Decentralized Asynchronous Learning in Cellular Neural Networks

Bipul Luitel, *Member, IEEE*, Ganesh K Venayagamoorthy, *Senior Member, IEEE*

***Abstract*—Cellular neural networks (CNNs) described in literature so far consist of identical units called cells connected to their adjacent neighbors. These cells interact with each other in order to fulfill a common goal. The current methods involved in learning of CNNs are usually centralized (cells are trained in one location) and synchronous (all cells are trained simultaneously either sequentially or in parallel depending on the available hardware/software platform). In this paper, a generic architecture of CNN is presented and a special case of supervised learning has been demonstrated explaining the internal components of a cell. A decentralized asynchronous learning (DAL) framework for CNN is developed in which each cell of the CNN learns in a spatially and temporally distributed environment. An application of DAL framework is demonstrated by developing a CNN based wide area monitoring system for power systems. The results obtained are compared against equivalent traditional methods and shown to be better in terms of accuracy and speed.**

***Index Terms*—CNN, Decentralized asynchronous learning, high performance computer, multilayer perceptron, power systems, PSO, SRN, wide area monitoring**

I. Introduction

Two major variations of cellular neural networks (CNNs) have been studied in the neural networks community. CNN introduced by Chua and Yang in 1988 [1] consists of individual units (cells) connected to each of its neighbors on a cellular structure. Each cell of such a CNN is a computational unit and have been applied to pattern recognition [2], [3] and image processing [4], [5]. CNN is a highly non-linear system and its stability is important for real applications. Multistability of such CNNs is discussed in [6]. In [7], Werbos introduced a cellular implementation of simultaneous recurrent neural networks (SRNs), where each 'cell' is an SRN with same set of weights but different set of inputs. Such a CNN consisting of SRNs as cells are called Cellular SRN (CSRN) and that containing multilayer perceptron (MLP) as cells are called Cellular MLP (CMLP). CSRNs have been used in maze navigation problem [8], [9], facial recognition [10] and image processing [11]. Stability of recurrent neural networks in the presence of noise and time delays is discussed in [12]. Thus, [6] and [12] together provide a basis for stability of such CNNs containing neural networks in each of its cells. In the original CNN, each cell is connected only to its adjacent cells [1]. However, in CMLP and CSRN, the connection of different cells to each other is application dependent, as is shown in application to bus voltage prediction in a power system [13]. However, even with variations, most of the CNNs studied so far consist of identical units in each cell of the CNN and learning is centralized and synchronous - neural networks (NN) in each cell of the CSRN or CMLP are trained simultaneously in one location.

Distributed learning of artificial systems has been of interest for a long time in the research community. Many approaches have focused on either data decomposition or task decomposition methods to achieve parallelism by distributing among multiple processors [14]–[18]. Use of distributed learning in computational intelligence (CI) and machine learning (ML) paradigms has also been reported in literature [14], [19], [20]. Although distributed learning methods capture the essence of decentralized computing by reducing the volume of information shared by performing local computations at different 'nodes', the approaches either consist of a 'master' making decisions based on information from the rest of the nodes in the network [21], or the nodes being centrally located in one place and synchronized by a global clock. Learning may be carried out sequentially or in parallel by exploiting their inherent parallelism using a parallel computing platform. However, all of the nodes or cells are updated simultaneously for any change in the system and hence learning is not independent among the cells. As such, most current approaches, even though distributed, carry out centralized synchronous learning regardless of the hardware/software platform used for implementing them.

The major contributions of this paper are as follows:

1) A generic framework of CNN is presented and a special case of supervised learning has been demonstrated.
2) A decentralized asynchronous learning (DAL) framework for CNN has been developed and implemented on a heterogeneous CNN.
3) CNN with DAL framework has been implemented as a wide area monitoring system (WAMS) for power systems.
4) It is shown that multiple neural networks of different cells of a CNN can each, concurrently, learn information embedded in data obtained from a complex system.

The remaining sections of the paper are arranged as follows: Architecture of CNN is presented in Section II. Learning of learning systems is explained in Section III. Development of proposed DAL for heterogeneous CNN is explained in Section IV. Development of WAMS based on CNN with DAL is presented in Section V. Case studies with results and

Bipul Luitel and Ganesh K Venayagamoorthy are with Real-Time Power and Intelligent Systems Laboratory, Department of Electrical and Computer Engineering, Clemson University, Clemson, SC, 29634 USA. Contact: iambipul@ieee.org, gkumar@ieee.org

The funding provided by the National Science Foundation, USA under the CAREER grant ECCS #1231820 and EFRI #1238097 is gratefully acknowledged.

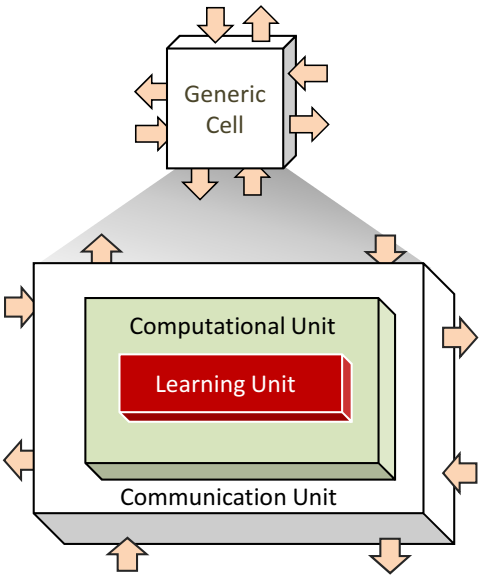

Fig. 1. Internal units of a generic cell in a CNN.

discussion are presented in Section VI, and conclusions are given in Section VII.

II. Cellular Neural Networks

A generic architecture of a CNN consists of cells connected to each other in some fashion. In this study, each cell consists of a computational unit, a learning unit and a communication unit. This is shown in Fig. 1. The computational unit/element can be appropriately chosen for different applications, but some variation of one of many CI paradigms will be more suitable. The purpose of the computational element is to utilize the information available to it (either directly or through its interaction with the neighbors) to produce an output in order to improve the performance over time. The ability to utilize gathered information for improving one's performance is also known as experience and is facilitated by the learning unit. The learning unit can have supervised, unsupervised or reinforcement based learning and provides a measure for evaluation of performance based on which a cell can improve itself. Since the individuals interact with their neighbors in a collaborative learning system (where neighborhood may be defined based on certain parameters which are application dependent, for example electrical distance in the application shown in this paper), a communication unit is also present in the cell. This unit consist of input/output interface for sending to and receiving from other cells in the network. Communication takes place according to a predefined rule with as many neighbors as is required by the application.

CNN architecture and learning can be classified as follows:

1) $Centralized$ vs. $decentralized$: A CNN formation is said to be $centralized$ if all the cells are located in one physical location. It is said to be $decentralized$ if cells are spatially distributed across different physical locations (like in grid computing).
2) $Homogeneous$ vs. $heterogeneous$: A CNN structure is said to be $homogeneous$ if all the cells have identical architecture, size and learning method. It is said to be $heterogeneous$ if one or more of the cells have different architecture, size or learning method.
3) $Synchronous$ vs. $asynchronous$: A CNN learning is said to be $synchronous$ if adaptation takes place concurrently on all the cells. It is said to be $asynchronous$ if individual cells learn and adapt at different times and do not depend on a global or a 'master' clock for the learning.
4) $Sequential$ vs. $parallel$: A CNN implementation is said to be $sequential$ if it takes place in a sequential computing environment using a sequential algorithm. It is said to be $parallel$ if it takes place in a distributed environment using parallel computing on suitable hardware/software platform.

Fig. 2 shows different variations of CNN in terms of formation, structure, learning and implementation. Any combination of these variants can form a different type of CNN. For example, CNN developed for one of the case studies in this paper (Test System II) has a decentralized formation, heterogeneous structure, heterogeneous learning method, asynchronous adaptation and is implemented in parallel.

Sometimes 'decentralized' and 'distributed' are interchangeably used in literature. The word 'distributed' is, however, contextual and can mean task, data or temporal distribution in parallel computing environment, or spatial distribution. For clarity, the word 'decentralized' henceforth is used explicitly to imply spatial distribution and 'distributed' is used in the context of parallel computing. Synchronized, centralized and sequential learning techniques for individual or networked (cellular or population based) systems have been studied and applied to a wide variety of problems using CI paradigms. With the development of advanced hardware/software platforms and parallel computing tools, distributed learning methods have also been developed and applied. However, to the best of authors' knowledge, there has not been any substantial study in the CI or ML community in the area of decentralized learning of networked, let alone heterogeneous, learning systems. The challenge here is that each individual or member of the system is a learning sub-system in itself and is directly and/or indirectly related to the knowledge and experience of the member(s) in its surrounding. In most real-life problems where each cell implements a nonlinear function, the change in the input is nonlinear which results in a nonlinear change in the output. This complex and iterative information dependency, nonlinear input/output behavior combined with the ability to learn asynchronously poses even greater challenge in the learning of such systems. This is known as learning of learning systems (LOLS).

