INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Lightweight, Fast and Efficient Distributed Hierarchical Graph Neuron-based Pattern Classifier

R. A. Raja Mahmood [1,*] , **A. H. Muhamad Amin** [1], and **A. I. Khan** [1]

[1] *Clayton School of Information Technology,*
Monash University, Victoria, Australia

Abstract: A lightweight, fast and efficient pattern classifier based on Distributed Hierarchical Graph Neuron (DHGN) is proposed and implemented. A classifier with such features is essential for the emerging networks such as wireless sensor networks and mobile ad hoc networks, where resources such as bandwidth and energy are limited. The proposed classifier adopts an in-network processing algorithm and has a one cycle learning capability. DHGN network is a new form of neural networks which consist of a hierarchical graph-based representation of input patterns. This paper compares the proposed solution with the well known Self-Organizing Map (SOM) classifier, in relation to accuracy and computational complexity. The results show that our solution offers lower computational complexity than SOM while guaranteeing satisfactory accuracy.

Keywords: Lightweight Classifier, Distributed Hierarchical Graph Neuron (DHGN), Self-Organizing Map (SOM), Computational Complexity.

1. Introduction

Pattern recognition is an important research field in a variety of scientific and engineering disciplines. The components of pattern recognition include preprocessing, feature selection, and extraction, classification and optimisation. Our research work focuses on the pattern classification area, in particular to propose and implement a lightweight, fast and efficient pattern classifier. In general, the function of a classifier is to separate a number of patterns into appropriate classes. Many of the existing classifiers adopt one of the following methods: (i) similarity-based or template matching, (ii) statistical or probabilistic, or (iii) error minimisation. Some examples of classifiers include Bayes rule, K-mean, K-nearest neighbour, Multilayer Perceptron (MLP), Self-Organizing Map (SOM), decision tree and Support Vector Machine (SVM) with Kohonen SOM [1] being the most prominent classifier and has been widely implemented in numerous applications [2–5]. Most of these classifiers including Kohonen SOM are iterative in nature and accurate. However, they are time consuming and resource intensive. Although Kohonen SOM has been proven effective and efficient as an intrusion detection mechanism, with a high percentage of detection rate and low false alarm rate, in wired environment [5,6] - with high support for computational resources - but it may not be practical in the resource constrained networks.

Wireless sensor networks and mobile ad hoc networks are the central parts of the emerging wireless networks. In recent years, there has been a rapid growth in research interest to explore the capabilities and limitations of these networks. These networks consist of tiny, low-powered battery devices with many limitations, including being unsecured and having limited resources in relation to their processing capacity, power, and memory. In relation to the security aspect of these wireless networks, it is known that the networks can easily be crippled by the infamous Distributed Denial of Service (DDOS) attack. Therefore, having an effective and efficient security

* Corresponding author.
Email address:
Raja.Mahmood@infotech.monash.edu.au.

mechanism, such as intrusion detection system (IDS) is critical for reliability of these networks. Adopting SOM approach in the IDS for these networks may not be practical due to its intensive usage of resources. Hence, there is a demand for not only an efficient classifier, but also a lightweight classifier for such networks. It is our aim to propose an accurate classifier with lightweight and fast response features through Distributed Hierarchical Graph Neuron (DHGN) algorithm. DHGN is a single-cycle pattern recognition algorithm that offers high recognition accuracy within short recall time [7]. Ultimately, it is our main goal to implement DHGN-based classifier as part of IDS solution in the Mobile Ad Hoc Networks (MANETs).

In this paper, we aim to show the efficiency and effectiveness of our proposed solution by comparing DHGN with Kohonen SOM. In particular, we investigate the classification accuracy and computational complexity of these algorithms. We also provide the average execution time taken for DHGN to classify the input patterns to emphasise its fast response time. This paper consists of 6 sections. Section 2 provides an overview of the Kohonen SOM and section 3 presents our proposed algorithm. Section 4 describes the data and method used in the experiments. Section 5 presents the experimental results and provides the discussion on the findings. Finally, Section 6 presents our conclusion.

2. Kohonen Self-Organizing Map

In general, Kohonen Self-Organizing Map (SOM) is a feedforward neural network that has the ability to learn the characteristics of similar items in a newly presented data set and group them into different classes or clusters. By mapping similar input values onto closely neighbouring neurons, SOM is able to preserve the original topological relationship among the objects in the data set. SOM also acts as a visualisation tool, where it is able to map a high-dimensional data set onto a lower dimensional space, normally resulting into a two dimensional map. For visualisation purpose, the unified-distance matrix (U-matrix) has frequently been used to provide the representation of the clusters formation and the cluster boundaries, as presented in Figure 1. The clusters can be seen as neurons of small distances (refers to the values on the colorbar) separated by the large-distance neurons.

