Neuromorphic Acceleration for Context Aware Text Image Recognition

Qinru Qiu, Zhe Li, Khadeer Ahmed
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY, USA
{qiqiu, zli89, khadmed}@syr.edu

Hai (Helen) Li, Miao Hu
Department of Electrical and Computer Engineering
University of Pittsburgh, Pittsburgh, PA, USA
{mih73, hal66}@pitt.edu

***Abstract*—Although existing optical character recognition (OCR) tools can achieve excellent performance in text image detection and pattern recognition, they usually require a clean input image. Most of them do not perform well when the image is partially occluded or smudged. Humans are able to tolerate much worse image quality during reading because the perception errors can be corrected by the knowledge in word and sentence level context. In this paper, we present a brain-inspired information processing framework for context-aware Intelligent Text Recognition (ITR) and its acceleration using memristor based crossbar array. The ITRS has a bottom layer of massive parallel Brain-state-in-a-box (BSB) engines that give fuzzy pattern matching results and an upper layer of statistical inference based error correction. The framework works robustly in noisy environment. A parallel architecture is presented that incorporates the memristor crossbar array to accelerate the pattern matching. Compared to traditional microprocessor, the accelerator has the potential to provide tremendous area and power savings and more than 8,000 times speedups.**

Keywords—neuromorphic*; *text recognition*; *memristor crossbar array

I. INTRODUCTION

Military planning, battlefield situation awareness, and strategic reasoning rely heavily on the knowledge of the local situation and the understanding of different cultures. A rich source of such knowledge is presented as natural-language text. In 2009, DARPA launched the Machine Reading program to develop a universal text-to-knowledge engine that scavenges digitized text to generate knowledge that can be managed by the artificial intelligence reasoning systems. The Machine Reading program limits its scope to the texts available on the World Wide Web. In real life, text exists in many forms other than its ASCII representation. These include printed texts such as books, newspapers and bulletins or hand written texts. There are many occasions when only the scanned or photographed image of the texts is available for computer processing. While the machine reading system bridges the gap between natural language and artificial intelligence, another bridge has to be constructed to link the natural state of text to its unique encoding that can be understood by computers.

Conventional Optical Character Recognition (OCR) tools or pattern recognition techniques are not enough to meet the challenges in this task. Because the text images are usually captured under extreme circumstances, sometimes the images will be noisy, or incomplete due to the damages to the printing material, or obscured by marks or stamps. Pattern recognition is extremely difficult, if not impossible, when the image is partially occluded or even missing. However, such tasks are not too difficult for humans, as the errors in image recognition will be corrected later using semantic and syntactic context. Most human cognitive procedures involve two interleaved steps, sensing and association. Together, they provide higher accuracy.

Computing models have been developed for performing cognitive functions on raw input signals such as image and audio. One representative area in this category is the associative neural network model, which is typically used for pattern recognition. We generally say that this kind of model performs the "sensing" function. In the other category, models and algorithms are researched to operate on the concept-level objects, assuming that they have already been "recognized" or extracted from raw inputs. In a recent development, the cogent confabulation model was used for sentence completion [1][2]. Trained using a large amount of literatures, the confabulation algorithm has demonstrated the capability of completing a sentence (given a few starting words) based on conditional probabilities among the words and phrases. We refer these algorithms as the "association" models.

The brain inspired signal processing flow could be applied to many applications. A proof-of-concept prototype of context-aware Intelligence Text Recognition system (ITRS) is developed on high performance computing cluster [3]. The lower layer of the ITRS performs pattern matching of the input image using a simple non-linear autoassociative neural network model called Brain-State-in-a-Box (BSB) [4]. It matches the input image with the stored alphabet. A race model is introduced that gives fuzzy results of pattern matching. Multiple matching patterns will be found for one input character image, which is referred as ambiguity. The upper layer of the ITRS performs information association using the cogent confabulation model [1]. It enhances those BSB outputs that have strong correlations in the context of word and sentence and suppresses those BSB outputs that are weakly related. In this way, it selects characters that form the most meaningful words and sentences.

Both BSB and confabulation models are connection based artificial neural networks, where *weight matrices* are used to represent synapses between neurons and their operation can be transformed into matrix-vector multiplication(s). Hardware realizations of neural networks require a large volume of memory and are associated with high cost if built with digital circuits [5].

