computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

AN IMPROVED ARCHITECTURE FOR COMPETITIVE AND COOPERATIVE NEURONS (CCNS) IN NEURAL NETWORKS

M. Kamrul Islam

School of Computing,
Queen's University,
Kingston, K7L 3N6, ON, Canada
islam@cs.queensu.ca

Abstract: *In neural networks, the associative memory is one in which applying some input pattern leads to the response of a corresponding stored pattern. During the learning phase the memory is fed with a number of input vectors and in the recall phase when some known input is presented to it, the network recalls and reproduces the output vector. Here, we improve and increase the storing ability of the memory model proposed in [1]. We show that there are certain instances where their algorithm can not produce the desired performance by retrieving exactly the correct vector. That is, in their algorithm, a number of output vectors can become activated from the stimulus of an input vector while the desired output is just a single vector. Our proposed solution overcomes this and uniquely determines the output vector as some input vector is applied. Thus we provide a more general scenario of this neural network memory model consisting of Competitive Cooperative Neurons (CCNs).*

Keywords: *Competitive cooperative neuron, associative memory, vector, frequency bands.*

1. INTRODUCTION

The ability to store and retrieve information is critical in any type of neural network. In neural network, the memory, particularly associative memory, can be defined as the one in which the input pattern or vector leads to the response of a corresponding stored pattern (output vector). That is, when an input vector is presented, the network recalls the corresponding output vector associated with the input vector. There are two types of associative memories: *autoassociative* and *heteroassociative* memory. In the case of autoassociative memory, both input and output vectors range over the same vector space. For example, a spelling corrector maps incorrectly spelled words (e.g. "matual") to correctly spelled words ("mutual"). Heteroassociation involves the mapping between input and output vectors over a different vector space. For example, given a name ("John") as input, the system will be able to recall its corresponding phone number ("657-9876") stored in memory.

In the context of neural network, an associative memory consists of neurons (known as conventional McCulloh-Pitts [2] neurons) that are capable of processing input vectors and recalling output vectors. These conventional model neurons use inputs from each source that are characterized by the amplitude of input signals. In this way each neuron can receive, process, and recall only one component of a memorized vector. Towards realizing the concept of associative memory, one of the commonly used techniques uses *correlation matrix memory* [3] which encodes all input and output vector pairs $\{y_k,x^T_k\}(k=1,2,3,...,n)$ into a correlation matrix, $M = \sum_{k=1}^{n} y_k x_k^T$. Later in the recall phase the matrix M is decoded to extract the output vector when the corresponding input vector is introduced to the network. The limitation of correlation matrix memory, in terms of memory capacity, is that it requires exactly *n* neurons to recall *n* components of a vector. In this paper we study the problem of increasing memory storage and recall capacity of a general associative memory and offer an idea that provides and ensures more storage and correct recall ability of the memory model (one layer Competitive Cooperative Neuron (CCN) network model) proposed in [1]. Our proposed method has an improved architecture of the CCN network where we need only N neurons (CCNs) to store and recall NR memories where R is the number of zones (defined later) of a CCN.

The organization of the paper is as follows. First

in Section 2, we provide a general description of a CCN. In Section 3 we show how a network of such neurons can be formed and how they work by providing an example. Section 4 provides evidence to show the limitations found in the CCN network. Our result, that is, the improvement of the CCN network model is given in Section 5 which overcomes the shortcomings of the model [1]. Experimental results are described and shown in Section 6 followed by future research direction in Section 7. Finally, we conclude in Section 8.

2. DESCRIPION OF A CCN

Here we provide a concise description of the CCN which is the building block of a CCN neural network, the reader is referred to [1] for details. In order to increase the memory storage and the recall capacity of an associative memory compared to correlation matrix memory, the paper [1] introduces a novel type of model neuron called CCN as the building block of an associative memory. This model offers two new features: one is that the input signals are characterized by a two-dimensional parameter set representing the *amplitude* and the *frequency* of signals, whereas the conventional inputs have only component, the amplitude. The other feature of the CCN is that it consists of several distinct and autonomous receptor zones where each zone is able to receive a number of such input signals. Each zone is capable of selecting just one signal (called the *winning signal*) from the inputs signals it receives. A model of such a CCN is given in Figure 1.

