A Multi-Feature Pattern Recognition for P2P-based System Using In-network Associative Memory

Amiza Amir[1], Anang Hudaya M.Amin[2], and Asad Khan[1]

[1] Clayton School of IT, Monash University, Melbourne, Australia
{amiza.amir,asad.khan}@monash.edu
[2] Universiti Teknologi PETRONAS, Malaysia
ananghudaya@petronas.com.my

Abstract. Associative memory enables recall through invoking associations between past experiences in memory. In this paper, we demonstrate our fully distributed associative memory approach, the Distributed Associative Memory Tree (DASMET), to deal with multi-feature recognition in a peer-to-peer(P2P)-based system. The scheme constructs logical tree like structures within a peer-to-peer network enabling the nodes to search for patterns comprising multiple temporal or spatial features within a fixed number of steps using in-network processing. In doing so, the information held at individual peers is integrated into a common knowledge base which can be associatively searched by any peer within the network. We show that our scheme is fault tolerent and incurs low complexity overhead. By comparing our scheme to Backpropagation network and Radial Basis Function (RBF) network on two standard datasets, we prove its scalability and accuracy. Practically, this work has many advantages for the P2P domain. For example, various aspects such as file piracy issues, sensitive data leaking, and content pollution problem may be addressed.

1 Introduction

Peer-to-peer (P2P)-based systems encompass powerful capabilities but have been negatively viewed due to copyright infringement issues and more recently for sensitive data leaking problems. We aim to provide pattern recognition capabilities towards solving some problems in P2P networks (e.g. file pollution, copyright infringement, security issues, etc). As this involves dealing with complex datasets such as images, audios, and videos, significant effort is required to analyse these datasets. This limits the current methods recognition capability. Analysing a complex dataset within a peer is infeasible (usually an expensive process) since each peer is usually only willing to share a limited fraction of its available resource.

Current approaches in pattern recognition require significant amount of effort to analyse different forms of features. This limits their ability to seamlessly and effectively perform recognition and classification involving complex datasets.

Current schemes [10, 8, 1] incur high computational complexity that inhibits their capability to scale up to handle a large number of features.

Multiple-feature implementations enable a holistic approach towards pattern recognition. They consider all significant features which represent a particular set of pattern, such as images and sensor readings. This tends to reduce the bias effect of selecting only a single feature for classification/recognition. Multi-feature pattern recognition is useful in the P2P domain where objects are maybe processed by different types of analysis (e.g. frequency and spectral analysis) and different types of forms (e.g. sound, vision, and text) at different locations.

Existing methods of on multi-feature recognition have been able to produce high recognition accuracy given the increased number of features considered. These include [3, 19, 18, 9]. Nevertheless, this results in increased complexity of overall recognition scheme. For instance, the multi-feature face detection scheme proposed by [19] implements probabilistic-based AdaBoost [7] to train different features for recognition. The scheme does not scale well due to iterative probabilistic interpretation process which needs to converge these features to global minimum. In other research, [3] have proposed multiple-expert systems by adopting an iterative clustering algorithm. Furthermore, [5] have also considered the use of combined classifiers involving multiple features. In their research, a number of classifiers, including statistical methods, machine learning methods and neural networks have been applied. Nevertheless, that incur significantly-high computational costs.

Our underlying approach requires that all peers share their memory and processing power in recognizing objects or patterns. Each peer is responsible to sense, interpret and remember perceived sensations and work collaboratively with other peers to remember the object. A peer can join and leave a P2P network easily without facing any bureaucracy. Thus some P2P networks can scale up to hundreds of thousands or even to millions of peers. This enables access to a large amount of information over the internet. Therefore encourages understanding of the world and improves our problem solving ability [17].

In this paper, we attempt to design a scalable and efficient pattern recognition scheme with respect to time and space complexity. This is done by harnessing resource pooling within the P2P network. By considerably a set of distributed computational networks working together, forming a distributed recognition network can alleviate the issue of scalability against the increasing number of features to be considered.

2 The Graph Neuron and The Hierarchical Graph Neuron

Our approach is adapted from the Graph Neuron (GN). The Graph Neuron is a pattern recognition scheme for a wireless sensor environments. Other than its readily distributed nature, it is straight forward for us to adapt in our problem, the strength of this scheme is its simplicity and single learning cycle.

