everything–new insights into the role of the cerebellum in timing motor and cognitive tasks.” *Journal of Child Neurology*, vol. 17, no. 1, pp. 1–9, 2002.

[57] M. Rapoport, R. van Reekum, and H. Mayberg, “The role of the cerebellum in cognition and behavior,” *Journal of Neuropsychiatry and Clinical Neurosciences*, vol. 12, pp. 193–198, 2000.

[58] J. S. Albus, “A theory of cerebellar function,” *Mathematical Biosciences*, vol. 10, no. 1, pp. 25–61, 1971.

[59] D. Marr, “A theory of cerebellar cortex,” *Journal of Physiology of London*, vol. 202, pp. 437–470, 1969.

[60] M. Ito, *The Cerebellum and Neural Control*. New York, Raven Press, 1984.

[61] J. C. Houk, J. T. Buckingham, and A. G. Barto, “Models of the cerebellum and motor learning,” *Behavioral and Brain Sciences*, vol. 19, no. 3, pp. 368–383, 1996.

[62] M. Ito, “Mechanisms of motor learning in the cerebellum,” *Brain Research*, vol. 886, pp. 237–245, 2000.

[63] E. D. Schutter, “A new functional tole for cerebellar long term depression,” *Progress in Brain Research*, vol. 114, pp. 529–542, 1997.

[64] J. A. Kleim, E. Lussnig, E. R. Schwars, T. A. Comery, and W. T. Greenough, “Synaptogenesis and FOS expression in the motor cortex of the adult rat aftr motor skill learning,” *The Journal of Neuroscience*, vol. 16, no. 14, pp. 4529–4535, 1996.

[65] J. A. Kleim, R. A. Swain, K. A. Amstrong, R. M. A. Napper, T. A. Jones, and W. T. Greenough, “Selective synaptic plasticity within the cerebellar cortex following complex motor skill learning,” *Neurobiology of Learning and Memory*, vol. 69, pp. 274–289, 1998.

[66] J. S. Albus, *Brains, Behavior and Robotics*. BYTE Books, McGraw-Hill, 1981.

[67] S. D. Teddy, “State-of-the-art cmacs: A literature survey,” Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, Tech. Rep. C2i-TR-06/009, 2006.

[68] K. Ang and C. Quek, “Stock trading using PSEC and RSPOP: A novel evolving rough set-based neuro-fuzzy approach,” *Proc. IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1032–1039, 2005.

[69] T. Kohonen, *Self-Organization and Associative Memory (3rd ed.)*. Berlin, New York: Springer-Verlag, 1989.

[70] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[71] S. D. Teddy, “The MPSEC density-based clustering algorithm for non-linear memory assignment in the PSECMAC network,” Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, Tech. Rep. C2i-TR-06/006, 2006.

[72] S. J. Orfanidis, *Introduction to Signal Processing*. Prentice Hall, 1995.

[73] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. New Jersey: Prentice-Hall inc., 1985.

[74] S. D. Teddy, “The PSECMAC neural correlates,” Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, Tech. Rep. C2i-TR-06/007, 2006.

[75] C. W. Ting, "Learning convergence of tsk^0-fcmac: A novel fuzzy cmac based on the zero-ordered tsk fuzzy inference scheme," Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, Tech. Rep. C2i-TR-04/003, 2004.

[76] S. D. Teddy, "The PSECMAC learning convergence," Center for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore, Tech. Rep. C2i-TR-06/008, 2006.

[77] W. L. Tung and C. Quek, "GenSoFNN: A generic self-organizing fuzzy neural network," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1075–1086, 2002.

[78] K. K. Ang and C. Quek, "RSPOP: Rough set-based pseudo outer-product fuzzy rule identification algorithm," *Neural Computation*, vol. 17, no. 1, pp. 205–243, 2005.

[79] R. Kohavi, "The power of decision tables," in *Proceedings of the European Conference on Machine Learning*, ser. Lecture Notes in Artificial Intelligence 914, N. Lavrac and S. Wrobel, Eds. Berlin, Heidelberg, New York: Springer Verlag, 1995, pp. 174–189.

[80] The University of Waikato, "WEKA 3: Data mining software in java," *[Online] http://www.cs.waikato.ac.nz/ml/weka/*.

[81] D. M. Chance, *An Introduction to Derivatives & Risk Management*, 6th ed. Thomson, 2004.

[82] L. T. Nielsen, *Pricing and Hedging of Derivative Securities – Textbook in continuous-time finance theory*. Oxford University Press, 1999.

[83] F. Black and N. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, pp. 637–659, 1973.

[84] R. J. R. Jr. and B. J. Bartter, "Two-state option pricing," *Journal of Finance*, vol. 34, pp. 1093–1110, 1979.

[85] P. Radzikowski, "Non-parametric methods of option pricing," *Proc. of Informs-Korms (Seoul 2000 conference)*, pp. 474–480, 2000.

[86] H. Amilon, "A neural network versus black-scholes: A comparison of pricing and hedging performances," *Scandinavian Working Papers in Economics, Lund University series, Department of economics, Lund, Sweden*, 2001.

[87] U. Anders, O. Korn, and C. Schmitt, "Improving the pricing of options - a neural network approach," *Journal of Forecasting*, vol. 17, no. 5–6, pp. 369–388, 1998.

[88] M. Qi and G. S. Maddala, "Option-pricing using artificial neural networks: the case of s&p500 index call options," *Neural Networks in Financial Engineering*, pp. 78–92, 1995.

[89] C. Keber, "Option pricing with the genetic programming approach," *Journal of Computational Intelligence in Finance*, vol. 7, no. 6, pp. 26–36, 1999.