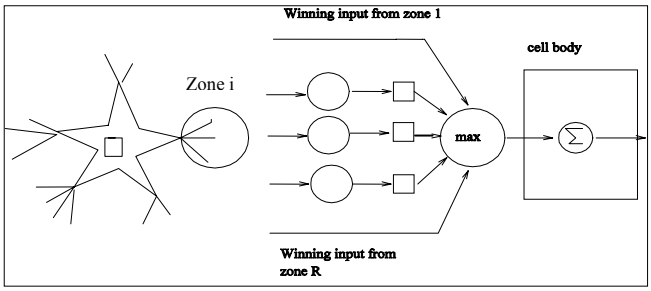

Fig. 1 – A CCN Model: the CCN on the left has five autonomous zones, each of which has a narrow bandwidth of frequencies that it can detect. Each zone receives m input signals. In each zone, only the input signals that have a frequency fi, r that falls within the zone's bandwidth participate in the competition and the winner is the signal with the highest effective amplitude. All the winning signals are propagated to the cell's body, where they cooperate and the cell is activated if the cumulative amplitude is greater than the cell's threshold.

The CCN consists of a number of zones R and each zone $r \in R$ collects input from many sources, $S(r) = \{S_1(r),S_2(r),S_3(r),...\}$. Each input signal $S_i(r) = (F_i(r),A_i(r))$ has two components which are normalized to one - the frequency $F_i(r) \in [0,1]$ which encodes the information [4] and the amplitude $A_i(r) \in [0,1]$-the strength of the signal. Each zone is sensitive to a small range of frequencies (also called *bands*) which means that any input signal whose frequency falls in the range can participate in the competition to be chosen by the zone.

The center of the band of input zone r of a CCN n at time t is denoted by B(n,r,t) and the tolerance level is T(n,r,t). The tolerance level is the dispersion from the center of the band which defines the range of frequency zone r can handle, that is, [B(n,r,t)-T(n,r,t),B(n,r,t)+T(n,r,t)]. The tolerance level and the center of band are not constant, they change over time. A zone can detect the input signals whose frequencies that are fall in the range of its frequencies and the amplitudes of these exceed a certain threshold value τ (n,r,t) $\in [0,1]$. An input i in zone r wins if $A_i(r) >= \tau$ (n,r,t)>0 and $F_i(r) \in$ [B(n,r,t)-T(n,r,t),B(n,r,t)+T(n,r,t)]. A zone is called active if it can select such an input signal. This way each zone propagates its winning signal to the cell body. Finally, the CCN fires if the combined amplitude of all the winning input signals from all the zones exceeds the threshold v(n,t) of the CCN body, that is, $\sum_{r=1}^{R} A_{i(w)}(r) \geq v(n,t)$ where $A_{i(w)}(r)$ is the amplitude of the winning signal of zone r. As the CCN is fired (activated) it sets the center of the frequency band of an active zone r to its corresponding winning signal, i.e., $B(n,r,t)=F_{i(w)}(r)$ where $F_{i(w)}(r)$ is the frequency of the winning signal of zone r. As the CCN fires, it generates the output vector whose components are the winning signals of all the active zones of the CCN. However, the output vector can be modified by using Hebbian learning [5] protocol. Initially, a CCN body threshold is set with the value v(n,t), which is greater than or equal to the sum of the zone thresholds, i.e., $v(n,t) \geq \sum_{r} \tau(n,r,t)$, so that in order for the CCN to fire, either all the zones must be active or some of them must receive a very strong signal to activate the CCN.

When fired, the CCN decreases tolerance levels of the corresponding active zones by some pre-specified value and the amplitude of the threshold of the CCN is also decreased to some level. Similarly if the CCN is not fired the corresponding tolerance levels of active zones are increased (anti-Hebbian learning). However, when the CCN fires we say the CCN specializes or learns. Thus by modifying the tolerance levels and the amplitude threshold of the CCN, the CCN is trained. In this way, the training

phase stores an input vector whose components are the values of the frequencies of the winning signals which are set in the corresponding centers of the bands of the active zones. When it receives input in some but not all zones it uses that input to recall the previous inputs to the idle zones. For example [1], if a CCN has three zones and it fired when the input was the vector that represents the triple (Red, Sweet, Strawberry), then the next time it receives only "Red" and no input from the other zones, it will fire ("Red", "Sweet", "Strawberry") provided that the amplitude of the input signal ("Red") exceeds the cell's threshold.