The Graph Neuron is a connected graph with a set of vertices, V and a set of edges, E. The network represents all possible points in representing a pattern. Each node, called a *neuron*, holds the $\{u, s\}$ pair information, where u represents the domain value associated with the neuron and the s represents the position of the neuron in the pattern space.

Fig. 1 shows the GN structure as introduced in [13] where we consider an input pattern of size 7 bits, with input values {0,1}.

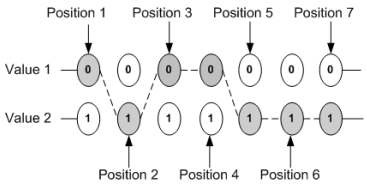


Fig. 1. A flat GN structure for a binary pattern with domain values: {0,1} and pattern size 7. The possible values was represented by the number of rows while the size of the pattern is represented by the number of columns.

The dotted line represents the inter-node communication of the pattern **0100111**. Each neuron communicates with its adjacent nodes being the left and right neighbours. The value corresponding to a specific position in the input pattern is mapped and compared with the neuron at the same position which holds the same value for memorisation (signifies a new input pattern) or recall (signifies an old pattern). This scheme has been applied in WSN applications where its quick recall time and high scalability characteristics have been proven [12].

The Hierarchical Graph Neuron (HGN) [15] is an improvement on the GN algorithm. The crosstalk problem of the GN algorithm resulting from the limited local knowledge at each node (see [15] for more details) is eliminated in the HGN by constructing a hierarchical structure so that the neuron at the top can perceive the overall pattern. However, the number of processing nodes and communication costs have significantly increased in this scheme.

3 The Distributed Associative Memory Tree (DASMET)

Our approach is viewed as a multi-sensory system in which each peer in the P2P system acts as a sensor. Each peer is only responsible to process a subset of features. These results are then combined iteratively in a tree structure until converge at the root. Therefore, an increasing number of features can be easily handled by recruiting more peers [3] into the associative memory network. Since

[3] which is not a problem in most P2P systems which usually have more than hundred thousands of nodes available

all subsets of features are processed simultaneously, the learning time remains low resulting in a scalable algorithm capable of handling high dimensional multi-feature datasets.

In DASMET, the responsibility to interpret the result is handled through the collaboration of peers. This is usually handle by a single base station in other GN-based scheme. Thus, DASMET is a fully distributed associative memory algorithm to facilitate the accurate, efficient and scalable pattern classification. It inherits the single cycle learning and accuracy characteristics of the HGN but significantly reduces the number of nodes and communications required by the HGN. The trade-off is that a single node in DASMET has to undertake a greater workload than a node in the HGN. However, the magnitude of this workload is reasonable for a small portion of resources of a peer.

3.1 Structure of DASMET

The structure of DASMET is a rooted tree $G = (V, E)$, where V is the set of vertices and E is the set of edges (see Fig. 2). Each vertex in a DASMET is an entity or a computational unit, x, that is $x \in V$. The structure depends on the maximum number of children of each node, φ. This is equal to the maximum input connections (in-degree) allowed for a node. The number of leaves, n_H is equal to the number of different feature sets. n_H may also be set to the number of feature subsets which are split from the original set of features. This is to reduce the dimensionality of the feature set. We classify the entities in DASMET into three roles: 1) The rootnode which is responsible to finalise the result; 2) The leadernodes, which are internal nodes responsible to collect and combine the results from nodes at a lower level and propagate the results to nodes at a higher level; and 3) The hubnodes which are leaf nodes responsible to recognize or classify a feature set locally and independently.

Each feature set is processed by a hubnode. The results from all hubnodes are propagated to nodes at the next higher level. The memorised/recall processes are successively performed by nodes at each level until the final result is calculated at the rootnode. The computational load amongst the nodes is maintained by growing the DASMET tree. This increases the scalability of the algorithm when handling high number of hubnodes.