III. Learning of Learning Systems

Learning of learning systems is a social behavior of swarms where each individual learns at different pace, at different times and in different environment while still interacting with the other individuals of the society. This behavior is comparable to students performing certain projects in a virtual classroom in which distant students learn at their own pace, in

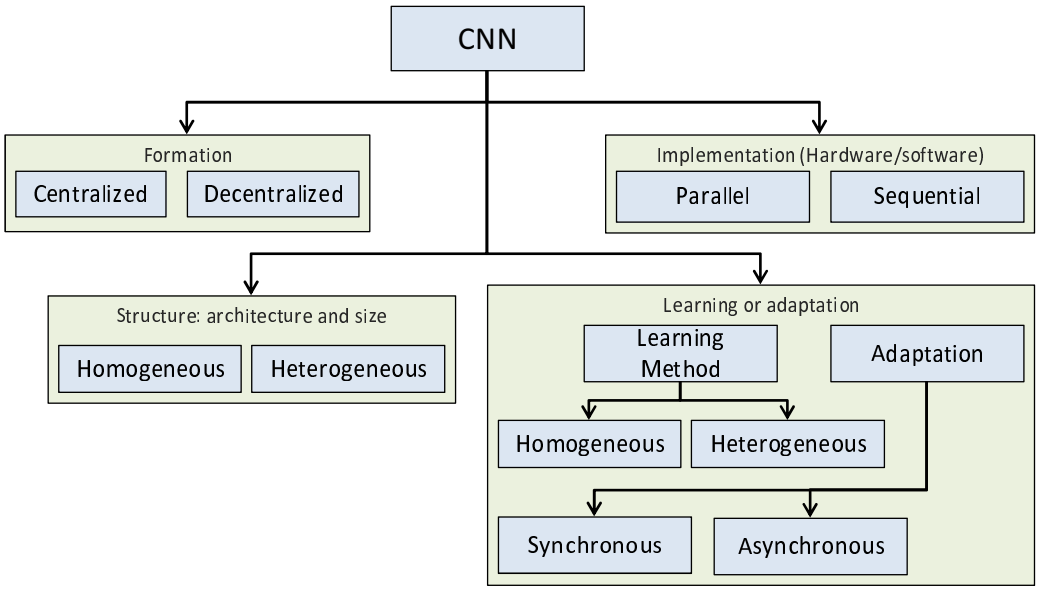

Fig. 2. CNN variants - structure and implementation.

their own environment at different times while still working on a common objective. For the same reason, decentralized learning has developed as an instructional paradigm in the field of distance education where both synchronous as well as asynchronous learning methods are utilized [22]. In a collaborative distributed learning [23], learning occurs through communication and collaborative interactions. In [24], Salomon and Perkins have pointed out that learning involves both cognitive as well as social aspects. They have also pointed that a social entity can itself be identified as a learning system where collective outcome of the system as a whole is more important than one or more individuals' output. Learning in an artificial system having spatially distributed interacting individuals is the major focus of LOLS.

Consider a generic system consisting of different interconnected subsystems as shown in Fig. 3. The output of any subsystem connected to N neighboring subsystems as part of the LOLS can be represented by (1):

$$O_{ss_i}(k) = f\left(\alpha_i O_{SS_i}(k-1), \alpha_n^1 O_{SS_n^1}(k-1), \ldots, \alpha_n^N O_{SS_n^N}(k-1), KS_i, KD_i\right) \quad (1)$$

where α is the discount factor associated with each subsystem. For any subsystem, the discount factor affects the amount of influence of its own past experience or the knowledge of its neighbors in its future decisions. The output of each subsystem might also be governed by other static and dynamic parameters associated with the subsystem, which are represented as KS_i and KD_i in the equation. In the proposed DAL framework for CNN, each cell learns in its own location based on its own inputs and the information obtained from its connectivity with the neighbors. The objective of the learning is to improve the overall output of the whole system and not just one or few individuals of the group, which closely resembles the cognitive and social learning aspects of swarms in a biological

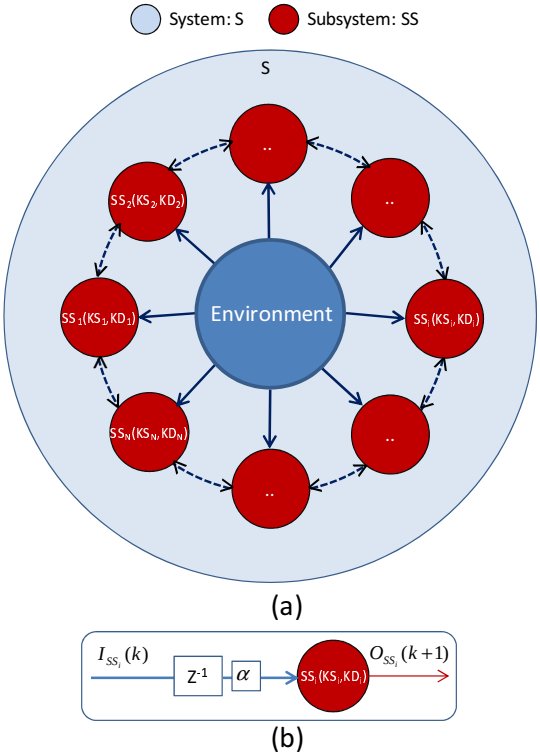

Fig. 3. (a) A system of interconnected subsystems in the context of LOLS. (b)Each subsystem in a LOLS showing its output as a function of its inputs.

ecosystem. The learning is decentralized because the individuals (cells) can be spatially distributed and communicate certain information with only a few neighbors of interest. The learning is asynchronous because it can be temporally distributed among the individuals (cells). That means, in decentralized asynchronous learning in a heterogeneous CNN, different types of cells can learn at different times as deemed necessary depending on availability of information and deviation from its

current output from the desired. At each location, the learning can be either sequential or parallel. Thus, a DAL framework for CNN also represents a learning of learning systems.

IV. Decentralized Asynchronous Learning

When the size of the network grows, or the parameters of one or more components of the network change, then the input/output behavior of the overall network changes. That means, in a centralized CNN implementation of a networked system, any change in one or more nodes in the system will require a change in the architecture of the entire CNN representing the network. Also, a synchronized learning mechanism will require all of the nodes to update themselves simultaneously to learn the new change introduced in one or more parts of the network, although not all of the cells may be affected by the change. At the hardware level, this translates into more processing, memory and power consumption. Therefore, a DAL framework for CNN is developed which allows for dynamic changes to the network topology without affecting the operation of the CNN. The learning unit in each cell quickly captures any change in the network topology or parameters affecting the cell and updates the computational unit to reflect the changes. This takes place at each affected cell asynchronously. There are both cognitive and social aspects of learning in DAL. The cognitive learning takes place when parameters directly affecting the cell change and the cell has to update itself to reflect the change. This new acquired knowledge is then transferred to the other members of the network (neighboring cells) through the communication unit. As a result, the neighbors observe a change in the behavior, and update themselves. The new knowledge acquired by one cell thus propagates through the network which results in social learning. In Fig. 4, propagation of knowledge across the network is shown with a typical example of four cells. Each column represents the change beginning at four different cells while the rows represent the propagation of the change across the network over time (samples). It shows how the CNN transitions from one state to the other asynchronously when a change occurs at any one of the cells. If the change in the input causes significant change in the output, then the learning unit will activate. This happens at every cell in the network thus leading to a ripple of changes. This leads the network to a new state of knowledge whenever a change is introduced in the network. Therefore, DAL framework supports addition of cells to the network. A centralized implementation is vulnerable to attacks but DAL framework makes a CNN fault-tolerant and scalable.

Implementation of DAL in a cell is explained in the following paragraphs with reference to the structure of a cell shown in Fig. 5. The following terms are introduced with respect to DAL:

- $Dynamic\ database$ - A buffer of historical data (inputs and target values) used by the learning unit to train the NN.
- $Window$ - The interval at which the $Database$ is updated with new data.
- $Dynamic\ database\ buffer$ - A buffer of data received by the communication unit during the $Window$. This data is used to update the $Dynamic\ database$ after each $Window$. The size of this buffer determines how often the historical data is updated.
- $SendList$ - A list of neighbors to which each cell has to send its output.
- $RecvList$ - A list of neighbors from which each cell has to receive its input.