2.1 CCN NETWORK MODEL

A simple one-layer feedforward network with three CCNs and three input sources is shown in Fig. 2. Each CCN has three receptor zones represented by the vertices of the triangles. The number of inputs to the different zones is not necessarily the same, but it can be made the same by adding zero-weight input signals.

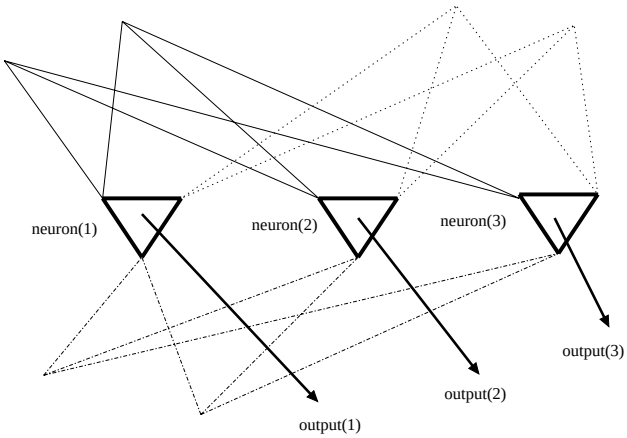

Fig. 2 – A simple one-layer feedforward network with three CCNs and three input sources is shown. Each CCN has three receptor zones represented by the vertices of the triangles.

3. HOW A CCN NETWORK WORKS

In order to understand the function of the network explained above, consider a one layer feed-forward network of CCN with N CCNs where each CCN $n \in N$ has 3 zones. Assume the centers of the frequency bands $B(n,r_1,t)$, $B(n,r_2,t)$, $B(n,r_3,t)$ of zones 1, 2 and 3 of CCN n are set to 0.150, 0.450, and 0.750 respectively. Let the tolerance level T for all zones be 0.050. The range of frequencies for each zone becomes [B-T, B+T], i.e., [0.100, 0.200], [0.400, 0.500] and [0.700, 0.800] and let the thresholds of zones 1, 2, and 3 be τ_1=0.20, τ_2=0.15, τ_3=0.15 respectively and the CCN body threshold v=0.32

Assume that we want the network to store and recall vectors consisting of name, gender and id of students. Let the input vector be ("Sarah", "Female", "4781234") which is represented by frequencies (0.130, 0.416, 0.725). Let 0.20 be the amplitude of each of the components of the vector. As the input vector (assuming first component of the input vector to zone 1, second component to zone 2 and so on) is applied to the network, all the zones become active and the CCN starts firing since 0.20+0.20+0.20>0.32. If it does not fire then the tolerance level of inactive zones can be increased gradually to accommodate the frequency (anti Hebbian learning [5]). Now as the CCN fires the threshold to each zone is reduced to some minimum level (to some minimum value required to activate the corresponding zone) and the threshold to the cell body is also reduced to some minimum value. Let the cell body threshold be reduced to v=0.18. Now the center of the frequency band of each zone will be assigned the frequency of its winning signal, i.e., frequencies (0.130, 0.416, and 0.725) are assigned to zones 1, 2, and 3 respectively and the output vector becomes (0.130, 0.416, 0.725).

Now in the recall phase if we apply only input "Sarah" to zone 1 and no input from the other zones, the CCN will fire (because the threshold 0.20 of signal "Sarah" is greater than the CCN threshold 0.18) and produce the whole output vector (0.130, 0.416, 0.725) representing the vector ("Sarah", "Female", "4781234"). Therefore, if there are R (R-dimensional vector) zones in a CCN, then a single input signal to a zone will result in R recalled features (R-1, if we exclude the activating input) from the other zones, which is more efficient than recalling only one feature from every input compared with the correlation matrix memory. In general, if there are N CCNs each with R zones (i.e., a CCN can store and recall an R- dimensional vector) in a one-layer feed forward CCN network then it is able to recall total NR memories. This is because, after the training is complete a signal (as given in the previous example, only input component "Sarah") to a zone in a CCN will be strong enough to fire the CCN and recall all the other signals of other zones of that CCN. As a whole, only N input signals to N CCNs will suffice to recall NR memories. On the other hand, the correlation matrix memory needs NR CCNs to recall NR memories. Therefore the achievement of performance in terms of stored-features/number-of-CCNs ratio is higher in CCN network as compared to correlation matrix memory [3].