Vertices are added with edges connecting to parent node and the hubnodes is distributed among these nodes. We start the DASMET tree construction with a rootnode and several hubnodes. These hubnodes become the children of the rootnode. Now we require that each node may only have maximum of φ children. In order to achieve this, we create intermediate nodes. We create as many levels of the tree given the limit of children. Fig. 2 shows an example of a DASMET structure for 8 different feature sets.

Given the length of pattern ρ and the number of nodes at level i, n_i is at most φ^i. The height of an DASMET tree $\hbar$ is $\hbar \geq log_{\varphi} n_H$. The total number of

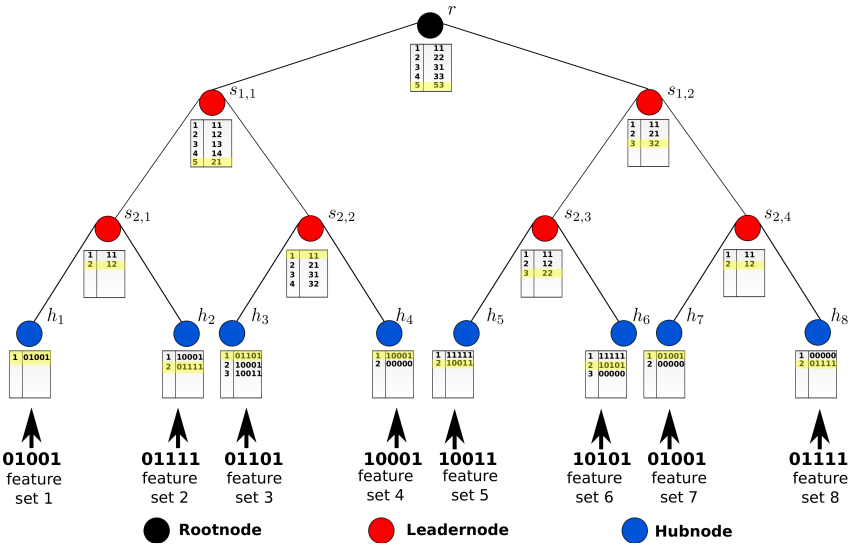


Fig. 2. Example of DASMET architecture for 8 different feature sets. The predefined maximum children of each node, φ is 2. Each feature set is assigned to a hubnode respectively. It has 8 hubnodes and 6 leadernodes with the height of the tree is 3. Each node holds a *biasArray* (represented by the box) which stores the memory being the learnt pattern. The yellow coloured entries represent the active entries for the input pattern.

nodes in DASMET, $|V|$ is:

$$|V| \leq \sum_{l=0}^{\hbar-1} \varphi^l + n_H \tag{1}$$

The total number of edges in DASMET, $|E|$ is equal to $|V| - 1$.

Each node holds a table called the *biasArray* where each entry in the table represents the memory of a pattern that has been allocated to each node. The format of the *biasArray* is varible dependent on the node's role. An entry of the *biasArray* for a hubnode is the value of the pattern's segment. While for a leadernode or rootnode, the entry for each column in a *biasArray* consists of a pair of $\langle biasIndex, position \rangle$ from its child. *position* is the position of a node from the parent's view and the information associated with the *position* is called as the *biasIndex*. This index for column j in the *biasArray* of a leadernode or a rootnode is the index of the active entry from its child at the position jth. Therefore the column number in the *biasArray* is equal to the number of children it has.

3.2 Store/Recall Procedure

The nodes in DASMET perform a basic store/recall procedure similar to the GN model [13].

Hubnode

Store Upon receiving an input feature from the *client*, the hubnode responds by performing a lookup through the *biasArray* and if the matched input feature is not found, it is stored as a new entry. If found it is recalled and there is no new entry inserted into the *biasArray*. The index of the matched row is then sent to the parent node.

Recall Create a temporary structure called a *voteTable*, which holds the similarity (vote score) of the pattern to recall with the stored patterns in *biasArray*. A vote is calculated by obtaining the frequency of a matched element in each column j in the *biasEntry* with corresponding column j in each row i of the *biasArray*. An element j is considered as matched when the value of its adjacent positions itself have the same value with a row. Now, the row with highest vote is obtained and sent the corresponding index of the node at upper level. Since ties with the highest votes are possible, the indices of all rows with highest vote are sent to a parent node.