Each cell maintains its own $Dynamic\ database$ which is updated after each $Window$ with new data received within that period and stored in $Dyanamic\ database\ buffer$. The contents of the $Dynamic\ database$ are shifted by the $Window$ size and the remaining data is filled by the content of the $Dynamic\ database\ buffer$. This way the historical data is kept current. After a step-ahead prediction is calculated at the output $O(\hat{k})$ from its input at instant k , $(I(k))$, it is compared with its target from previous time instant $k-1$, which is $O(k)$, and an absolute relative error (ARE) is calculated according

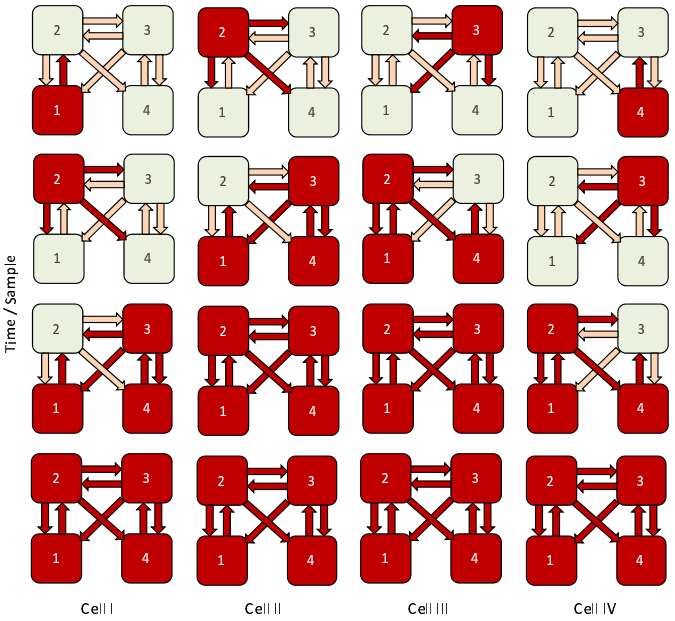

Fig. 4. Asynchronous propagation of knowledge across the network in a DAL framework.

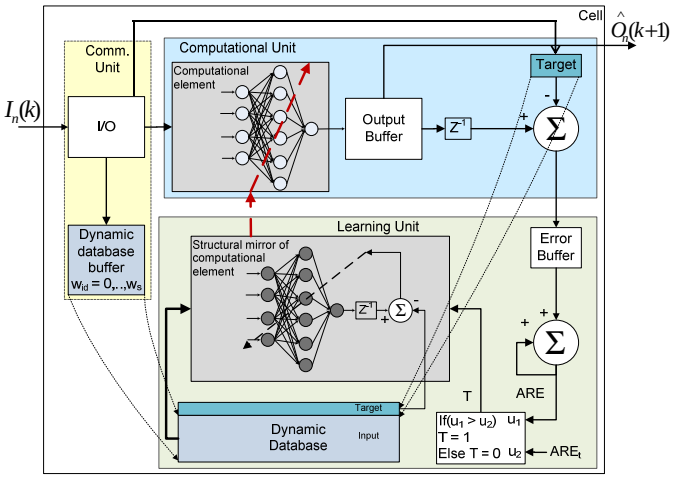

Fig. 5. Implementation of a cell for asynchronous learning.

to (2) as the measure of fitness of learning. The ARE may be taken over one or more steps, in which case it is stored on the error buffer and accumulated in order to take an average. If the error is greater than a predetermined threshold ARE_t, then a trigger (T) is activated according to (3). This triggers the learning unit and the parameters of the computational element(weights of NN if the computational element is a NN) are adjusted according to the learning algorithm. Thus an asynchronous online learning takes place at each cell. DAL framework in Fig. 5 presents a scenario for supervised learning where a predefined target is required. Therefore, "Target" shown in the figure is a buffer for holding the target values $O(k)$. In the application described in this paper, the target is generator speed deviation. Since generator speed deviation is also one of the inputs to the cell, the target is a step-ahead (time shifted) values of that input. The learning in each cell of a CNN with DAL is shown in flowchart of Fig. 6.

$$ARE_n(k) = \frac{\|O_n(k) - \hat{O}_n(k)\|}{\|O_n(k)\|} \tag{2}$$

$$T = \begin{cases} 1 & \text{If } ARE > ARE_t \\ 0 & \text{Otherwise} \end{cases} \tag{3}$$

V. Development of WAMS based on CNN with DAL

Power system consists of components and phenomena associated with them that have complex dynamic behavior. Such components such as generators and phenomena such as power flows can be represented by differential algebraic equations (DAEs) [25]. Since neural networks can be effectively used to learn the output of such DAEs, they can be a very suitable tool for modeling and monitoring the behavior of power system components and phenomena. A WAMS is used to assess the status of various components of a power system for providing predictive control. In future power systems (smart grids), the number of such components will be very large and hence it becomes challenging to perform accurate predictive state estimation using WAMS in a reasonable amount of time [26]. Therefore, a WAMS based on CNN with DAL is proposed.

The proposed CNN consists of neural networks in each cell and closely relates to the Object Net approach described in [27], [28]. These cells may be homogeneous having similar types and sizes of neural networks or heterogeneous depending on the type of application. Each cell of the CNN is used to predict the speed deviation of one generator in the power system. The architecture of CNN is developed such that it exactly represents the physical structure of the power system. This is done in two phases. In Phase I, the cells are connected to each other based on 'nearest-n neighbors' topology, which means previous sample outputs of n nearest neighbors of each cell are connected to the inputs of that cell. The number n should be picked such that it is less than the total number of cells (N), but also ensuring that the connectivity of the network is not lost. This is better understood by referring to the applications presented in the paper. In the proposed application, the "nearness" of components is defined as the electrical distance between the generators and is measured based on the length of the transmission lines separating the two generators. In this study, two nearest neighbors ($n = 2$) are considered for designing the CNN. For example in the Test System I (Fig. 7), two nearest neighbors of generator $G1$ are generators $G2$ and $G4$. This is represented in the CNN by connecting the outputs of the cells $C2$ and $C4$ to the inputs of the cell $C1$. Similarly for $G4$, two nearest neighbors are $G2$ and $G3$ and hence outputs of the cells $C2$ and $C3$ are connected to the inputs of the cell $C4$. This topology allows for the scalability of the CNN by keeping the size of the NN in each cell to a minimum. However, for Test System II (Fig. 8(a)), a complete connectivity of the system is not obtained by only considering nearest-2 neighbors between the generators, as is shown in Fig. 8(b). To avoid forming islands on a large power system, nearest generators in adjacent islands are connected in Phase II in order to obtain a CNN structure as shown in Fig. 8(c). Due to the extra connections in some

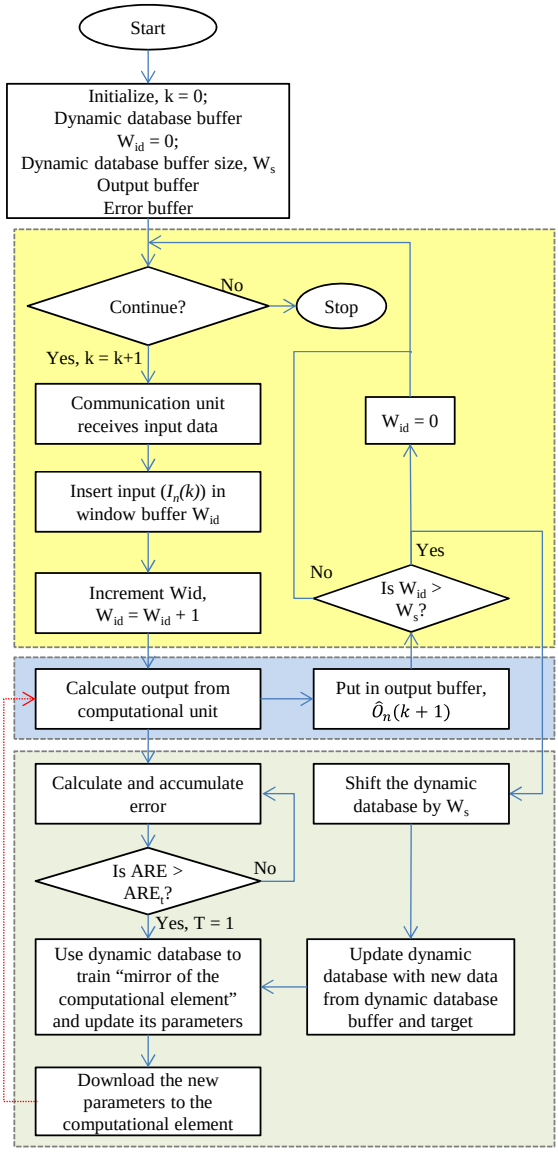

Fig. 6. Flowchart illustrating learning of a cell (Fig. 5) in a CNN.