The existence of the memristor was predicted in circuit theory about forty year ago [6] and its first physical realization was in 2008 [7]. Afterwards, many memristive materials and devices have been

This work is partially supported by the National Science Foundation under Grants CCF-1337198 and CCF-1337300, and AFRL under contract FA8750-11-1-0266.

rediscovered. Intrinsically, a memristor behaves similarly to a synapse: it can "remember" the total electric charge/flux ever to flow through it [8]. Moreover, memristor-based memories can achieve a very high integration density of 100 Gbits/cm^2, a few times higher than flash memory technologies [9]. These unique properties make it a promising device for massively parallel, large-scale neuromorphic systems. In particular, memristor crossbar, which employs a memristor at each intersection of horizontal and vertical metal wires, can naturally provide the capability of weight matrix storage and matrix-vector multiplication.

In this paper, we present the brain inspired information processing framework and its acceleration using memristor crossbar array using intelligent text image recognition as a case study. The remainder of the paper is organized as follows. In Section II we introduce the basics of models used for sensing and association in the ITRS system. Section III describes the overall system model and the algorithms in different layers. Section IV gives the details of hardware acceleration using memristor crossbar array. The experimental results and discussions are presented in Section V. Section VI summarizes the work.

II. Background

A. Neural Network and BSB Model

The BSB model is an auto-associative, nonlinear, energy minimizing neural network. A common application of the BSB model is to recognize a pattern from a given noisy version. It can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries. It has two main operations: *training* and *recall*. The mathematical model of BSB recall function can be represented as:

$$\mathbf{x}(t+1) = S\left(\alpha \cdot \mathbf{A}\mathbf{x}(t) + \beta \cdot \mathbf{x}(t)\right), \tag{1}$$

where $\mathbf{x}$ is an N dimensional real vector and $\mathbf{A}$ is an N-by-N connection matrix, which is trained using the extended Delta rule. $\mathbf{A}\mathbf{x}(t)$ is a matrix-vector multiplication, which is the main function of the recall operation. α is a scalar constant feedback factor. β is an inhibition decay constant. $S()$ is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1, & y \geq 1 \\ y, & -1 < y < 1 \\ -1, & y \leq -1 \end{cases}. \tag{2}$$

For a given input pattern $\mathbf{x}(0)$, the recall function computes (1) iteratively until *convergence*, that is, when all entries of $\mathbf{x}(t+1)$ are either '1' or '−1'.

Algorithm 1. BSB training algorithm using Delta rule.

Step 0. Initialize weights (zero or small random values).
Initialize learning rate α.

Step 1. Randomly select one prototype pattern $\gamma^{(k)} \in B^n$, $k=1,\ldots,m$. B^n is the n-dimension binary space (−1, 1).
Set target output to the external input prototype pattern $\gamma^{(k)}$: $t_i=\gamma_i$.

Step 2. Compute net inputs: $y_{in_i} = \sum_j \gamma_j w_{ji}$
(Each net input is a combination of weighted signals received from all units.)

Step 3. Each unit determines its activation (output signal):

$$y_i = S(y_{in_i}) = \begin{cases} 1, & y_{in_i} \geq 1 \\ y_{in_i}, & -1 < y_{in_i} < 1 \\ -1, & y_{in_i} \leq -1 \end{cases}$$

Step 4. Update weights: $\Delta w_{ij} = \alpha \cdot (t_j - y_j) \cdot \gamma_i$.

Step 5. Repeat *Steps 1-4* until the condition $|t(i) - y(i)| < \theta$ is satisfied in m consecutive iterations.

The most fundamental BSB training algorithm is given in Algorithm 1, which bases on the extended Delta rule [10]. It aims at finding the weights so as to minimize the square of the error between a target output pattern and the input prototype pattern.

B. Cogent Confabulation

Cogent confabulation [1] is an emerging computation model that mimics Hebbian learning, the information storage and interrelation of symbolic concepts, and the recall operations of the brain. Based on the theory, the cognitive information process consists of two steps: learning and recall. During learning, the knowledge links are established and strengthened as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-takes-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated and the winner neurons will activate other neurons through knowledge links. At the same time, those neurons that did not win in this procedure will be suppressed.

The confabulation model represents the observation using a set of features. These features construct the basic dimensions that describe the world of applications. Different observed attributes of a feature are referred as *symbols*. The set of symbols used to describe the same feature forms a *lexicon* and the symbols in a lexicon are exclusive to each other. *Knowledge links* (*KL*) are established among lexicons. They are directed edges from the source lexicons to target lexicons. Each knowledge link is associated with a matrix. The *ij*th entry of the matrix gives the conditional probability $\log\left[p\left(s_i|t_j\right)\right]$ between the symbols s_i in the source lexicon and t_j in the target lexicon. The knowledge matrix is constructed during training by extracting and associating features from the inputs.