Leadernode and Rootnode

Store A leadernode/rootnode receives k reports from its children (formed as *biasEntry*) where k is equal to the number of children it has and $k \leq \varphi$. The *biasEntry* that is received from these children is then compared to each row in the *biasArray*. If it is not found then a new memory corresponding to this entry is inserted into the *biasArray*. This new index is then sent to the parent node.

Recall The recall procedure in the leadernode/rootnode involves comparing indices that are received from its children with the pertaining elements of the *biasArray*. Similar voting procedures as performed in hubnode are performed here. The final decision at a rootnode is made by determining the majority class with highest vote.

3.3 Complexity Analysis

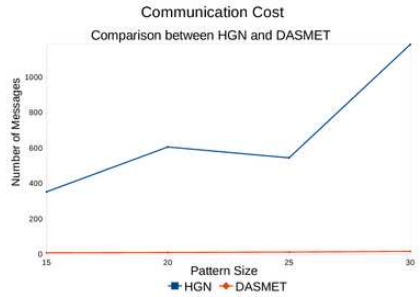
Since the algorithm works in parallel, the time requirement of the storing is the sum of the processing involved from the deepest leaf node to the rootnode. This includes the communication within DASMET network[4]. Let the number of levels is $(\hbar + 1)$ and the communication time to send each message is T_c. Let n be the

[4] where height of the tree $\hbar$ is equal to the number of required edges from a hubnode to a rootnode

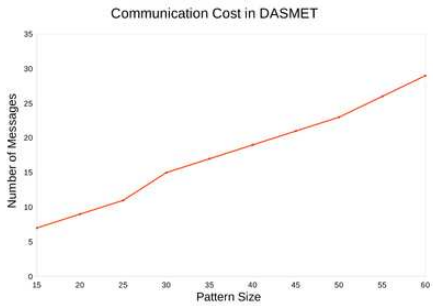
size of *biasArray*, m is the number of classes and number of features to be handled by a hubnode is at most $log_2\ \rho$. The estimate of storing time with respect to the total number of features ρ is $O\,(q + w)$ where $q = (\hbar) \times (n\ log_2\ \rho) + (n\ log_2\ \rho + m)$ and $w = \hbar T_c$.

3.4 Communication Cost

Total communication during store/recall in DASMET is the total number of edges in the DASMET structure.



(a) Comparison of communication cost in HGN and DASMET



(b) Communication cost in DASMET

Fig. 3. The DASMET structure in this example is defined by $\varphi = 5$. Fig. 3(a) shows that the number of messages required in HGN increases significantly compared to DASMET with increasing pattern size. Fig. 3(b) shows the small increase of communication cost in DASMET.

We plot the number of messages that is required in the HGN and the DASMET as shown as in the Fig. 3. The bigger size of the pattern results in dramatic increases of messages in the HGN. This occur since the number of nodes increases linearly with the pattern size. In DASMET, a node handles several positions of

the pattern size unlike the HGN in which each node only handles a position of the pattern space. Fig. 3 shows the number of messages in the DASMET remains relatively low and increases very slowly.

4 P2P Overlay Platform: Kademlia

Here, we use Kademlia [14] is used as the P2P platform. In Kademlia, stale nodes or nodes which are often connected for a short period of time are removed from the routing table. This provides us with a mechanism to build a strong network with less unstable peers. Using the uptime information, we can effectively assign main jobs to more stable peers. Kademlia uses tree-based and prefix-based routing table, similar to Pastry [16]. In contrast to the Pastry's recursive routing, Kademlia uses iterative routing, where a peer contacts the closest peers to the lookup key by incrementally track peers with smaller XOR distances to the lookup key[5]. Thus it is asymmetric. The latency is logarithmically increasing with an increased number of peers demonstrating the scalability of Kademlia.

5 Handling a Dynamic Network

Abrupt peers leaving the network affect the availability of the *biasArray*s and impede the system from working correctly. The local storage in our scheme is a dynamic entity with frequent data updates. Therefore, we implement consistent and secure backup updating using the byzantine fault tolerance replication protocol [4] to ensure data consistency and freshness at all peers. We use replication method for data redundancy across the network to ensure data availability and scheduled maintenance mechanism to maintain the availability of data.