The Kohonen SOM consists of two layers: the input layer and the Kohonen layer, with both layers fully interconnected. The input vector x_i and the weight vector w_i have the same number of dimensions, based on the number of variables in the data set under consideration. Unlike DHGN, SOM's learning algorithm is iterative. In the first step, all weight vectors of the Kohonen layer are initialised with random values. During each iteration, a single input neuron is randomly selected and the distance between this neuron and each neuron in the Kohonen layer is then calculated. Euclidean distance metric is usually used for this purpose. The Kohonen neuron with the least Euclidean distance to the selected input neuron, that is the closest to the selected input neuron, is chosen as the winner neuron or Best Matching Unit (BMU). Its weight vector is then moved towards the weight vector of the selected input neuron using the following formula for time k:

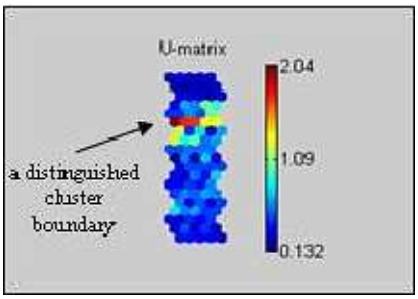

Figure 1. U-matrix of the IRIS data set [9].

$$w_i(k+1) = w_i(k) + \alpha * C * (x_i - w_i(k)) \quad (1)$$

$$C = e^{(-d_e^2/2*\sigma^2)} \quad (2)$$

The coefficient C describes the size of the neighbourhood around the winning neuron in Kohonen layer. Parameters α and σ are monotonically decreasing during the training, leading to convergence of the Kohonen layer. This process is iterated until a predefined number of cycles, hence a stable state is reached. Details of the algorithm can be found in [1]. As such iterative process requires extensive resources, SOM algorithm is not suitable for emerging wireless networks.

3. Distributed Hierarchical Graph Neuron

The Graph Neuron (GN) theory was first introduced in [10]. It uses a new form of neural network with the structure and its data representations are analogous to a directed graph. The processing nodes, termed as GN array, are mapped as a vertex set V of a graph, and the inter-node connections belong to the set of edges, E. Each node holds the $\{$*value, position*$\}$ pair information and the network represents all possible data points in the reference pattern space. Figure 2 shows the GN array of input pattern of size 5 bits, with input values of A, B and C. The dotted line represents the inter-node communication of pattern BABBC. The

communication overheads of GN remain low as the communications are restricted to the adjacent nodes only. The GN array compares the edges of the graph with subsequent inputs for memorisation (signifies a new input pattern) or recall (signifies an old pattern). It can be seen that the algorithm is able to detect patterns instantly, that is in only one cycle. However, the traditional GN suffers from the crosstalk phenomenon, therefore cannot be fully relied upon for accuracy.

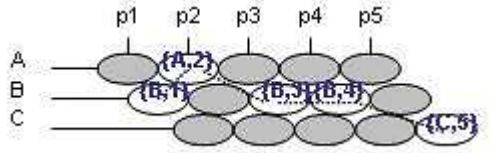

Figure 2. An input pattern BABBC in a GN array.

An improved version of the GN algorithm, termed Hierarchical GN (HGN), [8], was later developed to provide better recognition accuracy that is highly resilient to pattern distortions. HGN provides a bird's eye view of the overall pattern structure allowing it to accurately recognise both subpatterns and the entire pattern. However, the number of processing nodes significantly increases in HGN. As a comparison, the GN network requires only 15 nodes while HGN requires a total of 27 nodes to process an input pattern BABBC as shown in Figure 2. The network structures of GN and HGN arrays are depicted in Figure 3.

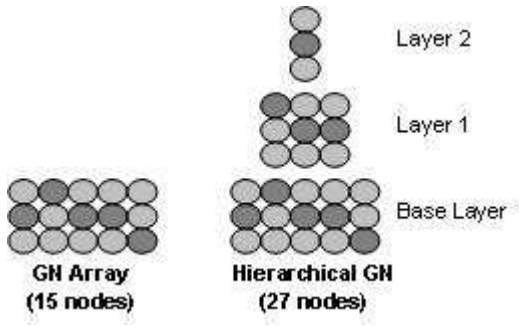

Figure 3. Comparison between GN and HGN network topologies for 3-element input pattern.