4. LIMITATIONS IN CCN NETWORKS

Here we consider a situation where the CCN model cannot achieve the performance as mentioned in the paper. First, we show that there are certain instances where the existing CCN model [1] fails to produce the expected output result. Then we offer an improvement to the architecture of the network model such that stipulated performance can be ensured. The associative network consisting of the proposed CCNs [1] functions well if only one input vector can be attracted to at most one CCN of the network during training. This is only possible when no two CCNs have all the centers of the frequency bands (B(n,r,t)) are equal. The network may suffer serious limitation in manipulating (storing and recalling) data when all the centers of the frequency bands of a CCN coincide with those of any other CCN. Mathematically this situation can be expressed as the following: if there are R zones of a CCN then there exist at least two CCNs n_i and n_j such that

$B(n_i,r_1,t)=B(n_j,r_1,t)$,
$B(n_i,r_2,t)=B(n_j,r_2,t),...,B(n_i,r_R,t)=B(n_j,r_R,t)$.

Under these circumstances, the network reaches a situation where the same input vector, M_j is stored in different CCNs. This is because the input, M_j stimulates and fires all those CCNs which have the same centers of frequency bands of their zones. Here we show how storage capacity decreases for the case stated above. As mentioned earlier, if the associative memory network has N CCNs and each CCN has R zones then we can store and recall N memory vectors (total NR memories) where each memory vector M_i consists of R-components. In this way, we can say this is equivalent to recall exactly NR memories in total. Let S be the number of CCNs that have the same centers of frequency bands of in their zones. This means that there is a memory vector M_s whose input can simultaneously fire S CCNs.

As M_s is stored in all the S CCNs, their centers of frequency bands will be assigned the corresponding frequencies of M_s(each component of M_s is represented by a frequency) and no other vector M_t, ($M_s \neq M_t$) can be stored in any of the S CCNs. So we have only N-S CCNs left to store N-1 vectors. If S>1, then we can not store all the remaining vectors (remaining N-1 vectors, since only one memory vector M_s is stored) to the memory. Therefore, this case does not allow us to store and recall N vectors. In the worst case, if all the N CCNs have the same centers of frequency bands then we can store only one vector in the whole network instead of N vectors. Thus the performance degrades down to 1/N percent which is quite worst for large values of N. In general, let S_1 be the number of CCNs having the same centers of frequency band $f^1_1,..., f^1_R$, S_2 be the number of CCNs with the same centers of frequency band $f^2_1,..., f^2_R$ and S_p be the number of CCNs with the same centers of frequency band $f^p_1,...,f^p_R$, then we can achieve p/N percent of vectors to be stored and recalled correctly where $S_1+S_2+...+S_p=N$. The following example demonstrates such a case.

Suppose the network is required to store input vectors {0,1,0,0}, {1,1,0,0}, {1,0,1,0} and recall when any of the vectors is presented to the network. Assume we have a network consisting of three CCNs each with four zones. Let the first, second, and third CCN's band centers are 0.1, 0.2, 0.1, 0.1; 0.1, 0.2, 0.1, 0.1 and 0.2, 0.2, 0.1, 0.1, respectively. Let 0 and 1 be encoded by the frequencies 0.1 and 0.2 respectively. Therefore, we obtain the equivalent representation of the four input vectors as {0.1, 0.2, 0.1, 0.1}, {0.2, 0.2, 0.1, 0.1} and {0.2, 0.1, 0.2, 0.1}, respectively. As we apply the input {0.1, 0.2, 0.1, 0.1} to activate some CCN, we find all the zones of CCNs 1 and 2 become active and they fire. Thus, the same input vector {0, 1, 0, 0} is stored in both CCNs. In this way, the two CCNs are stimulated and their centers of frequency bands are assigned the frequencies 0.1, 0.2, 0.1, and 0.1. When the second input pattern {0.2, 0.2, 0.1, 0.1} is presented it stimulates the third CCN and causes it to fire by storing the input frequencies to the corresponding centers of frequency bands. Now for the last input there is no CCN that can be activated since all the CCNs are already attracted to the two previous input vectors. Although we have three CCNs to store and recall three vectors according to the algorithm presented in the paper [1], we cannot store more than two input patterns in the associative memory for this particular example. Thus the performance of the proposed technique degrades in this case.