The cogent confabulation model has close resemblance to a neural system. The symbols are analogous to neurons and knowledge links between symbols are analogous to synapses between neurons. Whenever an attribute is observed, the corresponding symbol (i.e. neuron) is activated, and an excitation is passed to other symbols (i.e. neurons) through knowledge links (i.e. synapses).

The excitation of a symbol t in lexicon l is calculated by summing up all incoming knowledge links:

$$I(t) = \sum_{s_k \in F_l} \sum_{s \in S_k} I(s) \left[\ln\left(\frac{P(s|t)}{p_0}\right) + B\right], \tag{3}$$

where F_l is the set of lexicons that has knowledge links go into lexicon l, the function $I(s)$ is the excitation level of the source symbol s. The parameter p_0 is the smallest meaningful value of $P(s_i \mid t_j)$. The parameter B is a positive global constant called the *bandgap*. The purpose of introducing B in the function is to ensure that a symbol receiving N active knowledge links will always have a higher excitation level than a symbol receiving (N-1) active knowledge links, regardless of their strength. As we can see, the excitation level of a symbol is actually its log-likelihood given the observed attributes in other lexicons.

III. System Architecture

A. Overview of the ITRS

The ITRS is divided into three layers as shown in Figure 1. The input of the system is a text image. The first layer is character recognition based on BSB models. It recalls the stored patterns of the English alphabet that matches the input image. If there is noise in the image, multiple matching patterns may be found. The ambiguity can

be removed by considering the word level and sentence level context, which is achieved by the statistical information association in the second and third layer where word and sentence is formed using cogent confabulation models.

In this work, we designed a new "racing" algorithm for BSB recalls. The algorithm is based on the observations that the convergence speed of the BSB recall process indicates the distance between the input and remembered patterns. For a given character image, we consider all patterns that converge within a certain number of iterations as potential candidates that may match the input image. Candidate BSB outputs will be activated and used to trigger the corresponding symbols in the confabulation model for information association. By using the racing algorithm, if there is noise in the image or the character is partially damaged, multiple matching patterns will be triggered for the same input image. For example, a horizontal scratch will make the letter "T" look like the letter "F". In this case we have ambiguity in character recognition. The pattern that cannot form meaningful words and sentences will be eliminated in the later stages.

Figure 1 shows an example of using the ITRS to read texts that have been occluded. The BSB algorithm recognizes text images with its best effort. The word level confabulation provides all possible words that can be formed based on the recognized characters while the sentence level confabulation finds the combination among those words that gives the most meaningful sentence.