Data Replication We ensure the reliability and availability of DASMET nodes by replicating the information at several peers. By creating r replications of a DASMET node on r peers, we can reduce the possibility of losing bias array information up to r factor. A peer maybe responsible for a particular session before failure or go offline and another peer will be able to replace it without a significant degradation to the system. Each DASMET node has a backup called a *Shadow* and r replications of itself. We called the original node as *Source* and a backup node as *Shadow*. The *Shadow* is responsible to take over the *Source* role whenever the *Source* fails. Each *Source* has other backups called replicas. In order to synchronize local replicas, consistent communication and control mechanisms are provided by byzantine fault tolerance replication protocols [4]. The *Source* and its backups are placed at different peers.

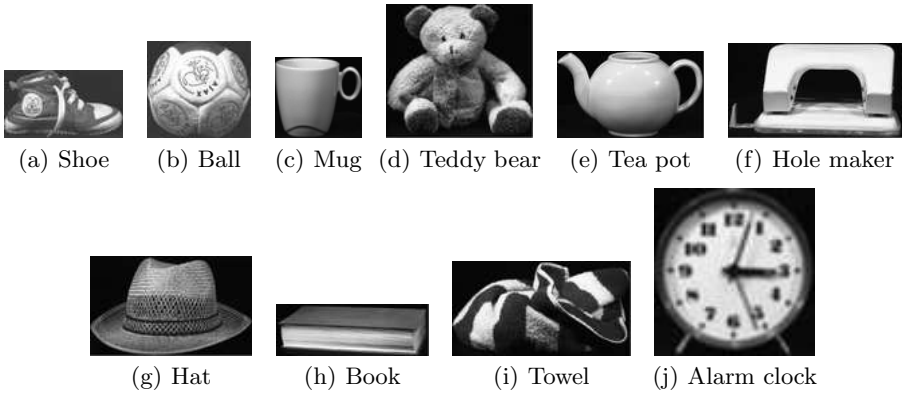
Periodically Recovery Mechanism The DASMET network periodically recovers through a scheduled recovery mechanism. All nodes in the DASMET network work collaboratively to detect any change to the system and perform a

[5] XOR of two node's identifier is the distance between them

node's recovery if needed. All children nodes act as failure detectors of the parent node. The parent node of a hubnode acts as a failure detector of the hubnode. Each *Source Peer* except the rootnode's *Source Peer* periodically contacts its parent to ensure that it is alive. For a hubnode's *Source Peer*, its parent detects a failure by ensuring all children (hubnode's *Source Peer*s) constantly communicate with it. The hubnode's *Source Peer*s do not contact the parent within certain duration, the parent contacts the hubnode to check if it is still alive. All detected failure are reported to the *Shadow Peer*. When a *Shadow Peer* receives a failure report, it contacts the *Source Peer*. If there is no response from *Source Peer*, the *Shadow Peer* takes over the *Source Peer*'s role. The node selection algorithm is then performed to select a suitable candidate to replace itself for a *Shadow Peer* position. The *Shadow Peer* might also go offline. In this situation, the replicas of the DASMET node can be found amongst close neighbours of *Source Peer*.

6 Experimental Result

To test the accuracy of DASMET in dealing with multi-feature datasets, we used ten training images [11] (as shown in Fig. 4). The collection we use in this experiment is obtained from The Amsterdam Library of Object Images [11]. During preprocessing, these images were resampled into 30×50 and then converted into binary format. The resulting number of features for this datasets is 1500.



(a) Shoe (b) Ball (c) Mug (d) Teddy bear (e) Tea pot (f) Hole maker

(g) Hat (h) Book (i) Towel (j) Alarm clock

Fig. 4. Example of ten raw images for accuracy testing.

In this experiment, we divide the original feature set into 300 feature subsets consisting of 5 features each. The number of maximum children a parent node may have was set to 5. Hence, we have a DASMET structure which has 406 nodes and height of 5. We simulated our scheme within the network with 1000 peers.