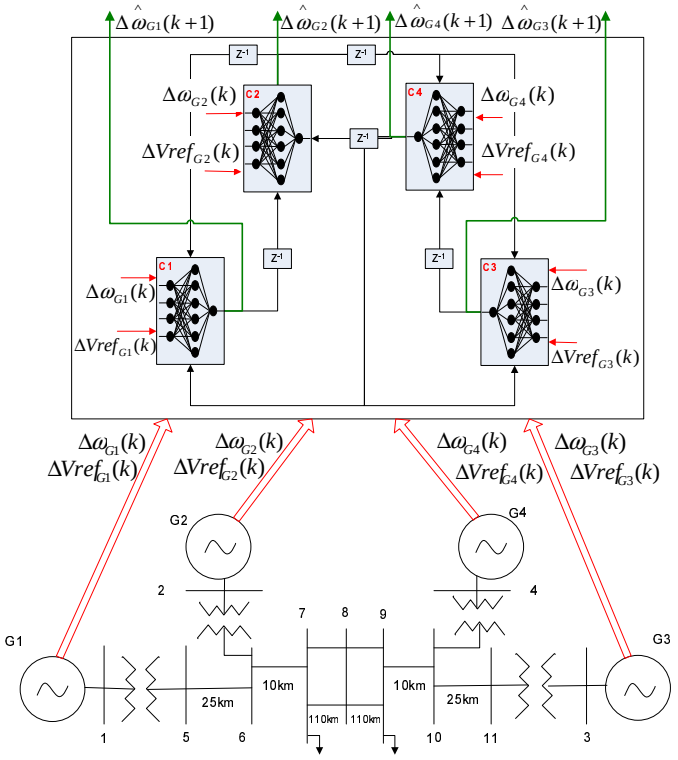

Fig. 7. CNN based WAM for a two-area four-machine system (Test System I).

of the cells due to Phase II connection, a heterogeneous CNN is developed. In this paper, a WAMS based on CNN with DAL is developed to predict the speed deviations ($\Delta\hat{\omega}$) of each generator in the multimachine power system at instant $k+1$ based on speed deviations ($\Delta\omega$) and deviation of the reference voltage ($\Delta Vref$) (shown in Fig. 9) of the generators at instant k as the inputs.

For Test System I, the computational unit in each cell is an MLP. Each MLP is a time-delayed neural network (TDNN) with an input layer of six neurons, a hidden layer of eight neurons and an output layer of a single neuron. Each cell consists of six inputs viz. actual reference voltage applied to the generator $\Delta Vref(k)$, current and time-delayed values of the actual speed deviation of the generator $\Delta\omega(k), \Delta\omega(k-1), \Delta\omega(k-2)$ and the predicted speed deviations of the nearest two generators, $\Delta\hat{\omega_{n1}}(k)$ and $\Delta\hat{\omega_{n2}}(k)$. For Test System II, the cells connected to two nearest neighbors have their computation unit implemented using MLPs where as those with more than two nearest neighbors (cells participating in inter-island connectivity of Phase II) are implemented using SRNs in order to capture the new dynamics. The number of neurons in the input, hidden and context layer of SRN is dependent on the number of neighbors each cell is connected to. Unlike MLPs, the inputs to the SRN consist of only current values of the the inputs i.e. $\Delta Vref(k)$, $\Delta\omega(k)$, $\Delta\hat{\omega_{n1}}(k)$, $\Delta\hat{\omega_{n2}}(k)$ and $\Delta\hat{\omega_{n3}}(k)$. The computational unit in each cell of the CNN is a neural network and is independently trained by the learning unit. The CNN presented here is heterogeneous in terms of both computational as well as learning unit. In this study, the learning unit of the cells whose computational unit

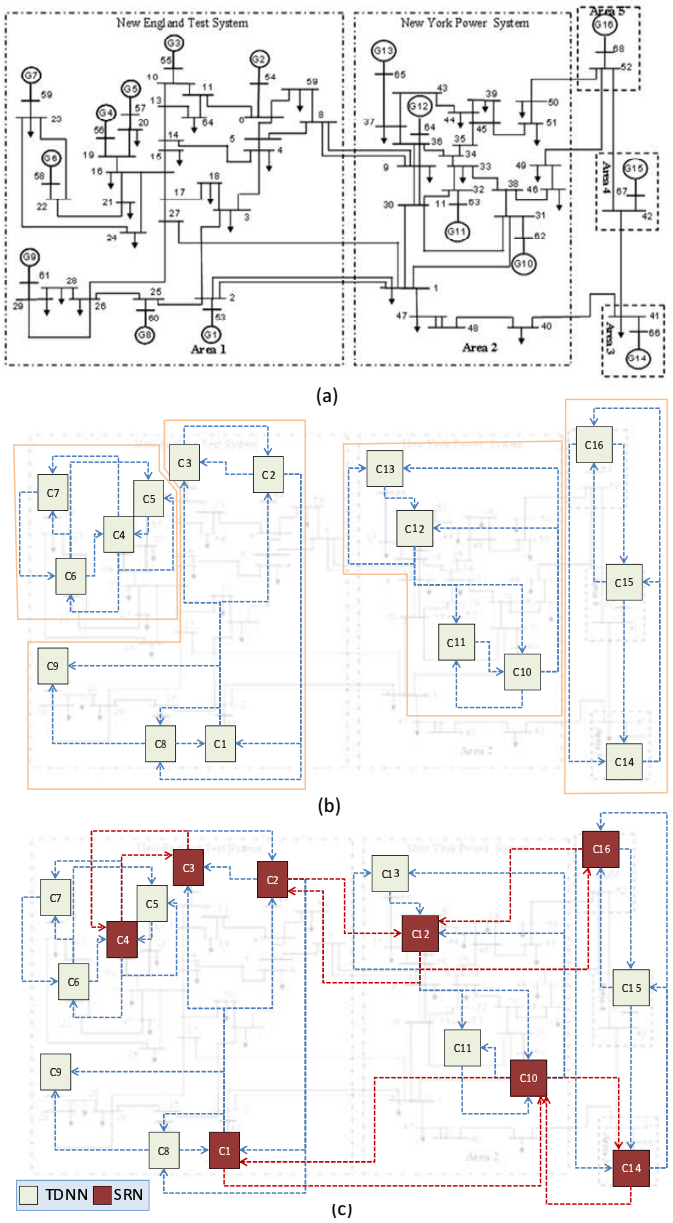

Fig. 8. (a) New York-New England 16-generator 68-bus system used as Test System II. (b) CNN implementation in Phase I shown on top of a faint background of the Test System. (c) Implementation in Phase II showing complete connectivity through heterogeneous CNN.

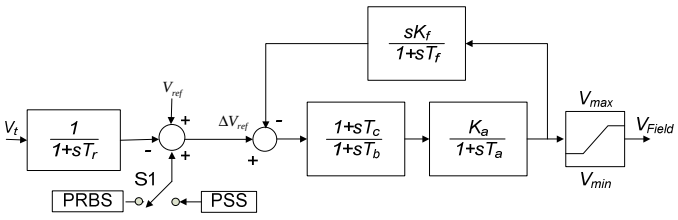

Fig. 9. Generator excitation system (showing $\Delta Vref$).

is an MLP uses backpropagation (BP) and that of cells whose computational unit is an SRN uses particle swarm optimization (PSO) [29] to train it. PSO has been shown to be effective in training of neural networks [29]–[31]. Since training of SRNs is a challenging task, PSO has been used to train them. For every sample of the input data $I(k)$, each cell produces a step-ahead predicted output $O(k+1)$. Therefore, for any input data of size $1, 2, \ldots, k, \ldots, K$ discrete samples, and W_n and V_n be the input and output weight matrices respectively of the SRN in n^{th} cell, then the output of each cell is given by (4).

$$\begin{aligned} O_n(k) &= \Delta\hat{\omega}_n(k) \\ &= f\left(I_n(k-1), W_n(k), V_n(k)\right) \end{aligned} \tag{4}$$

Considering the case of two nearest neighbors, the input vector for any cell will be:

$$\begin{aligned} I_n(k) &= [\Delta Vref_n(k), \Delta\omega_n(k), \Delta\omega_n(k-1), \\ &\quad \Delta\omega_n(k-2), \Delta\hat{\omega}_{n1}(k), \Delta\hat{\omega}_{n2}(k)] \end{aligned} \tag{5}$$