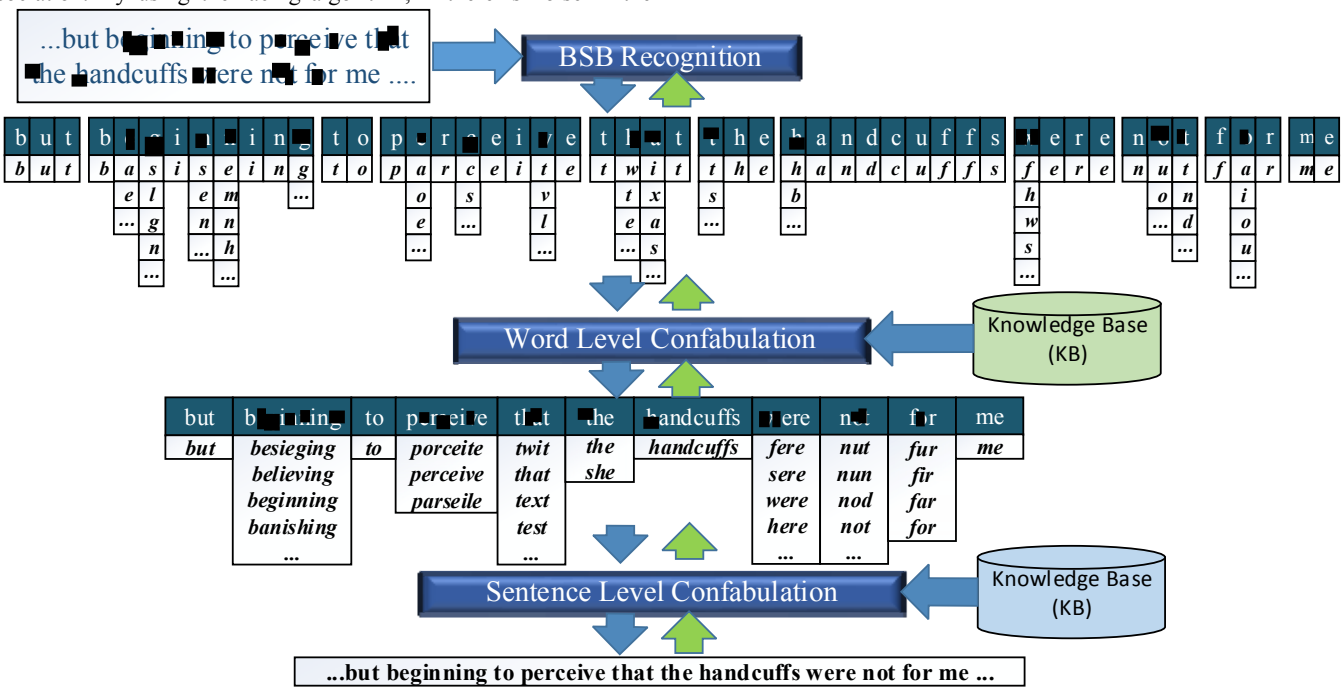

Figure 1. Overall architecture of the models and algorithmic flow.

B. *Character Level Image Recognition*

In this section we first describe the "racing" mechanism that we use to implement the multi-answer character recognition process.

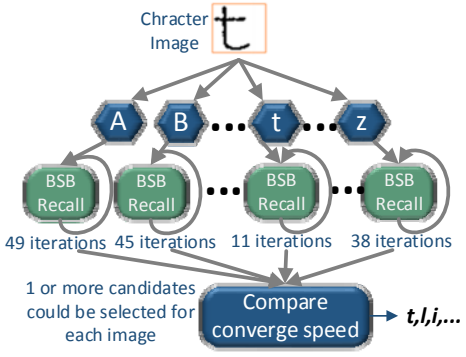

Figure 2 An input image is recalled by BSB models remembering different characters or symbols. Candidates are selected based on speed of convergence.

Let S denote the set of characters that we want to recognize. Without loss of generality, assume the size of S is 52, which is the number of upper and lower case characters in the English alphabet. We also assume that for each character, there are M typical variations in terms of different fonts, styles and sizes. In terms of pattern recognition, there is a total of $52*M$ patterns to remember during training and to recognize during recall.

One 256-dimensional BSB model is trained for each character in S. Therefore there will be a set of 52 BSB models. Each BSB model is trained for all variations of a character. The multi-answer implementation utilizes the BSB model's convergence speed to represent the similarity between an input image and the stored pattern. An input image is compared against each one of the 52 BSB models; therefore it triggers 52 recall processes. The number of iterations that each recall process takes to converge is recorded. Then we pick up to K "closest" candidates to work with high-level language models to determine the final output. Figure 2 gives an example of how the racing mechanism works.

C. *Word and Sentence Confabulation*

The inputs of word confabulation are characters with ambiguities referred as *candidates*. For each input image, one or multiple character level candidates will be generated by the BSB model. Candidates correspond to the same input image are exclusive to each other, therefore, they belong to the same lexicon and hence suppress each other. Higher order lexicons are also formed for pair of candidates corresponding to neighboring images.

Confabulation-based word and sentence recall heavily relies on the quality of the *knowledge base* (*KB*). The training of the KB is the procedure to construct the probability matrix between source symbols and target symbols. During recall, a lexicon that has multiple symbols activated is referred as ambiguous lexicon and the goal of confabulation is to eliminate the ambiguity as much as possible or to transform it into word level ambiguity, which can be further eliminated by sentence level confabulation. For each lexicon that has multiple symbols activated, we calculate the *excitation level* of each activated symbol. The *N* highest excited symbols in this lexicon are kept active. These symbols will further excite the symbols in other ambiguous lexicons. Each iteration one less symbol will be activated. The procedure will continue until there is only one active symbol in each lexicon.

The sentence level confabulation model is very similar to its word level counterpart except that there are three levels of lexicons. The first and second level LUs represent single words and adjacent word pairs; while the third level of LUs represent the parts-of-speech tags of the corresponding word.

IV. Hardware Acceleration of BSB Recall

A. Memristor and Crossbar Array

In 2008, HP Lab demonstrated the first memristive device, in which the memristive effect was achieved by moving the doping front within a TiO_2 thin-film [7]. The overall memristance can be expressed as:

$$M(p) = p \cdot R_H + (1-p) \cdot R_L , \tag{4}$$

where p ($0 \le p \le 1$) is the relative doping front position, which is the ratio of doping front position over the total thickness of the TiO_2 thin-film, R_L and R_H respectively denote the *low resistance state* (LRS) and the *high resistance state* (HRS) of the memristor. The velocity of doping front movement $v(t)$, driven by the voltage applied across the memristor $V(t)$, can be expressed as:

$$v(t) = \frac{dp(t)}{dt} = \mu_v \cdot \frac{R_L}{h^2} \cdot \frac{V(t)}{M(p)}, \tag{5}$$

where μ_v is the equivalent mobility of dopants, h is the total thickness of the thin film, and $M(p)$ is the total memristance when the relative doping front position is p. In general, a certain energy (or threshold voltage) is required to enable the state change in a memristive device. When the electrical excitation through a memristor is greater than the threshold voltage, i.e., $V(t) > V_{th}$, the memristance changes (in training). Otherwise, a memristor behaves like a resistor.

