An FPGA-implemented Associative-memory Based Online Learning Method

Fengwei An, Hans Jürgen Mattausch, Tetsushi Koide

***Abstract*— In this work, we propose an FPGA implemented associative-memory-based lazy learning method with a short/long-term memory concept which has an online learning capability. The conventional lazy or memory-based learning algorithm has low computational costs in the training stage since it simply stores the inputs and defers processing of training data until a query needs to be answered. However, it requires large storage capability, often pays high computational costs to answer a request, and is easily fooled by irrelevant attributes. To speed-up the query time, which is a deficiency of conventional lazy learning, we apply an associative memory which is implemented in an FPGA for searching the most similar data among the previously stored reference data in parallel. The proposed learning method can eliminate irrelevant attributes and reduce the storage requirements through a forgetting process in the short-term memory. When the online lazy learning model is applied to handwritten character recognition, the mismatch rate is reduced to 0.6% after a learning process with 450 alphabet letters learning and the number of reference is reduced by about 40% in comparison to the conventional method.**

***Index Terms*—FPGA implemented associative memory, short/long-term memory concept.**

I. Introduction

In artificial intelligence, lazy learning has attracted a widespread attention since the end of last century. It simply stores the training data rather than compile them into an intensional description by a rule set or neural network for future requests [1]. Therefore, the lazy learning is very fast in training data, can learn complex functions to yield highly adaptive behavior, and does not lose information. Many industrial applications and robot control systems have employed the lazy learning approach to solve learning tasks [2-5]. However, this learning method needs large memory storage for the training data and has high computational costs for finding relevant data in memory to answer a particular requirement. This type of lazy learning is also referred to as memory-based or instance-based learning. Relevance of possible answers is usually measured using a distance function where the nearest distance has the highest relevance. Distance is often measured by using the Manhattan or Euclidean distance metric which depends on the type of the relevance attribute.

Manuscript received March 22, 2011.

Fengwei An, Hans Jürgen Mattausch and Tetsushi Koide is with Research Institute for Nanodevices and Bio systems (TEL: +81-82-424-6265 FAX: +81-82-424-6265 EMAIL: anfengwei, hjm, koide@hiroshima-u.ac.jp).

Given a set of feature-values for instances in memory, a prediction function has to distinguish the similarity by the distance. In the lazy learning method, local prediction functions such as the k Nearest Neighbor algorithm (KNN) can be used to fit all of the training data. Many algorithms are to find an optimal number of neighbors efficiently [6-8]. However, training the neighborhood size k leads to high complexity and costs. The previous works on lazy learning algorithms have difficulties to deal with the case of sudden changes in the input distribution. The first contribution of this work is that we develop a fully-parallel associative memory for reducing the query time to find the nearest distance which is taken as a combination of Manhattan and Hamming distance.

Methods to overcome the drawbacks of the lazy learning have been studied for many years. Daelemans et al. introduced an algorithm which uses decision trees to compress the pattern memory by removing feature-value redundancy to reduce storage requirements and query time [9]. The lazy learning retains all the training data which are represented by feature-values. When many features are irrelevant to the performance task, the lazy learner or classifier would be fooled. Thus many researchers have proposed approaches to decrease the influence of irrelevant attributes. A parallel racing algorithm has significant speed advantages for both model and feature selection tasks [10]. The context-sensitive feature selection algorithm allows distinguishing the relevance of features [11]. These previous works focus on distinguishing the task characteristics using a specific weighting function. We propose a short/long-term memory-based learning model with an online learning capability to eliminate the irrelevant feature attributes and to reduce the storage requirements.

In this paper, we first introduce the fully-parallel mixed digital/analog architecture for searching the nearest Manhattan and Hamming distance and the FPGA-implemented associative memory for searching the nearest Euclidean distance. In section III, we illustrate the proposed learning model based on short/long-term memory. This learning model can achieve an online learning capability by the forgetting process. We then apply our lazy learning model to handwritten character recognition in section IV. To reduce the feature complexity, we propose a novel gradient-based feature extraction method in section V. The experimental results in section VI are to show the efficiency and accuracy of the proposed online lazy learning method. At last, we conclude in section VII.