HGN was then implemented in a distributed platform, termed the Distributed HGN (DHGN)[7]. In DHGN, each input pattern is divided into subpatterns and these subpatterns are processed concurrently. In order to determine the status of these subpatterns, either as new or old subpatterns, these are compared against their respective bias indices. If such patterns are found in the bias indices, these are considered old patterns and are recalled as the matched indices. However, if the patterns are not found in the bias indices, these are considered as new patterns and their information is stored, with new indices being created. These subpatterns' indices are then collected by the decision-making module, which then identifies the class to which the input belongs to, based on a majority voting scheme.

DHGN maintains similar accuracy performance as HGN, and at the same time decreases the number of processing nodes required as well as the computational loads. For comparison, in order to process a 35-bit binary value input pattern, the HGN network requires a total of 648 nodes, while the DHGN algorithm divides the pattern into few smaller subpatterns and requires a total of only 126 nodes. It can be seen that DHGN significantly decreases the number of processing nodes required in the network. Moreover, the final index propagation process in HGN was eliminated in this implementation - that is the top index value is not propagated to all nodes in the network - and doing so further decreases the DHGN's computational complexity. Figure 4 shows the DHGN implementation, which consists of a DHGN array and the decision-making module. In this implementation, the 35-bit input pattern has been divided into seven subpatterns of 5 bits each. All of the subpatterns are then processed concurrently and the results, in the form of a list of indices, are passed to the decision-making module. The decision-making module then determines the class or cluster to which the input pattern belongs. One drawback of DHGN however is that it requires a bit of preprocessing, that is dividing the pattern into several subpatterns before any recognition process can take place. Simple PERL scripts have been used for this preprocessing purpose.

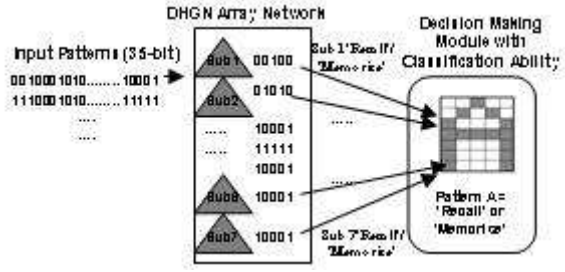

Figure 4. A 7x5 bitmap of letter A is mapped as subpatterns of 5-bit strings over 7 hierarchically formed GN sub-networks.

4. Experiments

Two series of experiments have been conducted to investigate the classification performance of SOM and DHGN in a supervised environment.

4.1. Test 1: Distorted images of characters I, S and A

The aim of this test is to investigate the accuracy of both algorithms in classifying distinct patterns with certain applied distortion. In this test, the distinct characters I, S and A were used. Supervised learning was deployed in this study, with the training data comprising only the perfect pattern of I, S, and A alphabets. The testing data sets consisted of three thousands distorted and six thousands distorted, I, S and A alphabets respectively. The distortion percentage ranges from 3 percents to 60 percents or from 1bit to 21bit were applied for each of these data sets. For the three thousand patterns data set, the training-testing data ratio used in this study was 3:3000 or 0.1-99.9, while the training-testing data ratio used for six thousand patterns data set was 3:6000 or 0.02-99.98. The characteristics of the data used in this study are presented in Table 1.

Table 1. Data and their characteristics.

Features	Test 1	Test 2
Alphabets	I, S and A	I, S, A, J, Z, H and X
Input Values	0 and 1	0 and 1
Input Size	5x7 bits (total = 35 bits)	5x7 bits (total = 35 bits)
Distortion Rate	1 bit (3%) - 21 bits (60%)	1 bit (3%) - 10 bits (29%)
Test Data	3000 characters (1000 for each alphabet) and 6000 characters (2000 for each alphabet)	2500 mixed distorted characters (I=500, S=500, A=500, J=400, Z=300, H=200, X=100

4.2. Test 2: Distorted images of characters I, S, A, J, Z, H and X

The aim of this test is two-fold. Firstly, we aim to investigate the accuracy of both algorithms in classifying not only distinct patterns, but also similar patterns with certain percentage of distortion been applied. In this test, I, S, A, J, Z, H, and X alphabets were used. Some of these patterns share similar structure, such as I and J in their 5x7 bits configurations. Secondly, we aim to show DHGN network capability to learn in one cycle when new pattern is introduced to the network. In SOM's case, the network needs to be retrained and weights need to be readjusted until a stabilised network is achieved whenever new training patterns are introduced to the network.