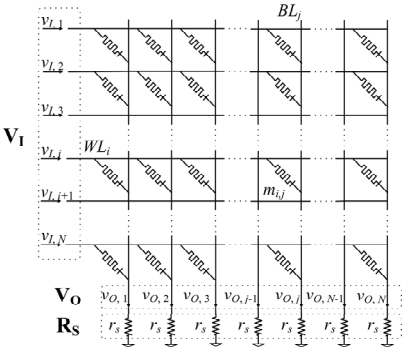

Figure 3 A memristor crossbar array

Crossbar array illustrated in Figure 3 is a typical structure of memristor based memories. It employs a memristor device at each intersection of horizontal and vertical metal wires without any selectors [11]. The memristor crossbar array is naturally attractive for implementation of connection matrix in neural networks for it can provide a large number of signal connections within a small footprint and conduct the weighted combination of input signals [12][13].

B. Matrix Multiplication using Memristor Crossbar

In order to use the *N*-by-*N* memristor crossbar array illustrated in Figure 3 for matrix computation, a set of input voltages $\mathbf{V_I}^T=\{V_{I,1}, V_{I,2}, \ldots, V_{I,N}\}$ is applied on the *word-lines* (WL) of the array, and the current through each *bit-line* (BL) is collected by measuring the voltage across a sensing resistor. The same sensing resistors are used on all BLs with resistance r_s, or conductance $g_s = 1/r_s$. The output voltage vector $\mathbf{V_O}^T=\{V_{O,1}, V_{O,2}, \ldots, V_{O,N}\}$. Assume the memristor sitting on the connection between WL_i and BL_j has a memristance of $m_{i,j}$. The corresponding conductance $g_{i,j}=1/m_{i,j}$. Then, the relation between the input and output voltages can be represented by:

$$\mathbf{V_O} = \mathbf{C}\mathbf{V_I} . \tag{6}$$

Here, $\mathbf{C}$ can be represented by the memristors' conductance and the load resistors as:

$$\boldsymbol{C} = \boldsymbol{D}\boldsymbol{G}^T = diag(d_1, \cdots, d_N) \begin{bmatrix} g_{11} & \cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,N} \end{bmatrix}, \tag{7}$$

where $d_i = 1/(g_s + \sum_{i=1}^{N} g_{i,j})$.

Please note that some non-iterative neuromorphic hardware uses the output currents $\mathbf{I_O}$ as output signals. Since the BSB algorithm discussed in this work is an iterative network, we take $\mathbf{V_O}$ as output signals, which can be directly fed back to inputs for the next iteration without extra design cost.

Equation (6) indicates that a trained memristor crossbar array can be used to construct the weight matrix $\mathbf{C}$, and transfer the input vector $\mathbf{V_I}$ to the output vector $\mathbf{V_O}$. However, $\mathbf{C}$ is not a direct one-to-one mapping of conductance matrix $\mathbf{G}$ as indicated in (7). Though we can use a numerical iteration method to obtain the exact mathematical solution of $\mathbf{G}$, it is too complex and hence impractical when frequent updates are needed.

For simplification, assume $g_{i,j} \in \mathbf{G}$ satisfies $g_{\min} \le g_{i,j} \le g_{\max}$, where $g_{\min}$ and $g_{\max}$ respectively represent the minimum and the maximum conductance of all the memristors in the crossbar array. Thus, a simpler and faster approximation solution to the mapping problem is defined as:

$$g_{j,i} = c_{i,j} \cdot (g_{\max} - g_{\min}) + g_{\min} . \tag{8}$$

A decayed version of the weight matrix $\hat{\mathbf{C}}$ can be approximately mapped to the conductance matrix $\mathbf{G}$ of the memristive array. Plugging (8) into (7), we have:

$$\hat{c}_{i,j} = \frac{c_{i,j} \cdot (g_{\max} - g_{\min}) + g_{\min}}{g_s + (g_{\max} - g_{\min}) \cdot \sum_{j=1}^{N} c_{i,j} + N \cdot g_{\min}} . \tag{9}$$