II. FPGA Implementation of Associative Memory Architecture

A. VLSI implementation

The convention methods for searching nearest distance often calculate between the input and reference patterns based on a general purpose computer in a sequential processing. Mixed digital/analog solutions for a fully-parallel associative memory have been reported for finding the nearest Manhattan and Hamming distance in the previous work [12, 13]. Currently, we introduce a time-domain concept for fully-parallel nearest hamming distance searching [14]. It maps the distance into time-domain with adjustable ring oscillators. The associative memory with time-domain Winner-Take-All circuit, which is scalable to small design rules and low supply voltages, detects the first arrived ring oscillator signal that is referred to as the winner (nearest-match) [15]. The time domain nearest Manhattan distance searching circuit will be developed in our future work.

The Manhattan distance is defined as in

$$D_i = \sum_{1 \le j < \infty}^{i < \infty} | SW_j - REF_{ij} | \quad (1)$$

where SW= {SW_1, SW_2, ..., SW_W} is the input data and REF_{ij}={REF_{i1}, REF_{i2}, ..., REF_{iW}} is the reference data. Both SW and REF are W-bit length integers. The Hamming distance is obtained when SW_j and REF_{ij} are 1-bit data.

B. FGPA implementation

We also have developed the VLSI circuit for searching the nearest Euclidean distance by the mixed digital/analog solution [16]. However, because of the computational complexity which needs many multipliers for the Euclidean distance computing, we implement the associative memory for searching nearest Euclidean distance in an FGPA which is a programmable VLSI device. The Euclidean distance is defined as

$$D_i = \sqrt{\sum_{1 \le j < \infty}^{i < \infty} (SW_j - REF_{ij})^2} \quad (2)$$

. Both SW and REF are real number. Figure 1 illustrates the parallel architecture of the associative memory which is implemented in the FPGA. The row RAM is used to store the reference features where the dimension of the corresponding vector determines its storage requirements. All the calculations are based on a fix-point decimal format. As a result, the floating-point calculation is avoided. The registers, which are used to the pipeline the arithmetic operations, help to improve the clock frequency so that high processing performance are obtained. Because the square root is difficult to be implemented, we only calculate the sum and simplify (2) to

$$D_i = \sum_{1 \le j < \infty}^{i < \infty} (SW_j - REF_{ij})^2 \quad (3)$$

This simplification has no influences on the results of the searching the minimum.

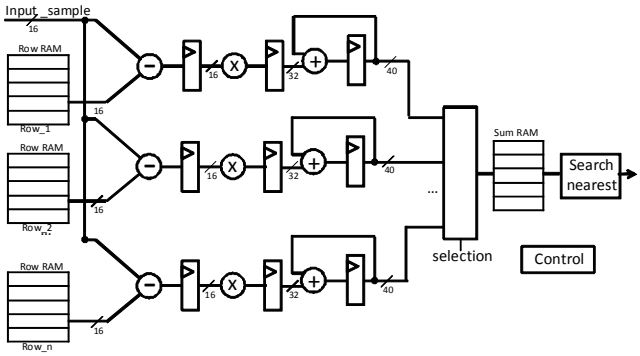

Figure 1. The associativ-memory's FPGA implementation for searching nearest Euclidean distance

However, the searching time of the above architecture is increasing when more rows for reference are added. For example, if the associative memory has 1024 rows, the processing of nearest distance search after the calculations for each row needs 1024 clock cycles when the memory is full. Therefore, we divided the searching for the nearest distance into 2 stages. We define the above architecture which only has 32 rows. In the first stage, there are 32 blocks where each block contains the architecture to search 32 rows in parallel. The second stage searches the nearest distance from the local nearest distances which are provided by the first stage. Therefore, the final searching time of the parallel associative memory is only 1/32 times in comparison to a single stage solution according to Fig.2. Obviously, the final output of FPGA-implemented associative memory is the address which has a minimum value.