Supervised learning was deployed in this study, with the training data comprising only the perfect pattern of I, S, A, J, Z, H, and X alphabets. Firstly, the network was trained with alphabet I, S, A and then tested with 300 distorted patterns, comprising of 100 distorted patterns per alphabet. Secondly, a new pattern, alphabet J was introduced to the network and a total of 400 distorted patterns of I, S, A and J, comprising of 100 distorted patterns per alphabet, was tested. Thirdly, we introduced pattern Z and then a total of 500 distorted patterns of I, S, A, J and Z, comprising of 100 distorted patterns per alphabet, was tested. Fourthly, pattern H was introduced and a total of 600 distorted patterns of I, S, A, J, Z and H, comprising of 100 distorted patterns per alphabet, was tested. Finally, the network was introduced with pattern X and total of 700 distorted patterns of patterns I, S, A, J, Z, H and X, comprising of 100 distorted patterns per alphabet, was tested. Each dataset consists of 2500 distorted patterns, with the distribution of the alphabets listed in Table 1. The distortion percentage in this test ranges from 3 percents to 29 percents or from 1bit to 10bit. The training-testing data ratio used in this study was 7:2500 or 0.28-97.2. The characteristics of the data used in this study are presented in Table 1.

The SOM Toolbox [11] was used to investigate the classification accuracy of the supervised SOM. A SOM with a grid of 5 x 2 hexagonal nodes was created using the perfect alphabets training data set for Test 1, as shown in Figure 5. In Test 2, a SOM with a grid of 5x3 hexagonal nodes was created as shown in Figure 6. For DHGN algorithm implementation, we employed MPICH2 message passing library on C programming language, on a single Intel Pentium 66MHz with 128 MB RAM test machine. A total of 18 processes were required at one time to classify each 5-bit sub pattern. Hence, a total of 126 (18 processes x 7 subpatterns) processes were required altogether to classify and recognise one single pattern.

5. Results and Discussion

5.1. Classification Accuracy

5.1.1. Test 1: I, S and A characters

Figure 7 shows the classification accuracy of DHGN and SOM algorithms in detecting distinct pat-

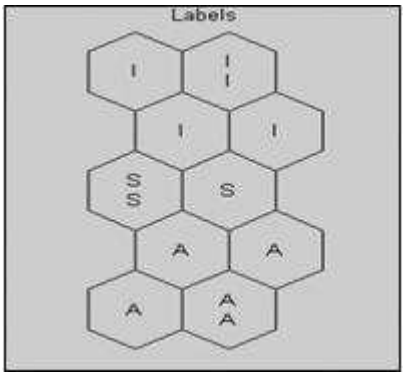

Figure 5. SOM of map size 5x2.

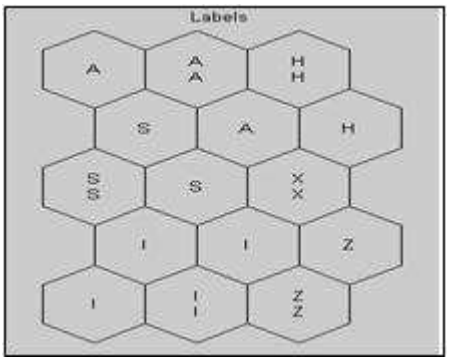

Figure 6. SOM of map size 5x3.

terns, in this case alphabet I, S and A. In general, both algorithms show comparable classification accuracy results for a range of 3% to 23% bits distortion, with more than 90% accuracy achieved. SOM is superior to that of DHGN for a range of 26% to 46% bits distortion with an average of 6.91% difference between the two and with the greatest difference of 11.2% at the 37% or 13 bits distortion. The high accuracy achievement by SOM requires a large number of iterations, usually in the thousands, to be performed during the learning process. SOM optimises its results by adjusting the weight of the neurons many times and only stops when the errors have been minimised, i.e. when a stabilised network state has been achieved. SOM has proven to achieve high classification accuracy results, especially under supervised learning in many studies, and hence these results are expected. Although DHGN performs better than SOM for distortion bits of 46% and more, we do not take this result into account since 50% distortion in any patterns is considered very high. With 50% or more distortion imposed on a pattern, the structural information of the pattern may have been lost.

We then increased the data set to 6000 distorted patterns and investigated the performance of both algorithms. The resulting graph, as depicted in Figure 8 shows similar pattern as the previously discussed graph (Figure 7). Similarly, the results for 6000

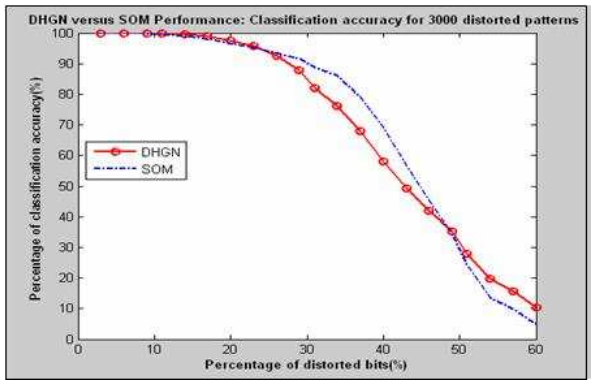

Figure 7. Comparison of classification accuracy between SOM and DHGN for alphabet I,S,A with 3000 distorted patterns of specified bits distortion.

distorted patterns show comparable result between DHGN and SOM for up to 23% bits distortion. On the average, SOM outperforms DHGN by 6.96% for a range of 26% to 46% bits distortion with the greatest difference of 11.2% at the 40% or 14 bits distortion.