Note that many memristive materials, such as TiO_2, demonstrate a large $g_{\max}/g_{\min}$ ratio [7]. Thus, a memristor at the high resistance state under a low voltage excitation can be regarded as an insulator, that is, $g_{\min} \approx 0$. Moreover, the BSB recall matrix $\mathbf{A}$ is a special matrix with a small $\sum_{j=1}^{N} c_{i,j}$. For example, all BSB models used for character recognition in our experiments show $\sum_{j=1}^{N} c_{i,j} < 5$ when N = 256. The term $\sum_{j=1}^{N} c_{i,j}$ can be further reduced by increasing the ratio $g_s/g_{\max}$. As a result, the impact of $\sum_{j=1}^{N} c_{i,j}$

can be ignored. These two facts indicate that (9) can be further simplified as:

$$\hat{c}_{i,j} = c_{i,j} \cdot g_{\max} / g_s . \quad (10)$$

In summary, with the proposed fast approximation function (8), the memristor crossbar array performs as a decayed matrix $\hat{\mathbf{C}}$ between the input and output voltage signals.

C. Transformation of BSB Recall Matrix

A memristor is a physical device with conductance $g>0$. Therefore, all elements in matrix **C** must be positive as shown in (7). However, in the original BSB recall model, $a_{i,j} \in \mathbf{A}$ can be either positive or negative. An alternative solution is moving the whole **A** into the positive domain. Since the output $\mathbf{x}(t+1)$ will be used as input signal in the next iteration, a biasing scheme at $\mathbf{x}(t+1)$ is needed to cancel out the shift induced by the modified **A**. The biasing scheme involves a vector operation since the shift is determined by $\mathbf{x}(t)$.

To better maintaining the meaning of the matrix **A** in physical mapping and leverage the high integration density of memristor crossbar, we propose to split the positive and negative elements of **A** into two matrixes $\mathbf{A}^+$ and $\mathbf{A}^-$ as:

$$a^+_{i,j} = \begin{cases} a_{i,j}, & if\ a_{i,j} > 0 \\ 0, & if\ a_{i,j} \le 0 \end{cases} \text{ and } a^-_{i,j} = \begin{cases} 0, & if\ a_{i,j} > 0 \\ -a_{i,j}, & if\ a_{i,j} \le 0 \end{cases}. \quad (11)$$

As such, (2) becomes

$$\mathbf{x}(t+1) = S\left(\mathbf{A}^+\mathbf{x}(t) - \mathbf{A}^-\mathbf{x}(t) + \mathbf{x}(t)\right), \quad (12)$$

where we set $\alpha=\beta=1$. Thus, $\mathbf{A}^+$ and $\mathbf{A}^-$ can be mapped to two memristor crossbar arrays $\mathbf{M}_1$ and $\mathbf{M}_2$ in a decayed version $\hat{\mathbf{A}}^+$ and $\hat{\mathbf{A}}^-$, respectively, by following (8).

D. Training Memristor Crossbars in BSB Model

A software generated weight matrix can be mapped to the memristor crossbar arrays based on the assumption that every memristor in the crossbar could be perfectly programmed to the required resistance value. However, the traditional crossbar programming method faces accuracy and efficiency limitations due to the existence of the sneak paths [11]. Although some recent works were presented to improve the write/read ability of memristor crossbars by leveraging the device nonlinearity [11], the controllability of analog state programming is still limited. In spite of preparing the memristor crossbars with open-loop writing operations, we propose a *close-loop training method* which iteratively tunes the entire memristor crossbar to the target state. This technique is based on a modification of the software training algorithm.

Let's use the Delta rule in Algorithm 1 as an example. A weight w_{ij} corresponds to the analog state of the memristor at the cross-point of the ith row and the jth column in a crossbar array. A weight updating Δw_{ij} involves multiplying three analog variables: α, $t_j - y_j$, and x_i. Though these variables are available in training scheme design, the hardware implementation to obtain their multiplication demands unaffordable high computation resources. Thus, we simplify the weight updating function by trading off the convergence speed as:

$$\Delta w_{ij} = \alpha \cdot sign(t_j - y_j) \cdot sign(x_i). \quad (13)$$

Here, $sign(t_j - y_j)$ and $sign(x_i)$ are the polarities of $t_j - y_j$ and x_i, respectively. $sign(t_j - y_j) \cdot sign(x_i)$ represents the *direction* of the weight change.

The simplification minimizes the circuit design complexity meanwhile ensuring the weight change in the same direction as that of the Delta rule.