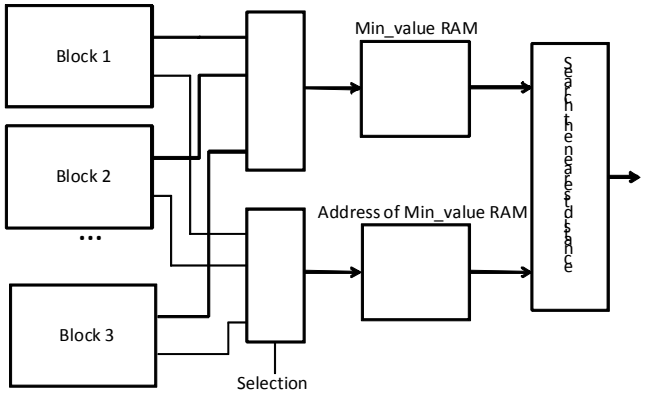

Figure 2. The parallel architecture of the associativ-memory's FPGA implementation for searching nearest Euclidean distance: Each block contains the arehitecture in Figure 1.

The FPGA implementation has lower speed and higher power consumptions than a realization of VLSI ASIC, because the FPGA is a reconfigurable device. However, the FPGA-implemented associative –memory is flexible to the versatile feature vectors for the different applications.

III. Learning Algorithm Based on Short/Long-term Memory Concept

The learning algorithm based on a short-term and a long-term memory has been proposed for the pattern recognition previously [17, 18]. The memorized reference data are classified into two parts according to their ranks in

this concept. The rank, which can determine whether the reference data is in short-term or long-term memory, may be equivalent the address of each reference data. The top address has the highest rank which implies it belongs to the long-term memory.

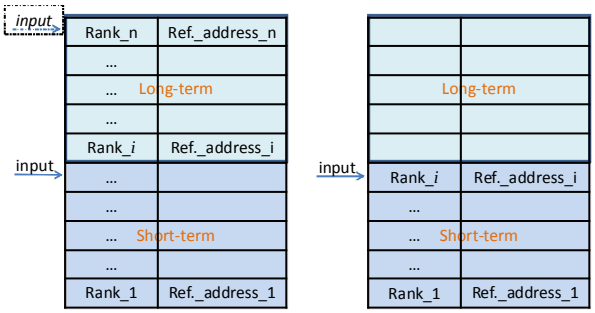

Figure 3. Two input types of short/long-term memory mechanism for memorizing the reference.

The reference data are temporarily memorized in the short-term part and they can be memorized for a longer time without receiving the direct influence of input patterns in the long-term memory. The transition of reference data between short-term and long-term storage parts is carried out by a ranking algorithm. There are two types of ranking approaches as shown in Fig.3. The difference of the two approaches is the way of inputting data to memory. In other words, it is the influence to the long-term memory from input data. One approach is that the input data are inserted in the top address of the short-term memory. The previous stored contents should move down to make way for the new input. The rank of previous stored data goes down one by one. As a result, the reference data of lowest rank is deleted and that leads to a forgetting process in Fig.4 when the short-term memory is full.

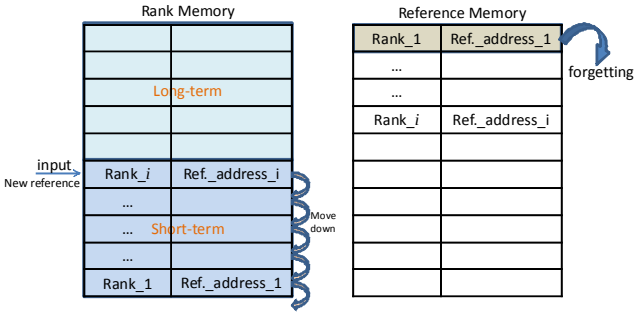

Figure 4. The forgetting process when a new sample is added in the reference memory

The input data are firstly stored to the long-term memory in the other approach. When the long-term memory is full, the input port is changed to short-term part. The following processing is same to the first approach. The forgetting process not only discards the content in short-term memory but also deletes the reference pattern in the reference memory. The discarded address of reference data may not be the inserted data at the beginning because the ranking algorithm changes the original order in the short/long-term memory.

The ranking algorithm is illustrated in Fig.5.The rank is the index of a memory which contains the address of a reference. The rank is moved up when a new input is matched with the current reference pattern. The occurrence of matching is determined by the winner which is detected by the associative memory. If the winner distance is less than a given distance threshold value, we can consider this is an occurrence of matching that gives a new rank to the matched pattern. The new rank is obtained from a predefined jump value. The jump value is an adjustable variable to fit different application. Therefore, the new rank of an occurrence of matching is obtained by subtracting a jump value from the current index. It must be made clear that the lower index (address) of memory represents a higher rank. When the winner distance is more than a threshold value, there is no occurrence of matching. The non-matched sample is added as a new reference in the memory and gets the top rank in the short-term or long-term memory.