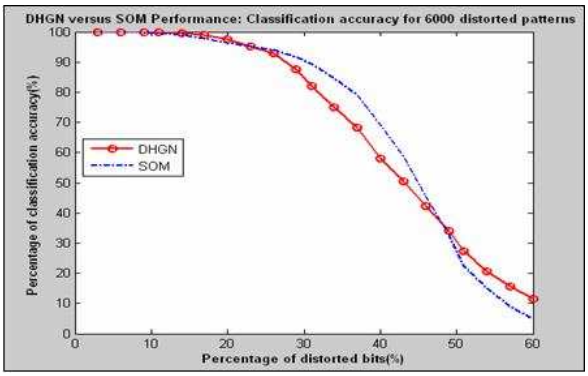

Figure 8. Comparison of classification accuracy between SOM and DHGN for alphabet I,S,A with 6000 distorted patterns of specified bits distortion.

With a difference average of only 6.91% (for 3000 distorted patterns) and 6.96% (for 6000 distorted patterns) in performance accuracy between these algorithms, DHGN's result is satisfactory considering that it employs a faster one-cycle learning process.

5.1.2. Test 2: I, S, A, J, Z, H and X characters

In this test, we increased the number of patterns to be classified with some of these patterns share similar structural information. The resulting graph, as depicted in Figure 9 shows that the accuracy of DHGN is superior to that of SOM. On the average, DHGN outperforms SOM by 17% with the greatest difference of 21% that occurs at the 20% or 7 bits distortion.

This particular result is promising for DHGN, but more rigorous testing using more patterns need to be done in the future for verification. We anticipated at the beginning that SOM would perform better or simi-

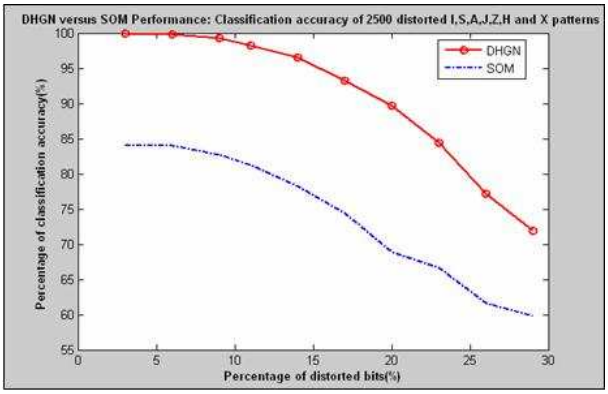

Figure 9. Comparison of classification accuracy between SOM and DHGN for alphabet I,S,A,J,Z,H,X with 2500 distorted patterns of specified bits distortion.

lar to DHGN in this test as well. The only explanation for SOM to perform poorly when similar alphabets were introduced in the network is perhaps the number of records taken for training dataset is very small. Unlike in the previous test, a very small training set is sufficient due to the distinct structural patterns of the classified alphabets. Perhaps in the future, SOM needs to be trained with distorted patterns as well in order to perform better in this test. Besides low accuracy percentage shown by SOM, extensive resources are also required to retrain its network in this test. For each new pattern been introduced to the network, SOM network needs to be retrained. This iterative activity of calculating BMUs and adjusting weights is computationally expensive, and definitely not feasible if it is to be done several times over short periods of time. In emerging networks, an on-line intrusion detection system (IDS) is commonly used, where network patterns or traffic conditions need to be updated in regular time periods in order to detect potential attacks. Using SOM for on-line IDS is not feasible as it is computationally expensive to retrain.

5.1.3. Quantization Error (QE)

Besides percentage of classification accuracy, Quantization Error (QE) is also used to measure SOM's performance - to measure the data representation accuracy. QE is computed as the means of the distances between the input vectors and their BMUs on the map. With default settings, the QE value generated by SOM Toolbox in Test 1 was 0.532. Whereas, the QE value for Test 2 was 1.549. We believe the QE value, can be reduced if better parameters configuration is used. The high error value in QE in Test 2 was reflected in the high error percentage of SOM classification as depicted in Figure 9.