For each of these images, we created 1000 versions by adding random distortions to the original image. For example, 5% distortion is created as follows. We select by 5% of the pixels in a binary image of interest and flip all '0's to '1's and vice versa. Starting from the first 1000 versions with 10% random distortion, we generated the next 1000 versions with 15% distortion and then slowly increase the distortion rate up to 50%. Then we compile the distorted versions into five testing datasets (2000, 4000, 6000, 8000, and 10000) where 10% from the dataset are the distorted versions of each image. Therefore, in total we have 45 datasets. Fig. 5 demonstrates the accuracy of our distributed algorithm. We conclude that

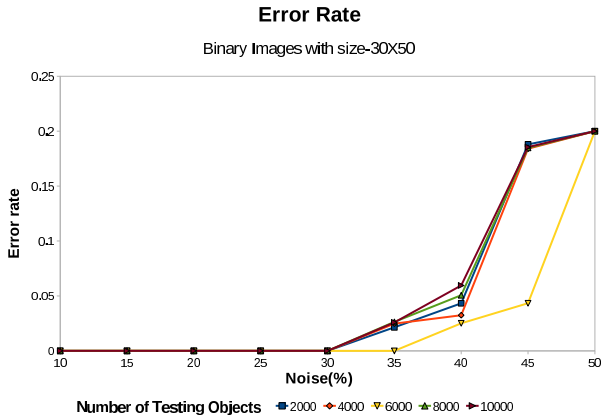


Fig. 5. Figure shows the error rate in 30×50 binary images recognition by using the DASMET. The error rate remains low with increasing distortion rate.

the error rate in DASMET is low with only 5% incorrect prediction when given a pattern with a very high distortion which can consists up to 35% noise. The error rate starts to increase significantly from 40% distortion. We obtained consistent results when testing the algorithm with different size of testing datasets (from 2000 to $10,000$).

To show the suitability of our scheme for P2P networks, we restricted each processing node in our simulations to a single machine with at most 2GB RAM. Due to this resource restriction, we are unable to test this dataset for comparison with BackPropagation network [8] and Radial Basis Function (RBF) network [2] since these methods require large amount of memory to process the large dataset. Therefore, to show the accuracy of our distributed scheme is compared to the state-of-the-art neural networks which run on a single site, we compare using two smaller datasets. They are Adult and Spam datasets from UCI Data Mining Repository [6]. The Adult dataset has 14 features and the Spam dataset has 57 features. The results are presented in the Table 1 below.

Table 1. Accuracy comparison of DASMET with single site implementations of neural networks

Methods	Adult (14 features)	Spam (57 features)
Backpropagation Network	84.5113	92.709
RBF Network	85.7704	92.665
DASMET	79	98.308

Our distributed approach performs worse than the Backpropagation network and the RBF network for Adult dataset. However, it provides better results for the Spam dataset. Note that it is not our intention to outperform other methods in this work but to show that our scheme can be easily distributed over P2P network with comparable accuracy to a centralized scheme. As shown in Fig. 6, our simulation result also shows the store time increases slowly with the increase of a stored pattern reflecting the scalability of our scheme.

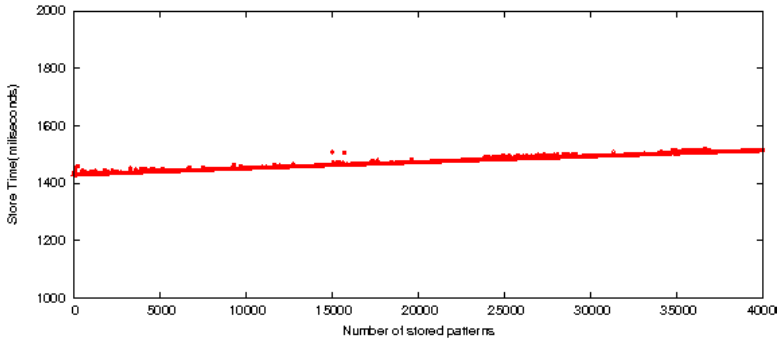


Fig. 6. Store time per input pattern with increasing number of stored patterns.