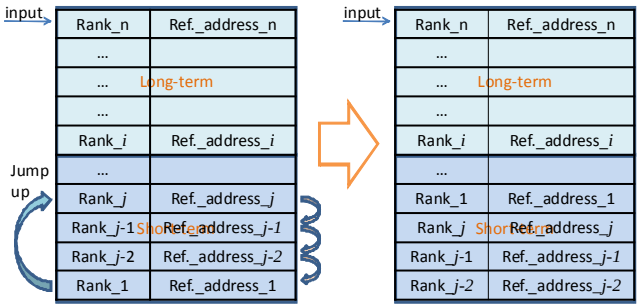

Figure 5. The ranking algorithm to reduce the storage and increase the relaibility of reference.

The boundary between short-term and long-term memory is used to judge whether the rank is short-term or long-term. Empirical results demonstrate that the learning capability is related to the size of short-term storage.

IV. Description of Online Learning Method

As described in the introduction, the lazy learning needs a prediction function to distinguish the similarity by distance. In this work, a combination of Manhattan and Hamming distance is used. The hardware approach is based on an associative memory with fully-parallel nearest distance search for enabling low power and real-time applications. The short/long-term memory learning model, which has an online learning capability [19], is to classify the training data.

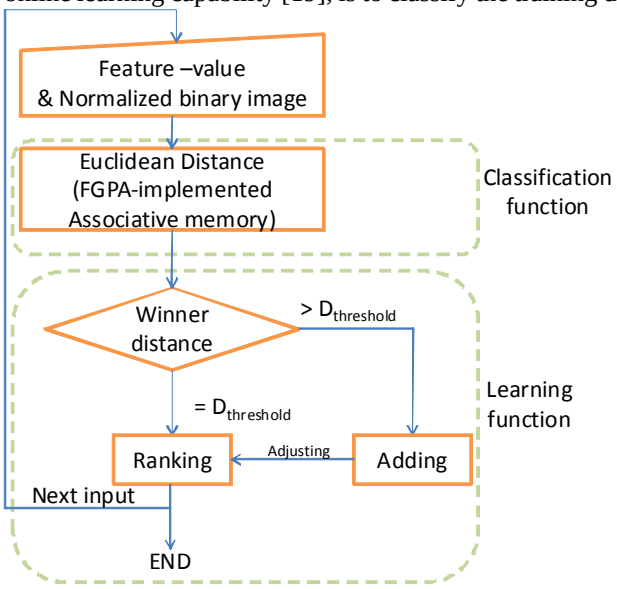

Figure 6. The demonstration of online learning model

The core part in this learning model is the classification function and the short/long-term memory. If the input matches a reference pattern in memory that makes the rank of this matched reference pattern jump up to a predefined location. Otherwise, the new input is inserted into the reference memory and the ranking algorithm gives it the top rank of the long-term or short-term memory as mentioned in section 3. Meanwhile, the location at lowest rank is forgotten and it also is deleted from the reference memory. With new reference data added into memory, an online learning capability is achieved and this also triggers the ranking algorithm to forget the lowest rank when the short-term memory is full. This online learning algorithm is illustrated in the flowchart of Fig.6.

The advantages of the proposed online lazy learning method are eliminating the irrelevant feature attributes and reducing the storage requirements through the forgetting process in the short-term memory. The other superiority is that the online learning based on an FPGA implementation structure can be used in real-time applications such as robot vision and video image recognition.

V. Character Recognition

We apply our online learning method to handwritten character recognition which attracts much attention to verify machine learning algorithms. In this paper, the handwritten alphabetic letters and digits from different writers are used to be learned and recognized.

A. Pre-processing

All handwritten characters are extracted from scanned images. The size and position of every character are not always alike. Thus, a preprocessing algorithm is used which contains noise removal, binarizing, labeling, segmentation and normalization.

Noise Removal: Because the characters are obtained from scanned images, the noise is certainly inevitable. In this work, to avoid the damage of each character's connectivity which affects the results of the labeling step, the 3×3 GAUSSIAN filter is used to remove noise. We only use this filter in the simulation as it is difficult to be implemented in hardware.