5.1.4. Summary

In brief, the accuracy results between DHGN and SOM in the first test are comparable, while DHGN outperforms SOM in the second test. Besides high accuracy percentage shown by DHGN, it is a fast algorithm for it has a one-cycle learning capability. Hence, whenever a new pattern is introduced to the DHGN network, it gets learned instantly, while SOM requires iterative process for this learning and retraining to occur. Ultimately, DHGN algorithm saves more energy and time in comparison to SOM. Since DHGN is able to provide such capability, we believe DHGN-based classifier is a working solution for intrusion detection system in these networks.

5.2. Computational Complexity

The Big-O notations for both SOM and DHGN have been estimated to study their complexity levels. Big-O notation is a theoretical measure of the execution of an algorithm, usually the time or memory needed, given a specified problem size [12].

The supervised SOM consists of three important stages: (i) weight initialisation, (ii) BMU calculation, and (iii) weight adjustment. In the weight initialisation stage, nodes are created with random assigned weight. At this stage, the computational complexity depends heavily on the number of created nodes. Hence, for a given weight initialization process, w, the complexity of n nodes can be simplified as $f(w) = O(n^3)$. Figure 10 shows the estimated time taken to initialise up to 100,000 nodes. The estimated time derived is based on the assumption that the instruction speed used is 1 microsecond (μs) per instruction.

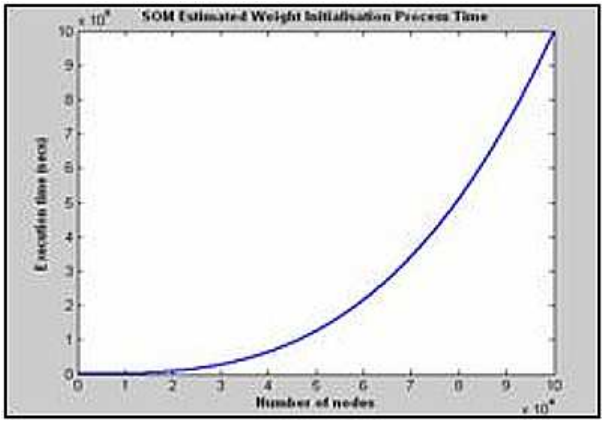

Figure 10. Complexity performance of SOM's weight initialisation process.

In the BMU calculation stage, the complexity depends heavily on the number of iterations during training as well as the number of the input vector. Hence,

for a given BMU calculation process, m, the complexity of training iterations, n, can be simplified as $f(m) = O(n^4)$. The estimated time taken to perform up to 100,000 iterations of calculating the Euclidean distance between the input values and all neurons in this stage is given in Figure 11.

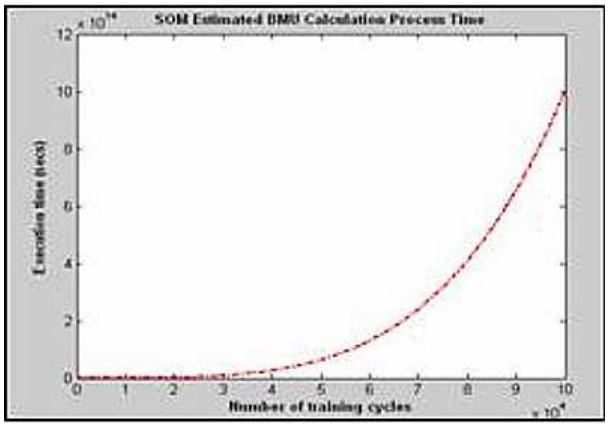

Figure 11. Complexity performance of SOM's BMU calculation process.

In the last stage, the weight adjustments are provided not only for the winning neuron but also for its neighbours in a certain neighbourhood. The degree of adjustment depends on the degree of similarity between the neuron and the input. As a result of weight adjustment, a group of neurons are obtained forming a cluster. The Big-O for the weight adjustment is similar to the BMU calculation (refers to Figure 11) and hence not provided.

In DHGN, the learning process consists of the following steps: (i) present input vector x in orderly manner to the network array, and (ii) compare the subpattern with the bias index of the affected node or neuron, and respond accordingly. There are two main processes in DHGN algorithm: (i) network initialisation, and (ii) classification. In the network initialisation stage, we are interested to find the number of created nodes and the number of initialised neurons. In DHGN, the number of generated neurons is directly related to the input pattern's size. However, only the neurons of the base layer of the hierarchy are initialised. Equation 3 shows the number of neurons in DHGN, N_{dhgn}, given the size of the pattern, S, the size of each DHGN array, R_{dhgn}, and the number of different elements within the pattern, n_e:

$$N_{dhgn} = n_e[\frac{\frac{S}{R_{dhgn}} + 1}{2}]^2 \quad (3)$$

The computational complexity for the network initialisation stage, G_{dhgn} for n number of iterations, n could be written as in the following equation:

$$f(G_{dhgn}) = O(n) \quad (4)$$

This equation proves that DHGN's initialisation stage is a low-computational process, and hence acquires less computational time in comparison to SOM's weight initialisation process. Figure 12 shows the estimated time for this process. Similar speed assumption of 1 microsecond (μs) per instruction is applied in this analysis. It can be seen that the time taken in the initialisation process of DHGN is far less than SOM. For instance, DHGN takes only 0.2 seconds (see Figure 12) while SOM takes about 3 x 108 seconds (see Figure 10) to initialise 20,000 nodes.

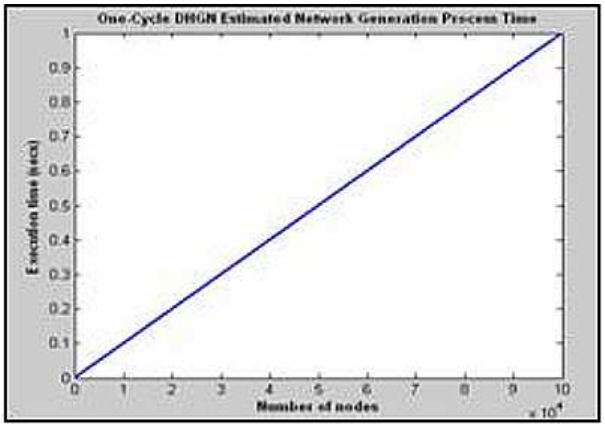

Figure 12. Complexity performance of DHGN's network generation process.

In the classification process, only few comparisons are made for each subpattern, i.e. comparing the input subpattern with the subpatterns of the respective bias index. The computational complexity for the classification process is somewhat similar to the network generation process, except an additional loop is required for the comparison purposes. The pseudo code of this process is as follows:

```
for each node in the network
{
   recognition()
   {
      for each bias entry
      {
        if input index = stored index
        {
         recall stored index
        }
        else
         store input index
      }
    }
    classification()
}
```

From this pseudo code, the complexity of the classification process C_{dhgn} for n number of iterations could be written as the following equation:

$$f(C_{dhgn}) = O(n^2) \quad (5)$$

It can be seen from Equation 5 that DHGN's classification process requires less computational complexity in comparison to SOM's BMU calculation and weight adjustment activities. For instance, the time taken for classification by DHGN in a network of 50,000 nodes is less than 3 seconds (see Figure 13) while SOM's BMU calculation process alone takes about 5 x 1013 seconds to complete (see Figure 11), and with similar time needed to perform weight adjustment.

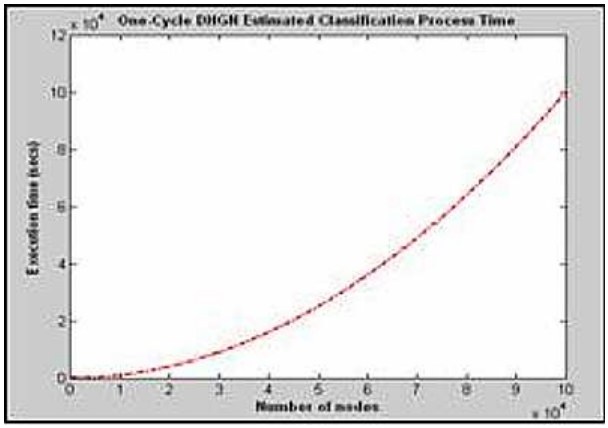

Figure 13. Complexity performance of DHGN's classification process.

In summary, the estimated time graph of the DHGN algorithm is linear, while the corresponding graph of SOM algorithm is exponential. This proves that the DHGN is an efficient algorithm - lightweight and fast - in comparison to SOM.

5.3. Execution Time

The DHGN algorithm takes advantage of the parallel computing. Figure 14 shows the average time taken by DHGN to classify one distorted pattern, which was between 12.25ms and 13.1ms. Since the classifying and recognition task was divided into several processes and working concurrently, the execution time has been reduced significantly. Taking into consideration the test machine that was used to run MPI and DHGN algorithm, we believe the time is relatively small and hence has shown that the DHGN is a fast classifier. A comparison in relation to execution time between SOM and DHGN is not feasible since SOM was tested on a high performance machine, that is on an Intel Pentium 2.8GHz, with 1GB of RAM.