V. Experimental Results

We test the ITRS using text images with different levels of noises. Our test case is extracted from the book "Great Expectations" by Charles Dickens. The text consists of 96767 letters or 23912 words. The text has not been read during the training process. In order to explicitly control the noise in the input, we use generated bit maps of text images instead of scanned text images. Horizontal scratches are added to the images of letters selected randomly. The amount of noise in the input is controlled by two parameters: (1) the thickness of horizontal scratches varies from one pixel wide to three pixels wide. Figure 4 shows examples of the three different types of horizontal scratches. Note that the scratches are located in the center of the text image, where most of the information to distinguish amongst various characters is found. Also note that each text image is 15x15 pixels, and a 1~3 pixel scratch across the image is equivalent to 7~20% missing information. (2) The probability that a character is scratched varies from 0.2, 0.4 to 0.6.

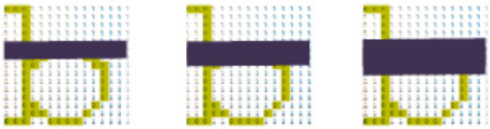

Figure 4 Three different horizontal scratches

Table 1 Sentence and word level recall accuracy

Scratch prob.	Sentence recall accuracy			Word recall accuracy			Overall % correct words		
	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch	1 scratch	2 scratch	3 scratch
0.2	0.92	0.90	0.86	0.98	0.98	0.97	0.99	0.99	0.98
0.4	0.87	0.82	0.76	0.98	0.97	0.95	0.98	0.98	0.96
0.6	0.82	0.74	0. 65	0.97	0.95	0.93	0.98	0.97	0.94

The outputs of ITRS are compared against the original text. A sentence (or a word) is considered inaccurately recognized if any word (or any letter) mismatches the original text. Table 1 gives the accuracy of word and sentence confabulation in columns 2 and 3. They are calculated as the number of sentences (or words) that have been correctly recalled divided by the number of sentence (word) confabulations that have been invoked. Please note that word/sentence confabulation is only invoked if there is ambiguity in the input image. The same table also gives the percentage of correct words in the fourth column. It is calculated as the total number of correct words (including both confabulated and none-confabulated) divided by the total number of words in the text. As we can see, the accuracy of word confabulation and the overall percentage of correct words are very close to each other. This is because the majority of words have at least one scratch. Therefore, they all need to go through the word confabulation process.

In order to reduce hand shaking time and to improve communication efficiency, every time a set of 96 character images is sent to a BSB engine and compared with 93 stored patterns. Each comparison involves 50 iterations of calculation of Equation (1). Hence, $93 \times 96 \times 50 = 446{,}400$ BSB iteration is considered as one unit workload in the BSB layer. In order to represent a 15x15 pixel character image, the input vector $\boldsymbol{x}(t)$ of the BSB is set to 256x1 dimension and the weight matrix $\boldsymbol{A}$ is 256x256. Overall,

each unit of BSB workload consists of 58 billion floating point operations.

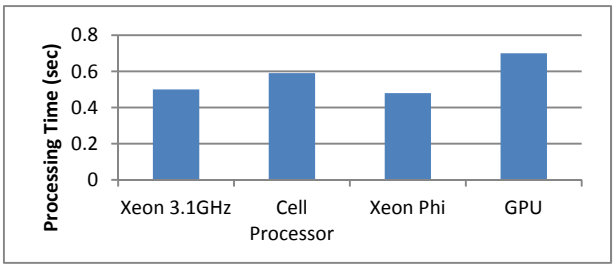

Figure 5 Processing time of a unit workload of BSB layer

We have implemented the BSB layer on different high performance computer platforms and compared their performance. The platforms include: (1) Intel Xeon Sandy Bridge-EP processor with dual CPU 8 cores 32 SMT running at 3.1 GHz, (2) IBM cell processor on Sony PlayStation 3, which has one PowerPC core and 6 synergistic processing elements (SPE) (3) NVIDIA C2050 GPU with 448 CUDA cores and 1288 GFLOP single precision peak performance, and (4) Intel 7110P Xeon Phi processor with 61 X86 cores running at 1.1 GHz frequency. Figure 5 compares the processing time of one unit workload of BSB on those high performance computing platforms.