7 Conclusion

In this paper, we introduce and analyse a variant of the Graph Neuron algoritm for multi-feature pattern recognition within a P2P network. The number of processing nodes and communications are significantly improved in our approach compared to the HGN algorithm. Improved scalability is achieved through distributing and linking a massive number of training objects/memories over a P2P network which may involve hundreds of thousands to millions of nodes. By breaking the dimension of pattern into smaller parts, computing these parts separately in parallel and combining the recognition result iteratively in a tree

structure, we significantly reduce the computation complexity at a node in terms of space and time. Space complexity at each peer remains low as several different patterns might have multiple repeating same sub-patterns and thus a repeating sub-pattern is only stored once (at the first time it occurs). This approach uses single cycle learning as objects in a P2P environment are massive and frequently updated. In future work, we intend to incorporate a load balancing mechanism to allocate the tasks and data efficiently within a P2P network.

References

[1] Browne, M., Ghidary, S.S., Mayer, N.M.: Convolutional neural networks for image processing with applications in mobile robotics. In: Speech, Audio, Image and Biomedical Signal Processing using Neural Networks, pp. 327–349 (2008)

[2] Buhmann, M.D.: Radial Basis Functions: Theory and Implementations. Cambridge University Press (2003)

[3] Cao, J., Ahmadi, M., Shridhar, M.: Recognition of handwritten numerals with multiple feature and multistage classifier. Pattern Recognition 28(2), 153–160 (1995)

[4] Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Transactions on Computer Systems (TOCS) 20(4), 398–461 (Nov 2002)

[5] Duin, R.P.W., Tax, D.M.J.: Experiments with classifier combining rules. In: MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems. pp. 16–29. Springer-Verlag, London, UK (2000)

[6] Frank, A., Asuncion, A.: UCI machine learning repository (2010), http://archive.ics.uci.edu/ml

[7] Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: Proceedings of the Second European Conference on Computational Learning Theory. pp. 23–37. Springer-Verlag, London, UK (1995), http://dl.acm.org/citation.cfm?id=646943.712093

[8] Hecht-Nielsen, R.: Theory of the backpropagation neural network. In: Neural Networks, 1989. IJCNN., International Joint Conference on. pp. 593–605 (Jun 1989), http://dx.doi.org/10.1109/IJCNN.1989.118638

[9] Hongtao, S., Feng, D.D., Rong-chun, Z.: Face recognition using multi-feature and radial basis function network. In: VIP '02: Selected papers from the 2002 Pan-Sydney workshop on Visualisation. pp. 51–57. Australian Computer Society, Inc., Darlinghurst, Australia, Australia (2002)

[10] Hopfield, J.J.: Neural networks and physical system with emergent collective computational properties. pp. 2554–2558 (1982)

[11] Jan-Mark Geusebroek, G.J.B., Smeulders, A.W.M.: The amsterdam library of object images. International Journal of Computer Vision 61(1), 103–122 (2005)

[12] Khan, A.I., Baqer, M., Baig, Z.A.: Implementing a graph neuron array for pattern recognition within unstructured wireless sensor networks. Lecture Notes in Computer Science pp. 208–217 (2006)

[13] Khan, A.I.: A peer-to-peer associative memory for intelligent information systems. In: Proceeding Thirteenth Australasian Conference on Information Systems. vol. 1 (2002)

[14] Maymounkov, P., Mazières, D.: Kademlia: A peer-to-peer information system based on the xor metric. In: Revised Papers from the First International Workshop on Peer-to-Peer Systems. pp. 53–65. IPTPS '01, Springer-Verlag, London, UK (2002), http://portal.acm.org/citation.cfm?id=646334.687801

[15] Nasution, B., Khan, A.I.: A hierarchical graph neuron scheme for real-time pattern recognition. IEEE Transactions on Neural Networks pp. 212–229 (2008)
[16] Rowstron, A., Druschel., P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems (2001)
[17] Solomonoff, R.: The time scale of artificial intelligence; reflections on social effects. Human Systems Management 5, 149–153 (1985)
[18] Yu, D., Ma, L., Lu, H.: Lottery digit recognition based on multi-features. In: Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE. pp. 1–4 (2007)
[19] Zhu, H., Zhang, L., Sun, H., Xiao, R.: Face detection using multi-feature. In: Advances in Cognitive Neurodynamics ICCN 2007, pp. 921–925. Springer Netherlands (2008)