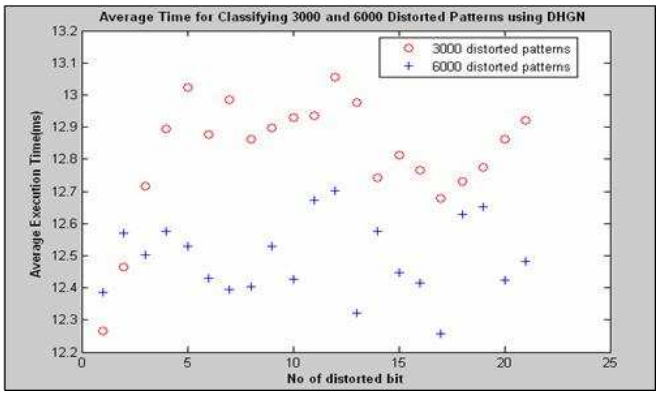

Figure 14. Average time taken for classifying 3000 and 6000 distorted patterns using DHGN.

6. Conclusion

The SOM and DHGN algorithms have been analysed and their performances have been presented. Unlike the iterative SOM algorithm, which is time consuming and resource intensive, DHGN algorithm adopts a faster one-cycle learning approach, and hence is a fast and lightweight classifier. This has been highlighted in the computational complexity results of both algorithms; the complexity of the DHGN algorithm is linear, while that of the SOM is exponential. In relation to the classification accuracy of distinct patterns, both algorithms show comparable results with SOM outperforms DHGN classification accuracy marginally by an average of less than 7%. However, in classifying similar patterns, DHGN outperforms SOM by an average of 17%. As SOM is computationally expensive, it requires massive resources to be implemented. Hence, SOM is impractical to be used in the resource-scarce wireless networks. We believe DHGN-based classifier is an effective solution for intrusion detection systems in these networks for it saves energy, time, and is comparatively accurate. In future work, we plan to perform more rigorous testing to validate our initial finding as well as employ our proposed algorithm for detecting DDOS attacks in mobile ad hoc networks.

References

[1] T. Kohonen, "Self-Organising Maps," Springer, Berlin, 2001

[2] Z. Chi, J. Wu, and H. Yan, "Handwritten numeral recognition using self-organizing maps and fuzzy rules", *Pattern Recognition* 28(1) 1995, pp.59-66.

[3] D. R. Chen, R. F. Chang, and Y. L. Huang, "Breast cancer diagnosis using self-organizing map for sonography", *Ultrasound in Medicine and Biology* Vol.26, 2000, pp. 405-411.

[4] J. Huysmans, B. Baesens, J. Vanthienen, and T. Van Gestel, ”Failure prediction with self organizing maps”, *Expert Systems with Applications* 30 (3), 2006, pp. 479-487.
[5] A. J. Hoglund, K. Hatonen, and A.S. Sorvari, ”A Computer Host-Based User Anomaly Detection System Using the Self-Organizing Map,” In: *IEEE-INNS-ENNS Intl Joint Conference on Neural Networks (IJCNN’00)*, vol. 5, 2000.
[6] P. Lichodzijewski, A. N. Zincir-Heywood, and M. I. Heywood, ”Host-based intrusion detection using self-organizing maps”, In: *Proc.of the IEEE International Joint Conference on Neural Networks*, May 2002.
[7] A. H. Muhamad Amin and A. I. Khan, ”Parallel Pattern Recognition Using a Single-Cycle Learning Approach within Wireless Sensor Networks”, In: *Proc. of the Ninth Intl. Conf. on Parallel and Distributed Computing, Applications, and Technologies (PDCAT) 2008*, IEEE Computer Society, December 2008.
[8] B. B. Nasution and A. I. Khan, ”A Hierarchical Graph Neuron Scheme for Real-Time Pattern Recognition,” In: *IEEE Transactions on Neural Networks*, vol. 19, pp. 212-229, 2008.
[9] E. Anderson, ”The irises of the Gaspe Peninsula”. Bulletin of the American Iris Society 59, 1953, pp. 2-5.
[10] A. I. Khan, ”A peer-to-peer associative memory network for intelligent information systems,” In: *Proc. of the Thirteenth Australasian Conf. on Information Systems*, vol. 1, 2002.
[11] J. Vesanto, E. Alhoniemi, J. Himberg, K. Kiviluoto, and J. Parviainen ”Self-Organizing Map for Data Mining in Matlab: The SOM Toolbox,” Simulation News Europe, vol. 25, 1999.
[12] P. E. Black, ”big-O notation”, in Dictionary of Algorithms and Data Structures [online], Paul E. Black, ed., U.S. National Institute of Standards and Technology. 2 November 2007. (accessed 16/05/2008) Available from: http://www.nist.gov/dads/HTML/bigOnotation.html