5. SOLUTION PROPOSED FOR CCN NETWORKS

In this section, we provide the improvement for the architecture of the CCN network to remedy the situation illustrated above. This is intended so that at most one CCN in the network can be stimulated (attracted) by a single input pattern. The modified network is shown in Fig. 3.

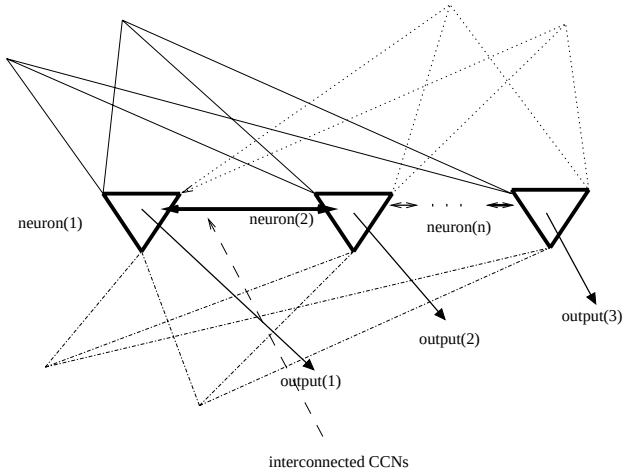

Fig. 3 – An improved simple one-layer feedforward network with three CCNs and three input sources is shown. CCNs are connected.

The main idea is to connect a CCN to its left and right neighbors and assign indices to them, except for the leftmost and rightmost CCNs which are only connected to their right and left CCNs, respectively. These indices, beginning from 1 to the number of CCNs, will be assigned arbitrarily among the CCNs. It is assumed that the CCN with the lowest index has the highest priority and priority will decrease with the increase of indices. After an input pattern is applied to the network, if a CCN gets stimulated (call it *active*) then it sends its index to all other CCNs. If it is not active then it refrains from sending its index. We ensure that the highest priority active CCN will be the one to be attracted to the input if there are more than one such active CCNs.

As each active CCN sends its index to all other, every active CCN compares the index it receives from other active CCNs and if any of the indices is smaller than its own index then it does not update its center of frequency band. This means that although it is a candidate for the input to store, it withdraws its candidacy and let other higher priority CCNs be attracted to the input. In this way, only the smallest indexed CCN wins and processes the input and changes its centers of frequency bands of its zones to the corresponding winning frequencies. Mathematically, this is a one-to-one function f: $\boldsymbol{S} \rightarrow$ N, where $\boldsymbol{S}$ denotes the set of CCNs in the network. Let $S \subseteq \boldsymbol{S}$ denote the set of CCNs simultaneously attracted to an input. It is obvious that there will be exactly one $C' \in S$ where $f(C') \neq f(C'')$ $(C'' \in S-\{C'\})$. By following the above procedure to propagate the indices among the CCNs, we obtain exactly one active C' which has the smallest index among the indices of the CCNs in S since f is one-to-one. For example, in a network of 11 CCNs, if CCNs with indices 2, 7, 11 become active for some input pattern, then the CCNs 7 and 11 will withdraw because they find the index 2 is smaller. As a result, CCN 2 will take over and become stimulated and attracted to the input. The introduction of priority ensures that at any time when an input pattern is presented in the network at most one CCN will be attracted to that input. Thus we eliminate the chance of firing more than one CCNs by a single input which overcomes the problem mentioned in earlier section.

6. EXPERIMENTAL RESULTS

According to the method presented in the previous section, we provide the numerical results and analyze the outcome by computing the percent of memory that can be stored and retrieved successfully. We compare the outcome of our results with the conventional network as proposed in [1]. There are two phases, namely, the training phase and the recall phase. In the training phase, we setup the network with a certain number of CCNs and the number of features or vector elements for each of the CCNs. After feeding the inputs in the network, we train the network to memorize the vectors in the CCNs. That is, the individual CCNs memorize the vectors by the technique mentioned above. The number of vectors to be stored equals the number of CCNs in the network and the number of elements in each vector is equal to the number of zones of each CCN. After we finish training the network, we perform the recall phase in which we provide only some element, m_i of a vector, $M=(m_1,m_2,...,m_R)$ to the network and the network produces the whole vector (i.e., all the elements of M).