Binarizing: OTSU's method is applied for binarizing in this work. In OTSU's method, the image is classified into foreground and background and an optimum threshold is used to separate those two classes. This step prepares a binary image for labeling the connected components.

Labeling: The labeling which gives a label number to the binary connected components, is based on 4-connectivity. The binary image is scanned from left to right and top to bottom. The Floyd-Warshall algorithm [20] is used to resolve equivalences.

Segmentation: The connected binary components obtain a label number after the labeling step. Therefore, they can be distinguished by the label number. Each connected segment is stored as one image that represents a separated character.

Normalization: Each segmented character image is different from the others in size and position. Therefore, the character binary images are rescaled to 16×16 pixels by the bilinear interpolation method. A thresholding process is used to guarantee a binary output image for feature extraction in this step.

B. Feature extraction

An extracted feature vector is used to calculate the Manhattan distance between the input and reference patterns. We apply a gradient feature which is constructed only from the direction of the greatest change in intensity in a small neighborhood of each pixel. The resulting gradient feature maps are divided into 8×8 blocks and the direction is independently obtained in each block.

The direction gradient features are computed by convolving two 3×3 Sobel operators on the normalized binary image. These Sobel operators are used to calculate the horizontal x and vertical y derivatives in the image.

The gradient in x and y direction of a center pixel is obtained by a function including its eight neighbors as follows. The expressions of both horizontal x and vertical y derivatives are $S_x(i, j)=I(i-1, j+1)+2I(i, j+1)+I(i+1, j+1)-I(i-1, j-1)-2I(i, j-1)-I(i+1, j-1)$ *and* $S_y=I(i-1, j-1)+2I(i-1, j)+I(i-1, j+1)-I(i+1, j-1)-2I(i+1, j)-I(i+1, j+1)$, where $I(i, j)$ represents the pixel value in location (i, j). The 16×16 image has been divided into an 8×8 grid where each grid point contains 4 pixels. The directions at each grid point according to (3) constitute the feature vector which is stored in the reference memory. The accumulations in x and y direction of the 4 pixels are used as horizontal and vertical derivatives to compute the gradient direction of each grid point.

$$\theta = \arctan \frac{\sum S_y(i, j)}{\sum S_x(i, j)} \tag{3}$$

VI. Experimental results

A. The resource usage of the FPGA implementation

The architecture as mentioned above is implemented on the ALTERA Stratix series FPGA. Because the FPGA platform can be connected with a host PC by the PCI-e bus, it is possible to verify the architecture of the associative memory for searching the nearest Euclidean distance as illustrated in Fig.7. The resource usages after synthesis of the architecture in Fig.1 are shown in Table I.

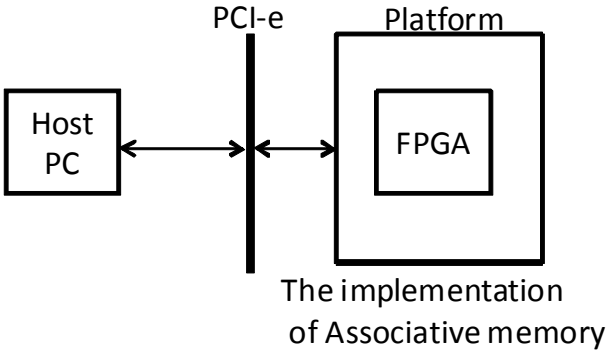

Figure 7. The verification of associative memory for searching the nearest Euclidean distance with Host PC and PCI-e bus.

TABLE I. Resource Usage after Synthesis of FPGA-implemented Associative Memory

Logic Units	Usage of FPGA Resources
Combinational Look Up Table	19931 (14%)
Dedicated logic	5924 (4%)

Registers	
Total Block Memory bits	1280 bits (<1%)
Synthesized Frequency	114.81MHz

The FPGA-implemented associative memory can achieve a 114.81MHz frequency where each clock cycle takes about 8.71ns. The feature vector, which has been calculated in Section V-B, has 64 dimensions, which implies that the calculation of the summary needs 256 (64*4) clock cycles. The nearest distance searching stage spends 32 clock cycles to find the local minimum from 32 rows references. Therefore the search time of 32 rows is about 2508.48ns.