The predicted speed deviations can be written as:

$$\begin{aligned} \Delta\hat{\omega}_1(k+1) &= f\left(W_1(k), V_1(k), \Delta Vref_1(k), \right. \\ &\quad \Delta\omega_1(k-2), \Delta\omega_1(k-1), \Delta\omega_1(k), \\ &\quad \left. \Delta\hat{\omega}_2(k), \Delta\hat{\omega}_4(k)\right) \end{aligned} \tag{6}$$

$$\begin{aligned} \Delta\hat{\omega}_2(k+1) &= f\left(W_2(k), V_2(k), \Delta Vref_2(k), \right. \\ &\quad \Delta\omega_2(k-2), \Delta\omega_2(k-1), \Delta\omega_2(k), \\ &\quad \left. \Delta\hat{\omega}_1(k), \Delta\hat{\omega}_4(k)\right) \end{aligned} \tag{7}$$

$$\begin{aligned} \Delta\hat{\omega}_3(k+1) &= f\left(W_3(k), V_3(k), \Delta Vref_3(k), \right. \\ &\quad \Delta\omega_3(k-2), \Delta\omega_3(k-1), \Delta\omega_3(k), \\ &\quad \left. \Delta\hat{\omega}_2(k), \Delta\hat{\omega}_4(k)\right) \end{aligned} \tag{8}$$

$$\begin{aligned} \Delta\hat{\omega}_4(k+1) &= f\left(W_4(k), V_4(k), \Delta Vref_4(k), \right. \\ &\quad \Delta\omega_4(k-2), \Delta\omega_4(k-1), \Delta\omega_4(k), \\ &\quad \left. \Delta\hat{\omega}_2(k), \Delta\hat{\omega}_3(k)\right) \end{aligned} \tag{9}$$

In order to simulate a decentralized operation, the CNN is implemented on a parallel computer where each cell resides on a separate processor and communicates to its neighbors (other processors) using message passing interface (MPI). Each cell maintains a $SendList$ and a $RecvList$ and uses MPI_send and MPI_recv commands to send/receive the data between the neighbors. For a CNN with N cells, $SendList$ and $RecvList$ are N length vectors with a value of '1' in the column representing its nearest neighbors or a value of '0' otherwise. For Test System I, the $SendList$ and $RecvList$ are shown in Table I. The two matrices will be transpose of each other. The communication protocol for each cell is given in the pseudocode.

for n = 1 to N **do**
  if $SendList(n)$ == 1 **then**
    Send output to $Cell_n$ using MPI_send.
  end if
  if $RecvList(n)$ == 1 **then**
    Receive input from $Cell_n$ using MPI_recv.
  end if
end for

VI. Results and Discussions

A. Test Systems I and II

A stream of data used for this study is obtained by simulation of the test systems in real-time digital simulator (RTDS) [32]. Since RTDS operates in real-time, it gives a more realistic representation of an actual system. A pseudo-random binary signal (PRBS) applied to the excitation system of the generator causes a change in the reference voltage $Vref$ which causes the speed deviation in the generators. Perturbation is increased on generators each time and 10 seconds of data at 100 Hz collected for each PRBS applied to additional generator. Thus, a 40 seconds of data is used as the input stream to the CNN. Output of each cell is a step ahead prediction of the input data corresponding to each generator of the test system. The output of the CNN plotted against the actual signal is shown in Fig.10. The proposed DAL framework for CNN consists of many variables that can be customized based on applications. These variables are parameters associated with learning methods (eg. learning gain, momentum gain, iterations, population size, acceleration constants, error threshold), computational units (eg. number or neurons in different layers) and the DAL framework (eg. neighborhood size, database and buffer size). Table II lists those variables/parameters and their values used in the application described in the paper. The values can be considered as a starting point in any future applications but are not claimed as optimal.

Test System II is represented by a CNN consisting of 16 cells each predicting the speed deviation of one generator in the system. The output of the CNN are shown in Fig.11.

B. Discussion

1) Asynchronous learning: Asynchronous behavior in CNN learning is as a result of learning unit trigger shown in (3). The cell undergoes weight updates using the historical data from $Dynamic\ database$ only when $T = 1$. Fig. 12 is the plot of the trigger vector 'T' over the training duration. The plot shows the asynchronous learning during implementation of Test System I where different cells are learning at different times depending on their performance. The x-axis shows the

TABLE I
$SendList$ AND $RecvList$ OF FOUR CELLS FOR TEST SYSTEM I

	$SendList$				$RecvList$			
	I	II	III	IV	I	II	III	IV
Cell I	0	1	0	0	0	1	0	1
Cell II	1	0	1	1	1	0	0	1
Cell III	0	0	0	1	0	1	0	1
Cell IV	1	1	1	0	0	1	1	0

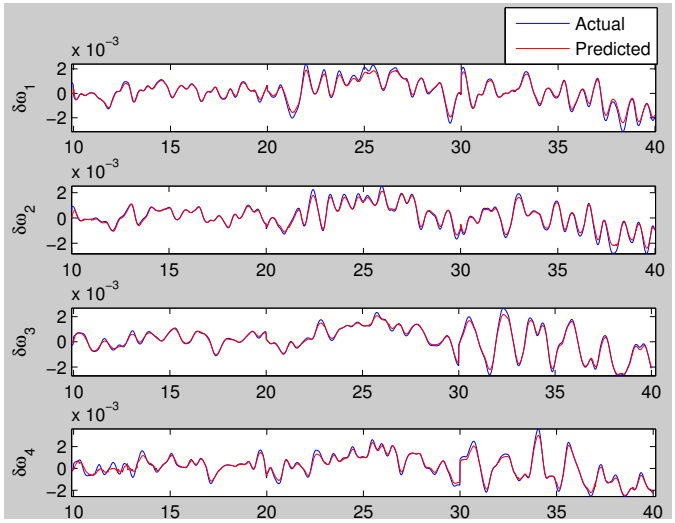

Fig. 10. Output of the CNN for Test System I.

TABLE II
PARAMETERS USED FOR TRAINING A CELL OF CNN.

Learning Gain	0.005
Momentum Gain	0.001
Number of PSO iterations	1000
Cognitive acceleration constant (c_1)	2
Social acceleration constant (c_2)	2
Momentum (w)	(Linearly) 0.9 to 0.4
Database size	1000
Window size	100
Error threshold (ARE_t)	5%
Case I: n-neighbors	2
CaseI: Learning method	BP
MLP (Test System I)	6x8x1
Case II: n-neighbors	2, 3, 4
MLP (Test System II)	6x8x1
SRN (Test System II, Cell 1,2,3,4,14,16)	13x8x1
SRN (Test System II, Cell 10, 12)	14x8x1
$Dynamic\ database\ buffer$	100
$Dynamic\ database$	1000

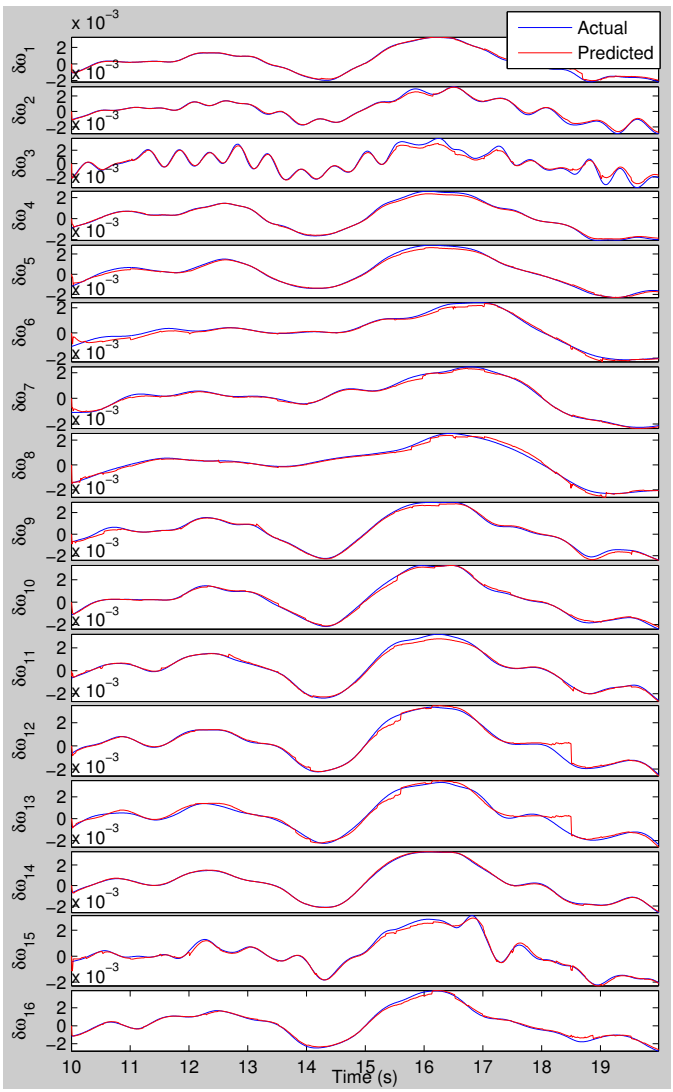

Fig. 11. Output of the CNN for generators in Test System II.