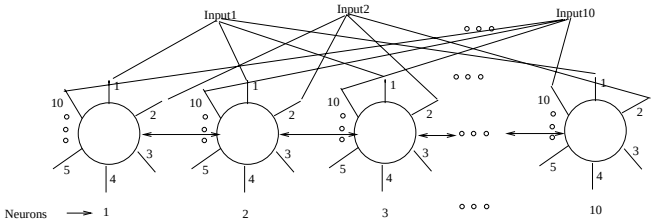

Fig. 4 – The CCN network model of our experiment is shown for 10 CCNs.

We perform our experiment with the network shown in Fig. 4 consisting of N=10 CCNs and each CCN has R=10 zones meaning each input vector $M=(m_1,m_2,...,m_{10})$ has 10 elements that can be accommodated by a single CCN. First, we remove the interconnections (denoted by the double arrow line segments between CCNs) among the CCNs so that the network becomes the one as described in [1] and then we perform the following experiments.

We have 10 sets of centers of frequency bands uniformly distributed over [0,1]. For example, the centers of frequency bands $B(n_i,r_1,t)$, $B(n_i,r_2,t)$,

$B(n_i,r_3,t),...,B(n_i,r_9,t)$, $B(n_i,r\_{10},t)$ for a CCN n_i can be 0.01,0.02,...,0.10 for zone 1,2,...,10, respectively. Thus, each set of frequency bands starts with 0.10*k+0.01 and ends with 0.10*k+0.10 where k=0,1,2,...,9. And the value of the tolerance level T for all zones is set to 0.005. In the experiment, randomly 10 sets of centers of frequency bands are assigned to 10 arbitrary CCNS.

Recall from the notation $S_1, S_2,...,S_p$ in Section 4 where S_i is the number of CCNs having the same centers of frequency band $f^i_1,..., f^i_{10}$, and $\sum_i S_i = 10$. We now give one sample of our experiment and then we provide the simulation results. In one random input for the network model, we find $S_1=2$, $S_2=1$, $S_3=1$, $S_4=3$, $S_5=1$, $S_6=1$, $S_7=1$ and $S_1+S_2+S_3+S_4+S_5+S_6+S_7=10$. This amounts to store and retrieve only seven (7) vectors ($S_1,S_2,...,S_7$) of 10*7=70 vector elements in total by the network model of the experiment.

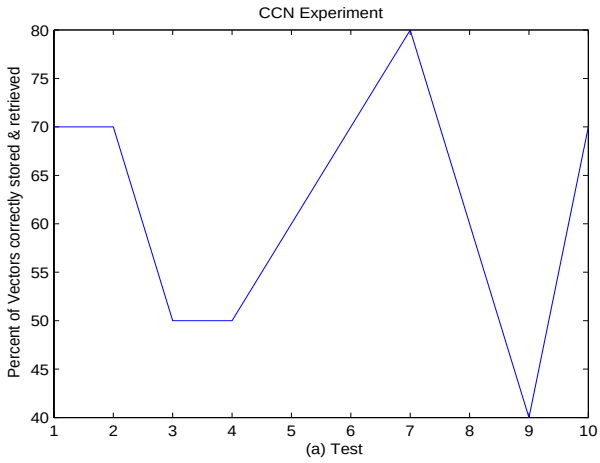

Fig. 5 – Showing result for a set of 10 runs where for each run the corresponding percentage of performance of correctly storing and retrieving vectors is shown along the y-axis.

This is because, for some S_i, $i \in \{1,2,...,7\}$ we can store only one vector (consisting of 10 elements) instead of all 10 vectors of S_i*R elements whereas our proposed method can store and retrieve all N=10 vectors of N*R=10*10=100 vector elements in total. Thus in this particular instance the performance of conventional method in terms of storing and retrieving vectors of around 70 percent of the total vectors.

Fig. 5 shows the graph for a set of 10 test runs where the test numbers are put along the x-axis and the corresponding percentage of vectors that can be stored and retrieved uniquely are shown in the y-axis. This random set of 10 test runs gives us an overview of how the CCN network [1] performs. Of the 10 runs we can achieve 80 percent performance once, i.e., eight vectors out of 10 are stored and retrieved correctly (on the 9th test run). And in general, the performance lies around 70 percent for a number of times and shows poor percentage (40 percent) once.