B. *Classification results*

In order to verify the online learning capability, experiments are performed using two databases where the first one contains 450 alphabet letters written by different writers (Fig.8) and the second one is a standard handwritten-digit database called MNIST (Fig.9) with 60000 training samples and 10000 test samples.

Figure 8. Excerpt from the handwritten alphabet-letters database.

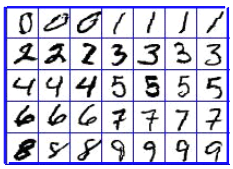

Figure 9. Excerpt from the handwritten-digit database:MNIST.

As described above, the distance of the winner is the nearest Euclidean distance searched by the associative memory. The Euclidean distance is computed by comparing the gradient features which are 64 dimension vectors (see section V-B).

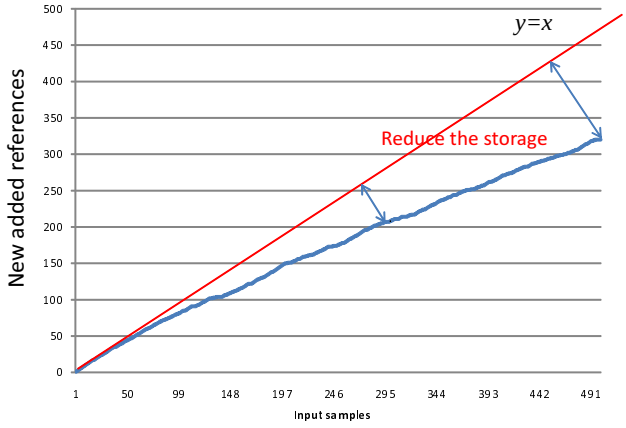

Figure 10. Relation of input samples and added new references to show the reduced storage.

The lazy learning method based on associative memory is applied to handwritten character recognition without initial reference data to verify the online capability. A pattern is added (learned) when the winner according to the hybrid distance measurement has a smaller distance than a predefined threshold value. In addition, each occurrence of matching increases the rank of the respective learned reference to become a more stable reference. After a period of learning, a number of references have been added to the reference memory or jumped to a stable area in the long-term memory. Meanwhile, the rate of newly added references and mismatched input data are decreasing. We examine the learning capability with the relation of input samples and added new references in Fig.10 where it illustrates that the number of reference is reduced by about 40%.

Along with an increase of the learning capability, the rate of new added patterns and mismatched patterns is decreasing. Because the alphabet letters are more complex than digits, the mismatch rate of digits is also lower than that of alphabet letters as confirmed by the experimental results.

TABLE II. CLASSIFICATION RESULTS

Stage	New Added	Misclassification	matched
1	206 (68.7%)	1.3%	94
2	76 (50.7%)	0.6%	74

We investigate the efficiency of the forgetting process to reduce the storage and to decrease the possibility of being fooled by irrelevant feature attributes. Suppose that, the first stage learning contains 300 alphabet characters and that the remaining 150 characters are used to test the classification results as shown in Table II. There are 206 characters from the first database which are online learned and a misclassification rate of 1.3% in 94 occurrences of matching. The rate of new learned references and misclassification are respectively decreased to 50.7% and 0.6% after only 300 characters have been learned firstly.

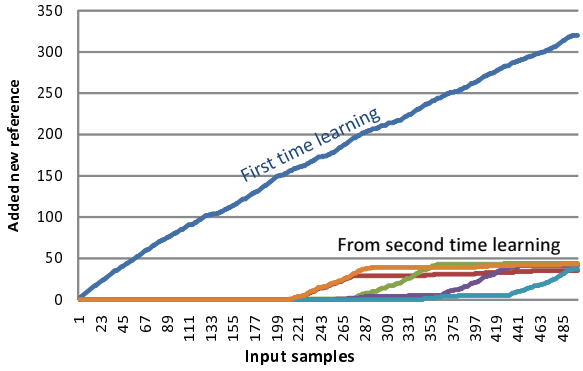

Figure 11. Forgetting process to increase the reliability of the reference patterns and reduce the storage.