progress in time. For each cell shown on each row in y-axis, the shaded part in x-axis corresponds to the time when each cell learns ($T = 1$) and the white spaces represent the times when it does not (because of its predictions being within the expected threshold, $T = 0$). Similarly, Fig.13 shows the asynchronous learning of the cells of Test System II in DAL framework. The MLP and SRN cells are distinguished by different shades, and rows 1, 2, 3, 4, 10, 12, 14 and 16 correspond to the SRN learning. For Test System I, asynchronous learning takes **271** seconds as opposed to **19** minutes for synchronous learning. Average mean squared errors (MSEs) between the actual and the predicted outputs of all the cells during testing on CNNs trained using synchronous (asynchronous) learning is 19.4 (12.97). The result is rather surprising and implies that updating the neural network weights on every sample of input is not the ideal way to train it, and may in fact deteriorate its performance in a connected network like the CNN. While the observations show obvious benefits of asynchronous learning in terms of speed, the paper does not claim that a threshold of 5% for triggering the learning unit is optimal. It requires more in-depth research into the weight update pattern with different thresholds and comparison of its performance in training as well as testing to come to a fair conclusion regarding speed/accuracy tradeoff in asynchronous learning.

2) Knowledge retention: The CNN trained using DAL framework is used to test the simulation data obtained from a 10-cycle 3-phase to ground fault on bus 8 of Test System I. Then a Prony analysis [33] is carried out on the predicted outputs of the CNN as well as the actual data. In Table III, the natural frequencies (ω_N) and damping ratios (ζ) obtained from the Prony analysis of the actual and the predicted signals are presented along with errors ($E\omega_N$ and $E\zeta$) between the actual and the predicted. The results show that the dominant inter-area and intra-area frequency modes present in the power system are also captured by CNN within less than 2% error in natural frequency and less than 4% error in damping ratio. In Table IV, Prony analysis performed on the training data obtained after different learning duration has been presented. The data presented shows that some frequency modes are extracted early on but some frequency modes are either not present or not accurate, which improves over time. For

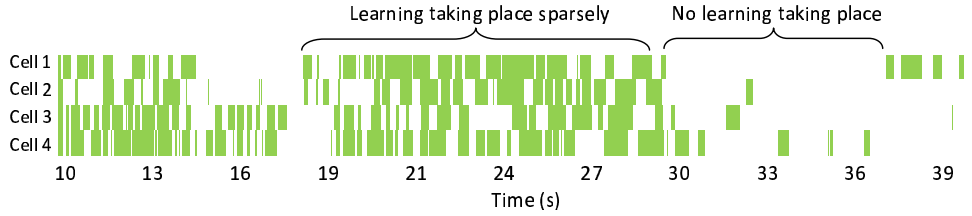

Fig. 12. Asynchronous learning of cells in the CNN. The shaded portion of the figure represents when each cell's learning unit was triggered.

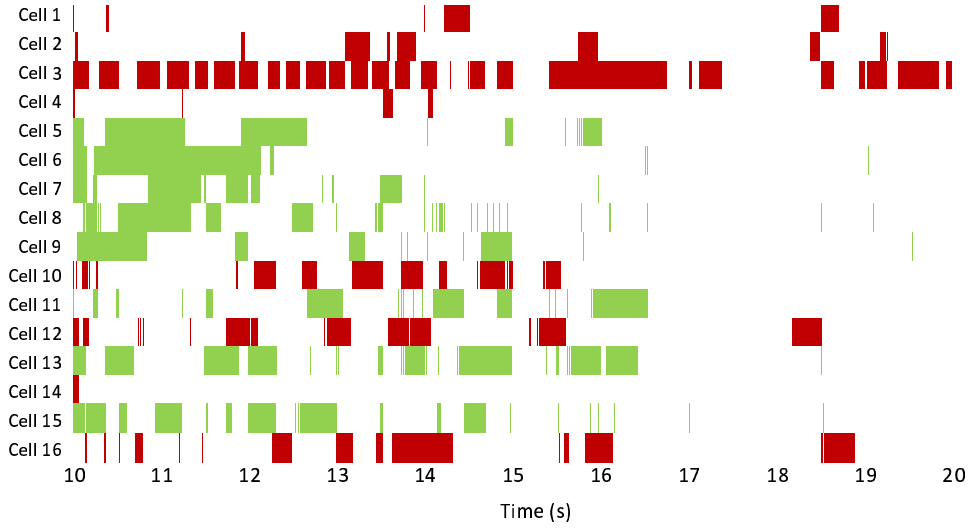

Fig. 13. Asynchronous learning of 16 cells in the CNN implementing Test System II. The shaded portion of the figure represents when each cell's learning unit was triggered. Darker shades represent the cells implemented using SRN and lighter shades represent those implemented using MLP.

TABLE III
NATURAL FREQUENCIES (ω_N) AND DAMPING RATIOS (ζ) OBTAINED WITH PRONY ANALYSIS OF THE CNN OUTPUT COMPARED WITH ACTUAL SIGNAL.

		Actual		Predicted		Error %	
		ω_N	ζ	$\hat{\omega_N}$	$\hat{\zeta}$	$E\omega_N$	$E\zeta$
G1	Mode 1	0.6023	0.1489	0.6035	0.1476	0.1992	0.13
	Mode 2	1.2075	0.1623	1.2026	0.1521	0.4058	1.02
G2	Mode 1	0.6023	0.1504	0.6039	0.1622	0.2657	1.18
	Mode 2	1.2482	0.1424	1.2298	0.1363	1.4741	0.61
G3	Mode 1	0.6036	0.1491	0.6059	0.1486	0.381	0.05
	Mode 2	1.251	0.1483	1.2311	0.1517	1.5907	0.34
G4	Mode 1	0.6036	0.1481	0.6051	0.1474	0.2485	0.07
	Mode 2	1.2196	0.1463	1.233	0.1802	1.0987	3.39

the intra-area frequency modes in the Test System I, The minimum errors in natural frequency modes are obtained after 40 seconds of training, which is the complete set of data used for Test System I. These result showing extraction of frequency modes from the output of CNN validates that a CNN has the ability to extract 'information' from 'data' based on cognitive and social learning.

3) Performance comparison: In this study, n-nearest neighbor topology has been used for developing the CNN where predicted time delayed signals from the neighbors have been used as inputs to each cell. For Test System I, if the same topology is to be implemented on an MLP, four time-delayed neural networks can be obtained by combining (6) to (9) and replacing the predicted outputs with time delayed values of the actual signal. For example, output of MLP for generator $G1$ can be obtained using (6), (7) and (9) as follows:

$$\begin{aligned}\Delta\hat{\omega}_1(k+1) = \; & f\,(W_1(k), V_1(k), \Delta Vref_1(k), \Delta\omega_1(k-2),\\ & \Delta\omega_1(k-1), \Delta\omega_1(k), \Delta Vref_2(k-1),\\ & \Delta\omega_2(k-3), \Delta\omega_2(k-2), \Delta\omega_2(k-1),\\ & \Delta Vref_4(k-1), \Delta\omega_4(k-3), \Delta\omega_4(k-2),\\ & \Delta\omega_4(k-1), \Delta\omega_3(k-1)) \end{aligned} \quad (10)$$

Thus, MLP equivalent of a cell has 13 inputs. Using 20 hidden neurons and one output, the performance of MLP can be equivalently compared with each cell of a CNN. It is not claimed that the number of inputs to the equivalent MLP are optimal, but in order to capture the inter-area and intra-area frequency modes in the power system, all the information available to each cell in CNN is also made available to the equivalent MLP. Test Case I is implemented on the MLP equivalent of the CNN and tested using 10-cycle 3-phase line to ground fault data. Coefficient of determination (R^2) is used to show the goodness of fit between the actual and the predicted signals. R^2 is a statistical measure of how well the regression line approximates the real data points. An R^2 of 1.0 indicates that the regression line perfectly fits the data. Fig. 14 shows the comparison of R^2 between the actual speed deviations of four generators and their predicted values obtained from CNN and its MLP equivalent. It shows that higher values of R^2 are obtained for predictions with CNN as compared to those with MLP. Comparison of MSEs between

TABLE IV
EVOLUTION OF LEARNING IN EACH CELL OVER TIME.