However, these random runs do not tell us about the true characteristics of the conventional CCN network. Thus we perform 100 runs and then average the results of the 100 runs which is depicted in Figure 6. Here we put the value of percentage of correctly storing and retrieving vectors in the x-axis and the number of times (out of 100 runs) the corresponding percentage is obtained is shown in the y-axis. This gives us a clear view of the characteristics of the CCN network [1].

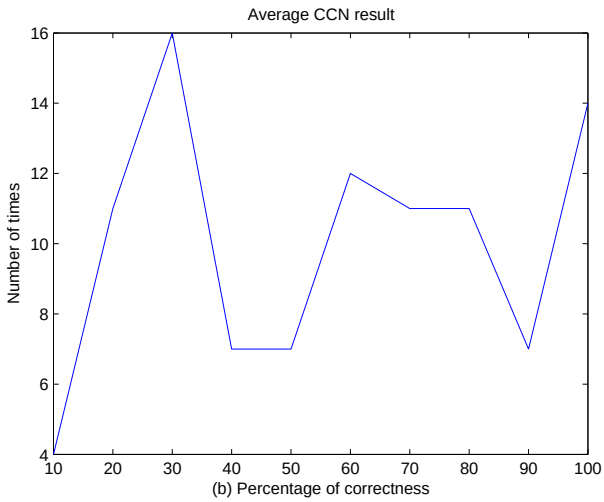

Fig. 6 – The average result of 100 runs where the percentage of performance is shown in the x-axis and the number of times the corresponding percentage achieved is shown in the y-axis.

We observe that full percentage (100%) of performance is achieved only 14 times (out of 100 times) and the low (10%) of performance is achieved only 4 times (out of 100 times). That means, we could excite or fire all (10) the CCNs in our experiment 14 times and only one CCN in 4 times for storage and retrieval of vectors. But these irregularities are alleviated by our proposed method when we perform experiments with the interconnections among the CCNs.

In order to get a broader view of how the system works for large networks, we changed the number of CCNs to N = 1000. We also have different values for the number of zones, namely, R has values in {10, 12, 14, 16, 18, 20, 22}. As can be understood by the intuition that the higher the number of zones, the fewer the error rate of correctly retrieving stored vectors. This is because, if the number of zones is higher then it will be less likely that two or more zones will have the same set of winning signals. The

algorithm in [1] does better when the number of zones in each CCN is higher. Figure 7 shows the graph for a network of 1000 CCNs where in the x-axis we have the values of different R and the y-axis shows the percentage of vectors correctly retrieved and stored by their algorithm [1]. The graph shows that as the number of zones increases the percentage of vectors manipulated (i.e., stored and retrieved) increases and vice versa. It can be observed from the graph that in almost all cases their algorithm achieves more than (96%) success. However, applying our algorithm results in correctly manipulating all the vectors. For each value of R, we conducted 100 runs to figure out the overall of the performance. Figure 8 shows the average results for different values of R. The performance of the average result is quite stable.

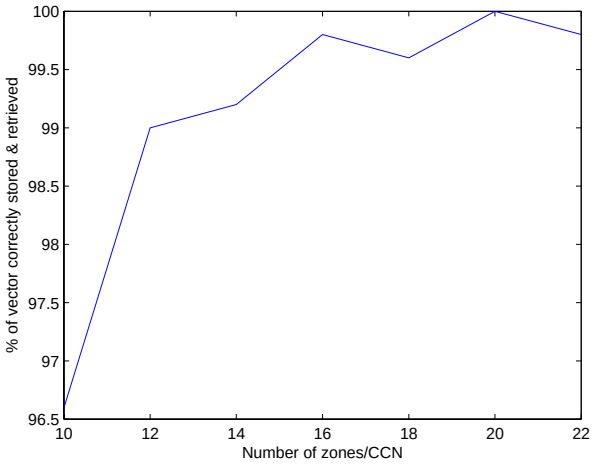

Fig. 7 – The graph shows the relationship between the number of zones and the percentage of correctly storing and retrieving the number of vectors.