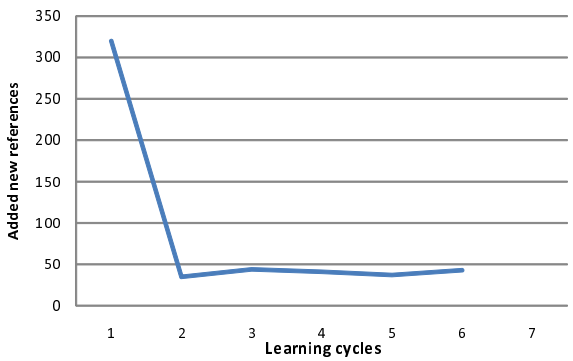

Figure 12. The relation of learning cycles and the new added references: from the second cycles the new added referecne is becoming stable.

In the short/long-term memory learning mechanism, the new learned references are added if the memory is not full. Otherwise, learning implies forgetting the lowest rank in short-term memory. The 450 characters are learned again and again. The reference data in memory are evolved to reduce the irrelevant features as shown in Fig.11.

Supposed the storage of short/long-term memory is less than the new learned patterns. After a number of learning cycles, the new learned patterns become less and less as shown in Fig.12. At last, the number of references is stable with a small amount of changes.

However, the sufficient references stored in the memory can attain a higher accuracy rate. Therefore, we present the FPGA-implemented associative memory which is reconfigurable to provide enough references.

VII. Conclusion

We presented the FPGA-implemented associative memory architecture, an online lazy learning based on this associative-memory and an application to handwritten character recognition. The core part is a learning concept with a short/long-term memory which reduces the storage and the number of irrelevant attributes by a forgetting process. This learning model can be operated without any initial reference patterns or a predefined database, which indicates an online learning capability. The online learning model can be used also in real-time applications because of the implementation possibility with a high-performance hardware structure.

References

[1] D. W. Aha, “Lazy Learning", The Artificial Intelligence Review, Vol.11, No.1-5, pp. 7-10, 1997.

[2] Jabbour K., Riveros J.F.V., Landsbergen D.,Meye W., "ALFA: automated load orecasting assistant", IEEE Transaction on Power System, Vol.3, pp. 908-914, 1988.

[3] R.H. Creecy, B.M. Masand, S.J. Smith, D.L. Waltz, “Trading MIPS and memory for knowlege engineering: classifying census returns on the Connection Machine", Communications of the ACM, Vol.35, No.8, pp. 48-63, 1992.

[4] T. Nguyen, M. Czerwinski, D. Lee, “COMPAQ QuickSource: Providing the Consumer with the Power of Artificial intelligence", Proceeding of the Fifth Conference on Innovative Applications of AI, pp. 142-151, 1993.

[5] G. Bontempi, M. Birattari, H.Bersini, "Lazy Learning for local modeling and control design", International Journal of Control, Vol.72, No.7-8, pp. 643–658, 1999.

[6] C. Böhm, S. Berchtold, D. Keim, "Searching in high-dimensional spaces-index structures for improving the performance of multimedia databases", ACM Computing Surveys, Vol.33, No.3, pp322-373, 2001.

[7] D. Cantone, A. Ferro, A. Pulvirenti, D. Reforgiato, D. Shasha, "Antipole indexing to support range search and k-nearest neighbor on metric spaces", IEEE Trans. on Knowledge and Data Engineering, Vol.17, No.4, pp. 535-550, 2005.

[8] T. Liu, A. Moore, A. Gray, “New algorithm for efficient high-dimensional nonparametric classification", Journal Machine Learning Research, Vol.7, pp1135-1158,2006.

[9] W. Daelemans, A.V.D Bosch, T. Weijters, “IGTree: Using Trees for Compression and Classification in Lazy Learning Algorithms", Artificial Intelligence Review,Vol.11, No.1-5, pp. 407-423, 1997.

[10] O. Maron, A.W. Moore, “The Racing Algorithm: Model Selection for Lazy Learners", Artificial Intelligence Review, Vol.11, No.1-5, pp. 193-225, 1997.

[11] P. Domingos, “Control-Sensitive Feature Selection for Lazy Learners", Artificial Intelligence Review, Vol.11, No.1-5, pp. 227-253, 1997.

[12] H.J. Mattausch, T. Gyohten, Y. Soda, T. Koede, "Compact Associative-Memory Architecture with Fully-Parallel Search Capability for the Minimum Hamming distance", IEEE Journal of Solid-State Circuits, Vol.37,pp218-227, 2002.