Table 2 Performance, power and area of 256x256 memristor crossbar array

Implementations	Processing time	Area (mm^2)	Power consumption
Memristor crossbar	60μs	151	875mW
Xeon processor	0.5s	435	183W

We created a Verilog-A memristor model by adopting the device parameters from [14] and scaling them to 65nm node based on the resistance and device area relation given in [15]. Simulation is carried out to measure the performance and power consumption of a neuromorphic computing accelerator (NCA) with a 64x64 memristor crossbar array. We scaled the results to obtain an estimation of the NCA with size 256x256. The NCA includes a set of op-amp that used as analog buffer to hold the data, a 256x256 memristor based crossbar array, and a set of DAC and ADC circuitry as the interface between the NCA and its digital environment. For a lower area cost, we assume that one AD/DA converter is shared among every 4 inputs/outputs. Please note that the racing model of BSB actually does not require any AD/DA conversion, because we are only interested in when the BSB converges, i.e. when all of the V_O's are squashed to +1 or –1, hence a simple comparator is sufficient. We further assume that a set of 93 NCAs are used and each of them implement one BSB model. Table 2 gives the area, power consumption and performance estimation of the accelerator. The processing time is estimated as the time needed to complete one unit workload of BSB computation, which is to check a set of 96 images. In the same table, we also list the power consumption, area and performance of Intel Xeon Sandy Bridge-EP processor as a reference. As we can see, the memristor based neuromorphic computing accelerator provides tremendous reduction from every perspective. We need to further point out that, even though we have assumed shared AD/DA converters, they still count for more than 95% of total power consumption and more than 80% of total area. As we mentioned, the AD/DA can be replaced by simple comparators, so the area and power consumption of the NCAs can be further reduced in the future.

VI. Conclusions

This paper presents our work in neuromorphic computing and acceleration. A brain-inspired information processing framework is developed that performs document image recognition using pattern matching and statistical information association. The framework has outstanding noise resistance and is capable of recognizing words and sentences from highly damaged images at high accuracy. The detailed structure of a memristor crossbar array based neuromoprhic accelerator is described. When applied to implement the pattern matching layer of the text recognition system, the NCA provides more than 8,000X speedups over the Intel Xeon processor. The area and power consumption of the NCA is only 1/3 and 0.5% of a Xeon processor respectively.

References

[1] R. Hecht-Nielsen, "*Confabulation Theory: The Mechanism of Thought*", *Springer*, Aug. 2007.

[2] Q. Qiu, Q. Wu, D. Burns, M. Moore, M. Bishop, R. Pino, R. Linderman, "Confabulation Based Sentence Completion for Machine Reading," *Proc. Of IEEE Symposium Series on Computational Intelligence,* April, 2011.

[3] Qinru Qiu, Q. Wu, M. Bishop, R. Pino, and R. W. Linderman, "A Parallel Neuromorphic Text Recognition System and Its Implementation on a Heterogeneous High Performance Computing Cluster," *IEEE Transactions on Computers*, Vol 62, No. 5, 2013.

[4] J. A. Anderson, "An Introduction to Neural Networks," *The MIT Press,* 1995.

[5] J. Partzsch and R. Schuffny, "Analyzing the scaling of connectivity in neuromorphic hardware and in models of neural networks," *IEEE Transactions on Neural Networks*, vol. 22, no. 6, pp. 919–935, 2011.

[6] L. Chua, "Memristor-the missing circuit element," *IEEE Transaction on Circuit Theory*, vol. 18, 1971, pp. 507–519.

[7] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, pp. 80–83, 2008.

[8] L. Chua, "Resistance switching memories are memristors," *Applied Physics A: Materials Science& Processing,* vol. 102, no. 4, pp. 765–783, 2011.

[9] Y. Ho, G.M. Huang, and P. Li, "Nonvolatile memristor memory: device characteristics and design implications," in *International Conference on Computer-Aided Design* (ICCAD), 2009, pp.485–490.

[10] J. Anderson, J. Silverstein, S. Ritz, and R. Jones, "Distinctive features, categorical perception, and probability learning: some applications of a neural model." *Psychological Review*, vol. 84, no. 5, pp. 413, 1977.

[11] A. Heittmann and T. G. Noll, "Limits of writing multivalued resistances in passive nano-electronic crossbars used in neuromorphic circuits," *ACM Great Lakes Symposium on VLSI* (GLSVLSI), 2012, pp. 227–232.

[12] U. Ramacher and C. V. D. Malsburg, *On the Construction of Artificial Brains*. Springer, 2010.

[13] T. Hasegawa, T. Ohno, K. Terabe, T. Tsuruoka, T. Nakayama, J. K. Gimzewski, and M. Aono, "Learning abilites achieved by a single solid-state atomic switch," *Advanced Materials*, vol. 22, no. 16, pp. 1831–1834, 2010.

[14] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbararray/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2011.

[15] B. J. Choi, A. B. Chen, X. Yang, and I.-W. Chen, "Purely electronic switching with high uniformity, resistance tunability, and good retention in pt-dispersed sio2 thin films for reram," *Advanced Materials*, vol. 23, no. 33, pp. 3847–3852, 2011.