7. FUTURE RESULTS

An improved and more general one-layer feedforward CCN network (more precisely associative memory network) depending upon the work of [1] has been introduced in the paper which can store R-dimensional input patterns in each of its neuron (each neuron has R-zones) and retrieve the whole vector with the presence of only a component of the input. We can further investigate the possibility of using recurrent network consisting of CCNs as recurrent network is well known and commonly used in the realm of neural network. In recurrent networks, the vector output of each of the neurons is fed back to that neuron's input lines, where each element of the output vector is fed back only to its corresponding zone. This ability may help us store and recall higher-order memories. Higher-order memory is one in which certain component of a vector can associate components or features with other vector. For example, consider the previous input pattern ("Sarah", "Female", "4781234"). After the network is trained, if we apply only the input "Sarah" we retrieve the whole vector ("Sarah", "Female", "4781234") as output. In the case of recurrent network we may feed the component "4781234" back to the appropriate zone which might result in recalling another vector with elements *School of computing*, *Queen's University*, *Canada* representing her department name, university name and the country where she is from.

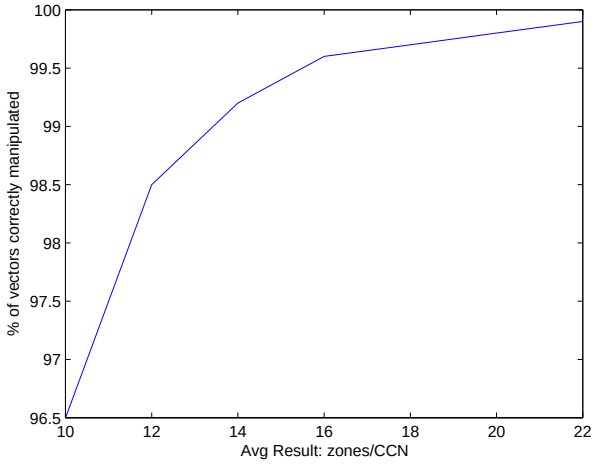

Fig. 8 – For each value of R, 100 runs were performed. y-axis shows the average of the percentage of manipulating vectors for each value of R.

Another important direction of this research could be identifying the data types of the elements of vectors and manipulating the data accordingly. Currently, we do not have the mechanism to separate different data types, namely identifying between string and integer types. It would be interesting if we could incorporate the data type separating mechanism in the network.

8. CONCLUSION

Motivated by the resemblance of a pyramidal cell [6] found in brain, the authors of [1] proposed a new type of model neuron (called CCN) to imitate the behavior of a pyramidal cell. The pyramidal cell is believed [7] to process both the frequency and the amplitude of the input signals and there is some sort of competition among inputs. Attempts are made to follow the physical structure and functional behavior of the pyramidal cell to some extent in the CCN, such as competition among the inputs and finally select the winner. In this paper, we provide an improvement to the CCN model [1] which is more generalized and can handle situation where there is a possibility of getting activated more than one CCN. Furthermore, we can also increase the memory with

our proposed modification to the architecture of the CCN. Thus the modified neuron model can increase the memory capacity substantially as demonstrated in this paper.

9. REFERENCES

[1] H. Bar. W. Miranker. A. Ambash. Competition and Cooperation in neural processing. *IEEE Transactions on Neural Networks* 53 (3) (2004).

[2] S. Haykin. *Neural Networks, a Comprehensive Foundation*. Upper Saddle River, NJ. Prentice-Hall, 1999.

[3] http://www.wikipedia.org/.

[4] J. Singh. *Great Ideas in Information Theory, Language and Cybernetics*. New York, Dover, 1966.

[5] D. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. New York, Wiley, 1949.

[6] M. Arbib. *The Metaphoricalbrain*. New York: Wiley-Interscience, 1972.

[7] L. Rutherford. S. Nelson. G. Turrigiano. BDNF has opposite effects on the quantal amplitude of pyramidal neuron and interneuron exciatory synapses. *Neuron*, 21, (1998), p. 521-530

Kamrul Islam *is currently pursuing Ph.D in the School of Computing in Queen's University, Kingston, Ontario, Canada where he obtained his Master's degree in 2005. His research interests include algorithm designs, complexity analysis in the field of Computational Geometry, Sensor Networks and Neural Networks.*