[13] Y. Yano, T. Koide, H.J. Mattausch, “Associative Memory with Fully Parallel Nearest-Manhattan-Distance Search for Low-Power Real-Time Single-Chip Applications", Proceedings of the Asia and South Pacific Design Automation Conference, pp. 543-544, 2004.

[14] H.J. Mattausch, W. Imafuku, T. Ansari, A. Kawata, T. Koide, "Low-Power Word-Parallel Nearest-Hamming-Distance Search Circuit based on Frequency Mapping", 36th European Solid-State Circuits Conference, pp. 538-541, 2010.

[15] M. Yasuda, T. Ansari, W. Imafuku, A. Kawabata, T. Koide, H.J. Mattausch, "Low-Complexity Time-Domain Winner-Take-All Circuit with High Time-Difference Resolution Limitted only by With-In-Die Variation",Solid State Devices and Materials, in press, 2010.

[16] Md.A. Abedin, Y. Tanaka, A. Ahmadi, T. Koide, H.J. Mattausch, "Nearest Euclidean-Distance-Search Associative Memory Architeture with Fully Parallel Mixed Digital-Analog Match Circuitry", Extended Abstracts of 2006 International Conference on Solid State Devices and Materials, pp. 282-283, 2006.

[17] Y. Shirakawa, M. Mizokami, T. Koide, H.J. Mattausch, “Automatic Pattern-Learning Architecture Based on Associative Memory and Short/Long Term Storage Concept",Ext. Abst. of SSDM2004, pp. 362-363, 2004.

[18] A. Ahmadi, H.J. Mattausch, M.A. Abedin, T. Koide, Y. Shirakawa, A. Ritonga, "Developing a Reliable Learning Model for Cognitive Classification Tasks Using an Associative Memory", IEEE Symposium on Computational Intelligence in Image and Signal Processing, pp. 214-219, 2007.

[19] A. Ahmadi, H.J. Mattausch, M.Saeidi,M.A. Abedin, T. Koide, "An Associative Memory Based Learning Model with an Efficient Hardware Implementation in FPGA", Elsevier, in press, 2010.

[20] R.C. Gonzalez, R.E. Woods, “Digital Image Processing", Addison Wesley, 1992.

About author -- Fengwei An received the B.E. of computer science from Qingdao University of Science and Technology, China, in July 2006. Then he worked in a software development Company from August 2006. He received his M.E. from Graduate School of Information Engineering of Hiroshima University, Japan, in March 2009. Now, he is studying a PhD candidate in Graduate School of Advanced Science of Matter of Hiroshima University where he has been engaged in the hardware implementation of Machine learning for real-time applications based on the associative memory.

About author -- Hans Jürgen Mattausch received the Dipl. Phys. Degree from University of Dortmund, Dortmund, Germany, in 1977, and the Doctoral degree from the University of Stuttgart Germnay, in 1981. In 1982 he joined the Research Laboratories of Siemens AG in Munich, Germany, where he was involved in the development of MOS technology as well as the design of memory and telecommunication circuits. From 1990 he led a research group on MOS-technology based power semiconductor devices, which include device design, modeling and packaging. In 1995 he joined the Siemens Semiconductor Group as Manager of the Department for Product Analysis and Improvement in the Chip Card IC Division. Since 1996 he is

with Hiroshima University, Japan, where he is presently a Professor at the Research Institute for NanoDvices and Bio Systems. His present interests are circuit design and device model issues related to effective utilization of nanodevices and nanotechnology. Dr. Mattausch is senior member of IEEE.

About author -- Tetsushi Koide received the B.E degree in Physical Electronics, the M.E and the PhD degrees in Systems Engineering from Hiroshima University in 1990, 1992, and 1998, respectively. He was a Research Associative and Associate Professor in the Faculty of Engineering at Hiroshima University in 1992-1999. From 1999, he was with the VLSI design and Education Center (VDEC), University of Tokyo as an Associate Professor and from 2001 he was an Associate Professor in the Research Center for Nanodevices and Systems and Graduate School of Advanced Sciences of Matter, Hiroshima University respectively. His research interests include system design and architecture issues for functional-memories-based intelligent architecture and sensing systems, real-time image processing VLSI, VLSI CAD/DA, and combinational optimization. Dr. Koide is a member of the IEEE, the Association for Computing Machinery, and the Information Processing Society of Japan.