[90] Y. Ait-Sahalia and A. W. Lo, "Nonparametric estimation of state-price densities implicit in financial asset price," *LFE-1024-95, MIT-Sloan School of Management*, 1995.

[91] W. L. Tung and C. Quek, "GenSo-OPATS: A brain-inspired dynamically evolving option pricing model and arbitrage trading system," *Proc. IEEE CEC 2005, Edinburgh, Scotland*, vol. 3, pp. 2429–2436, 2005.

[92] U. Chicago Mercantile Exchange, *[Online] http://www.cme.com*.

[93] W. L. Tung and C. Quek, "GenSo-FDSS: a neural-fuzzy decision support system for pediatric all cancer subtype identification using gene expression data," *Artificial Intelligence in Medicine*, vol. 336, no. 1, pp. 61–88, 2005.

[94] J. Sinkey Jr., "A multivariate statistical analysis of the characteristics of problem banks," *Journal of Finance*, vol. 1, pp. 21–36, 1975.

[95] D. Martin, "Early warning of bank failure: A logit regression approach," *Journal of Banking and Finance*, vol. 1, no. 3, pp. 249–276, 1977.

[96] R. Cole and J. Gunther, "Separating the likelihood and timing of bank failure," *Journal of Banking and Finance*, vol. 19, no. 6, pp. 1073–1089, 1995.

[97] P. Y. K. Cheng, "Predicting bank failures: A comparison of the cox proportional hazards model and the time varying covariates model," Ph.D. dissertation, Nanyang Business School, Nanyang Technological University, Singapore, 2002.

[98] Federal Reserve Bank of Chicago, "Repository for bank data," *[Online] http://www.chicagofed.org*.

[99] W. L. Tung, C. Quek, and P. Y. K. Cheng, "GenSo-EWS: A novel neural-fuzzy based early warning system for predicting bank failures," *Neural Networks*, vol. 17, no. 4, pp. 567–587, 2004.

[100] F. M. Ashcroft and S. J. H. Ashcroft, *Insulin: Molecular Biology to Pathology*. Oxford University Press, 1992.

[101] Illinois Institute of Technology, "Glucosim: A web-based educational simulation package for glucose-insulin levels in the human body," *[Online] http://216.47.139.198/glucosim/gsimul.html*.

[102] Health Promotion Board Singapore, *[Online] http://www.hpb.gov.sg*.

[103] C2iWeb, "Centre for Computational Intelligence, School of Computer Engineering, Nanyang Technological University, Singapore," *[Online] http://www.c2i.ntu.edu.sg*.

[104] S. D. Teddy, E. M.-K. Lai, and C. Quek, "Hierarchically clustered adaptive quantization CMAC and its learning convergence," *IEEE Trans. Neural Networks, to appear*, 2007.

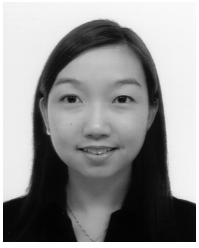
[105] W. L. Tung and C. Quek, "Falcon: Neuro fuzzy control and decision systems using FKP and PFKP clustering algorithms," *IEEE Trans. Syst., Man, Cybern. Part B, Cybern.*, vol. 34, no. 1, pp. 686–695, 2004.

[106] R. W. Zhou and C. Quek, "Antiforgery: A novel pseudo-outer product based fuzzy neural network driven signature verification system," *Pattern Recognition Letters*, vol. 230, no. 14, pp. 1795–1816, 2002.

[107] C. Quek, B. Tan, and V. Sagar, "POPFNN-based fingerprint verification system," *Neural Networks*, vol. 14, pp. 305–323, 2001.

[108] K. Quah and C. Quek, "Maximum reward reinforcement learning: A non-cumulative reward criterion," *Expert Systems with Applications*, vol. 31, no. 2, pp. 351–359, 2006.

[109] K. K. Ang and C. Quek, "Stock trading using RSPOP: A novel rough set based neuro-fuzzy approach," *IEEE Trans. Neural Networks*, vol. 17, no. 5, pp. 1301–1315, 2006.



S. D. Teddy received the B.Eng. degree (First Class Honors) in computer engineering from Nanyang Technological University, Singapore, in 2003, where she subsequently pursued her doctorate degree at the Centre for Computational Intelligence, School of Computer Engineering. She is currently a research engineer with the Data Mining Department of the Institute for Infocomm Research, Singapore. Her current research interests include the cerebellum and its computational model, artificial neural networks, the study of brain-inspired learning memory systems, computational finance, and autonomous control of bio-physiological processes.

C. Quek received the B.Sc. degree in electrical and electronics engineering and the Ph.D. degree in intelligent control from Heriot-Watt University, Edinburgh, Scotland, UK, in 1986 and 1990 respectively. He is an associate professor and a member of the Centre for Computational Intelligence, formerly the Intelligent Systems Laboratory, School of Computer Engineering, Nanyang Technological University. His research interests include intelligent control, neuro-cognitive architectures, AI in education, neural networks, fuzzy systems, fuzzy rule-based systems, and genetic algorithms and brain-inspired neuro-cognitive applications in Computational Finance and Biomedical Engineering.

E. M.-K. Lai (M’82-SM’95) received the B.E.(Hons) and PhD degrees in 1982 and 1991 respectively from the University of Western Australia, both in electrical engineering. He is currently a faculty member of the Institute of Information Sciences and Technology, Massey University at Wellington, New Zealand. Previously he has been a faculty member of the Department of Electrical and Electronic Engineering, The University of Western Australia from 1985 to 1990, the Department of Information Engineering, the Chinese University of Hong Kong from 1990 to 1995, Edith Cowan University in Perth from 1995 to 1998 and the School of Computer Engineering, Nanyang Technological University in Singapore from 1999 to 2006. His current research interests include artificial neural networks, digital signal processing and information theory.