	G1		G2		G3		G4	
	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2	Mode 1	Mode 2
ω_N	0.6023	1.2075	0.6023	1.2482	0.6036	1.251	0.6036	1.2196
$\hat{\omega_N} : 10s$	0.6042	1.2127	0.6052	1.2132	0.6044	1.2271	0.6034	1.2635
$\hat{\omega_N} : 16s$	0.603	1.188	0.6025	1.1745	0.6037	-	0.605	1.2707
$\hat{\omega_N} : 22s$	0.6025	1.1879	0.6009	1.2211	0.6029	1.1956	0.6049	1.272
$\hat{\omega_N} : 28s$	0.6035	1.1935	0.6041	1.1808	0.6033	1.1636	0.6047	1.2598
$\hat{\omega_N} : 34s$	0.6034	1.1931	0.6028	1.1898	0.6025	1.1873	0.6042	1.2615
$\hat{\omega_N} : 40s$	0.6035	1.2026	0.6039	1.2298	0.6059	1.2311	0.6051	1.233
$E\omega_N : 10s$	0.3155	0.43064	0.4815	2.804	0.1325	1.9104	0.0331	3.5995
$E\omega_N : 16s$	0.1162	3.6853	0.0332	5.9045	0.0166	-	0.2319	4.1899
$E\omega_N : 22s$	0.0332	1.6231	0.2324	2.1711	0.1160	4.4284	0.2153	4.2965
$E\omega_N : 28s$	0.1992	1.1594	0.2988	5.3997	0.0497	6.9864	0.1822	3.2962
$E\omega_N : 34s$	0.1826	1.1925	0.0830	4.6787	0.1822	5.0919	0.0994	3.4355
$E\omega_N : 40s$	0.1992	**0.4058**	0.2656	**1.4741**	0.381	**1.5907**	0.2485	**1.0987**

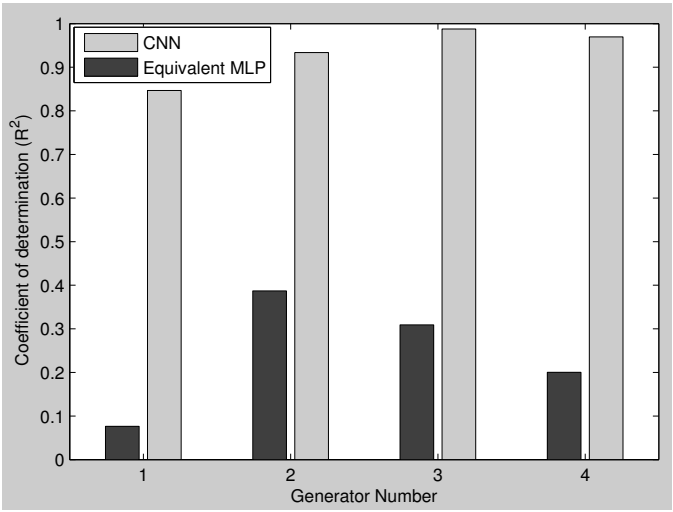

Fig. 14. Coefficients of determination showing goodness of fit between the actual and the predicted outputs using CNN and its MLP equivalent.

TABLE V
COMPARISON OF MSE OBTAINED USING CNN AND MLP

	Training ($\times 10e-8$)		Testing ($\times 10e-8$)	
	CNN	Equivalent MLP	CNN	Equivalent MLP
$G1$	1.9	2.3	30.4	183.2
$G2$	1.4	2.3	12.8	118.8
$G3$	1.9	2.0	3.2	178.4
$G4$	2.9	31.9	5.5	146.9
Avg.	**2.02**	**9.62**	**12.97**	**156.82**

the actual and the predicted signals for all outputs are shown in Table V. The results demonstrate that the performance of CNN is significantly better than that obtained using MLPs. Although most of the outputs of both CNN and its MLP equivalent are comparable during training, the performance of equivalent MLP significantly deteriorates during testing on a set of data that was not used during training. This means that the learning in MLP is being forced by the backpropagation of errors where as CNN is actually able to capture the dynamics of the system. The physical implementation of equivalent MLP will be more complex and requires more variables to be communicated.

Unlike Test Case I, where all the cells are homogeneous in size, architecture and learning method, Test Case II is heterogeneous with some of the cells consisting of different number of inputs, different type of computational and learning units. These heterogeneity in size of the cells was a result of increased number of connections in certain cells in order to capture the system topology. However, the heterogeneity in architecture and learning was a result of implementing the cells with larger number of inputs using SRN and hence training it using PSO. While this demonstrates the flexibility of the framework, it does not imply that the heterogeneity in architecture or learning is desired and will be problem specific. To validate this assumption Test Case II is also implemented using homogeneous cells where all of the cells consist of MLP as the computational unit and is trained using BP. Fig. 15 shows a comparison of absolute errors between the actual and the predicted signals for Test Case II when implemented using heterogeneous (top plot) and homogeneous (bottom plot) architectures and learning methods. These results show that the performance of homogeneous and heterogeneous cells in CNN are comparable in this application. The higher values of absolute error observed in the heterogeneous CNN shows the errors for Cells 12 and 16, which are implemented using SRNs. Implementation and learning in SRN is challenging and time consuming due to the inherent feedback. Therefore, based on the studies, it can be empirically concluded that use of simplified architectures of neural networks, such as an MLP, in individual cells of CNN is more efficient. It can also be inferred that the capability of CNN is dependent not only on its computational unit (which architecture of neural network is used in each cell) but also how well the CNN maps the system topology through its connectivity.

4) Speed comparison: Due to the size of the four MLPs and the four cells of a CNN being different, the time required for training of these networks on the same data differs significantly. Since both implementations are equivalent in terms of inputs and topology, they are both linearly scalable. Each cell of a CNN consists of 56 sets of weights (hence a total of 224 weight updates in each epoch). However, each equivalent MLP consists of 280 weights resulting in 1120 weight updates. Therefore, time required for training a CNN is **79** seconds as opposed to **277** seconds for the equivalent MLP on the same computer. Similarly, training of SRNs in Test System II using iterative algorithm (PSO) requires much longer time. This also

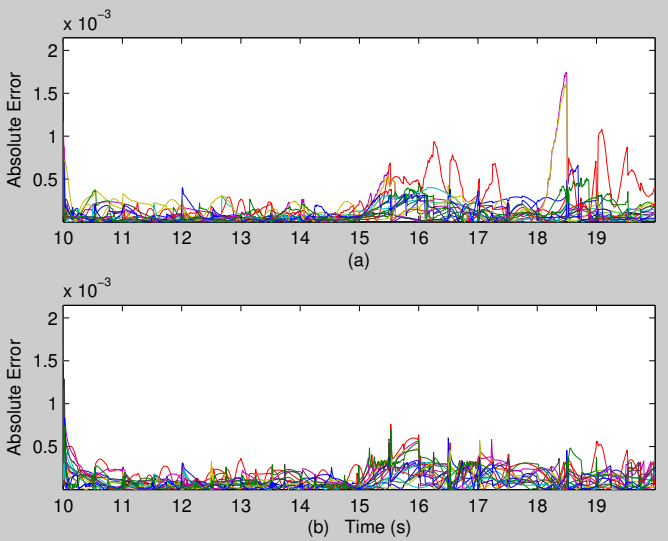

Fig. 15. Absolute error between the actual and the predicted outputs of 16 cells in Test Case II. (a) Errors obtained from a CNN with cells having heterogeneous architectures (MLP and SRN) and learning methods (BP and PSO). (b) Errors obtained from CNN with cells having homogeneous architectures (MLP) and learning methods (BP).

justifies the use of simple architectures and learning methods in each cell of a CNN.

VII. Conclusion

Learning of learning systems is a challenge due to individual learning systems interacting with other members of the swarm in a collaborative environment. The objective of such learning is to improve overall performance of the system. However, this is only achieved by improving each individual's (subsystem) outputs. Therefore, a cognitive as well as social learning is required. This is addressed in the paper by developing a decentralized asynchronous collaborative learning framework for CNN. In such a framework, learning takes place locally on spatially distributed cells that learn asynchronously. The cells collaborate to achieve learning of the overall system. An application of the proposed method is shown by developing a wide area monitoring system for a power systems using heterogeneous CNN. The performance of CNN is compared and shown to be better than that of equivalent MLP architecture. Through a series of empirical tests, it is also concluded that best performance of CNN is achieved by actually capturing the dynamics of the system through its connection architecture while simplifying the computational and learning units in each cell. As a result, CNN with DAL framework presented in the paper becomes a scalable, high performance, decentralized learning system. The presented concept of decentralized asynchronous learning more accurately represents real life scenarios. Although the cells in CNN have been implemented on different processors in parallel, the speedup obtained by parallelization is not the main focus of this paper and has not been compared with sequential platform and is not presented as findings of this research. Real-time parallel implementation of CNNs remains to be investigated as future work. Extension of the presented methods and applications in larger and more complex dynamic systems with more variables can also be area of research for future work.

References

[1] L. Chua and L. Yang, "Cellular neural networks," in *IEEE International Syposium on Circuits and Systems*, vol. 2, 1988, pp. 985–988.

[2] P. Tzionas, A. Thanailakis, and P. Tsalides, "A hybrid cellular automaton/neural network classifier for multi-valued patterns and its vlsi implementation," *Integration, the VLSI Journal*, vol. 20, no. 2, pp. 211 – 237, 1996.

[3] T. Su, Y. Du, Y. Cheng, and Y. Su, "A fingerprint recognition system using cellular neural networks," in *International Workshop on Cellular Neural Networks and Their Applications*. IEEE, 2005, pp. 170–173.

[4] L. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on circuits and systems*, vol. 35, no. 10, pp. 1257–1272, 1988.

[5] I. Aizenberg, N. Aizenberg, J. Hiltner, C. Moraga, and E. M. zu Bexten, "Cellular neural networks and computational intelligence in medical image processing," *Image and Vision Computing*, vol. 19, pp. 177 – 183, 2001.

[6] Z. Zeng and W. X. Zheng, "Multistability of neural networks with time-varying delays and concave-convex characteristics," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 293 –305, feb. 2012.

[7] P. Werbos and X. Pang, "Generalized maze navigation: Srn critics solve what feedforward or hebbian nets cannot," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, oct 1996, pp. 1764 –1769 vol.3.

[8] D. Wunsch, "The cellular simultaneous recurrent network adaptive critic design for the generalized maze problem has a simple closed-form solution," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, vol. 3, 2000, pp. 79 –82 vol.3.

[9] R. Ilin, R. Kozma, and P. Werbos, "Beyond feedforward models trained by backpropagation: A practical training tool for a more efficient universal approximator," *IEEE Transactions on Neural Networks*, vol. 19, no. 6, pp. 929 –937, june 2008.

[10] Y. Ren, K. Anderson, K. Iftekharuddin, P. Kim, and E. White, "Pose invariant face recognition using cellular simultaneous recurrent networks," in *International Joint Conference on Neural Networks (IJCNN 2009)*, june 2009, pp. 2634 –2641.

[11] K. Anderson, K. Iftekharuddin, E. White, and P. Kim, "Binary image registration using cellular simultaneous recurrent networks," in *IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing*, 30 2009-april 2 2009, pp. 61 –67.

[12] Y. Shen and J. Wang, "Robustness analysis of global exponential stability of recurrent neural networks in the presence of time delays and random disturbances," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 23, no. 1, pp. 87 –96, jan. 2012.

[13] L. Grant and G. Venayagamoorthy, "Voltage prediction using a cellular network," in *IEEE Power and Energy Society General Meeting*, july 2010, pp. 1 –7.

[14] J. Hunt and D. Cooke, "An adaptive, distributed learning system based on the immune system," in *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3. IEEE, 1995, pp. 2494–2499.

[15] F. Provost and D. Hennessy, "Scaling up: Distributed machine learning with cooperation," in *Proceedings of the National Conference on Artificial Intelligence*. Citeseer, 1996, pp. 74–79.

[16] P. Auer, H. Burgsteiner, and W. Maass, "Reducing communication for distributed learning in neural networks," *Artificial Neural NetworksI-CANN 2002*, pp. 133–133, 2002.

[17] A. Asuncion, P. Smyth, and M. Welling, "Asynchronous distributed learning of topic models," *Advances in Neural Information Processing Systems*, vol. 21, pp. 81–88, 2008.

[18] T. Alpcan and C. Bauckhage, "A distributed machine learning framework," in *Proceedings of the 48th IEEE Conference on Decision and Control*, dec. 2009, pp. 2546 –2551.

[19] M. Dorigo and L. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[20] O. Tilak and S. Mukhopadhyay, "Decentralized and partially decentralized reinforcement learning for distributed combinatorial optimization problems," in *Ninth International Conference on Machine Learning and Applications (ICMLA)*, dec. 2010, pp. 389 –394.

[21] L. Huang, M. I. Jordan, A. Joseph, M. Garofalakis, and N. Taft, "In-network pca and anomaly detection," in *In NIPS*. MIT Press, 2006, pp. 617–624.

[22] C. Dede, "Emerging technologies and distributed learning," *American Journal of Distance Education*, vol. 10, no. 2, pp. 4–36, 1996.
[23] M. Alavi, G. Marakas, and Y. Yoo, "A comparative study of distributed learning environments on learning outcomes," *Information Systems Research*, vol. 13, no. 4, p. 404, 2002.
[24] G. Salomon and D. Perkins, "Individual and social aspects of learning," *Review of research in education*, vol. 23, pp. 1–24, 1998.
[25] P. Kundur, *Power system stability and control*. McGraw-Hill Professional, 1994.
[26] A. Abur and A. Exposito, *Power system state estimation: theory and implementation*. CRC, 2004, vol. 24.
[27] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it," *Neural Networks*, vol. 22, no. 3, pp. 200 – 212, 2009.
[28] L. Werbos and P. Werbos, "Self-organization in cnn-based object nets," in *Cellular Nanoscale Networks and Their Applications (CNNA), 2010 12th International Workshop on*, feb. 2010, pp. 1 –6.
[29] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *IEEE International Conference on Neural Networks*, vol. 4, 1995.
[30] C. Zhang, H. Shao, and Y. Li, "Particle swarm optimisation for evolving artificial neural network," in *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 4. IEEE, 2000, pp. 2487–2490.
[31] R. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proceedings of the 2001 congress on evolutionary computation*, vol. 1. Piscataway, NJ, USA: IEEE, 2001, pp. 81–86.
[32] RTDS, "Real time digital simulator tutorial manual (rscad version)," RTDS Technologies, March 2008.
[33] J. Hauer, C. Demeure, and L. Scharf, "Initial results in prony analysis of power system response signals," *IEEE Transactions on Power Systems*, vol. 5, no. 1, pp. 80–89, 1990.

Bipul Luitel (S'08-M'12) received his MS (2009) and PhD (2012) degrees in Computer Engineering from Missouri University of Science & Technology, Rolla, MO, USA. He is currently a postdoctoral fellow at Clemson University, Clemson, SC. His research area includes development and application of cellular neural networks for situational awareness in smart grids, implementation on high performance computing platforms, and real-time power system data analytics. He was a recipient of IEEE CIS Walter Karplus Graduate Research Grant in 2010 and IEEE CIS Student Travel Grant in 2011.

Ganesh Kumar Venayagamoorthy (S'91-M'97-SM'02) received his Ph.D. degree in electrical engineering from the University of Natal, Durban, South Africa, in 2002. He is the Duke Energy Distinguished Professor of Electrical and Computer Engineering at Clemson University, Clemson, USA. Prior to that, he was a Professor of Electrical and Computer Engineering at the Missouri University of Science and Technology (Missouri S&T), Rolla, USA. He was a Visiting Researcher with ABB Corporate Research, Sweden, in 2007. Dr. Venayagamoorthy is Founder and Director of the Real-Time Power and Intelligent Systems Laboratory (http://rtpis.org). His research interests are in the development and applications of advanced computational algorithms for real-world applications, including power systems stability and control, smart grid applications, sensor networks and signal processing. He has published 2 edited books, 8 book chapters, and over 90 refereed journals papers and 300 refereed conference proceeding papers.

Dr. Venayagamoorthy is a recipient of several awards including a 2008 US National Science Foundation (NSF) Emerging Frontiers in Research and Innovation Award, a 2007 US Office of Naval Research Young Investigator Program Award, a 2004 NSF CAREER Award, the 2010 Innovation Award from St. Louis Academy of Science, the 2010 IEEE Region 5 Outstanding Member Award, the 2006 IEEE Power and Energy Society Outstanding Young Engineer Award, and the 2003 International Neural Network Society's Young Investigator Award. He is the recipient of the 2012 Institution of Engineering and Technology (IET) Generation, Transmission and Distribution Premier Award for the best research paper published during last two years for the paper "Wide area control for improving stability of a power system with plug-in electric vehicles."

Dr. Venayagamoorthy is involved in the leadership and organization of many conferences including the co-Chair of the 2013 IEEE Symposium of Computational Intelligence Applications in Smart Grid (CIASG). He is currently the Chair of the IEEE PES Working Group on Intelligent Control Systems, the Founder and Chair of IEEE Computational Intelligence Society (CIS) Task Force on Smart Grid, and the Chair of the IEEE PES Intelligent Systems Subcommittee. He is currently an Editor of the IEEE Transactions on Smart Grid.

Dr. Venayagamoorthy is a Fellow of the IET, UK, and the South African Institute of Electrical Engineers (SAIEE).