A Neuromorphic Architecture for Context Aware Text Image Recognition

Qinru Qiu, Zhe Li, Khadeer Ahmed, Wei Liu, Syed F. Habib
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY, USA
{qiqiu, zli89, khadmed, wliu46, sfhabib}@syr.edu

Hai (Helen) Li, Miao Hu
Department of Electrical and Computer Engineering
University of Pittsburgh, Pittsburgh, PA, USA
{hal66, mih73}@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. Optimizations on each layer of the framework are introduced to improve system performance. A parallel architecture is presented that incorporates the memristor crossbar array to accelerate the pattern matching. Compared to traditional multicore 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. Autonomous and intelligent recognition of printed or handwritten text image is one of the key features to achieve situational awareness. Although generally effective, *Conventional Optical Character Recognition (OCR)* tools or pattern recognition techniques usually have difficulties in recognizing images that are noisy, partially occluded or even incomplete due to the damages to the printing material, or obscured by marks or stamps.

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 auto-associative 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 memristor has been discovered as a promising device for massively parallel, large-scale neuromorphic systems. A memristor can "remember" the total electric charge/flux ever to the flow through it [6], which is analogous to synapses among neurons. Moreover, memristor-based memories can achieve a very high integration density of 100 Gbits/cm^2, a few times higher than flash memory technologies [7]. Due to these properties, memristor crossbar, which employs a memristor at each intersection of horizontal and vertical metal wires, is proposed to facilitate weight matrices storage and matrix-vector multiplication.

This work was partially supported by the National Science Foundation under Grants CCF-1337198 and CCF-1337300.

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

II. Related Works

During recent years, neuromorphic computing has become an important research area. The research works range from applications to hardware implementations.

In [22] Voorhies et al. introduced a uniquely structured Bayesian learning network with combined measure across spatial and temporal scales on various visual features to detect small, rare events in far-field video streams. However, this structured Bayesian learning network does not fit applications like text recognition and completion easily. Authors of [23] proposed a sophisticated method based on spiking neuromorphic systems with event-driven contrastive divergence trained Restricted Boltzmann Machines and apply the model to recognize the image of MNIST hand-written digit. However their application limits only in the pattern matching layer, and did not go beyond that. M. Schmuker at el. [24] demonstrates a brain-like processing using spiking neuron network, which achieves classification of generic multidimensional data. No specific application, however, is discussed in this work. It only provides a proof of concept design of analog electronic microcircuits to mimic behavior of neurons for real-world computing tasks.

Many existing neuromorphic computing researches concentrate on pattern matching applications such as video detection or character recognition. Very few of them study the function of probabilistic inference in neuromorphic computing. Some works also focus on developing general hardware architecture for neuromorphic computing. For example, IBM's TrueNorth [25] is an efficient, scalable and flexible non-von Neumann architecture, which integrates 1 million programmable spiking neurons and 256 million configurable synapses. The hardware is suited to many applications such as multi-object detection and classification. Other novel architectures utilize emerging device technologies such as memristor crossbar or *phase change memory* (*PCM*). Authors of [26] attempt to implement data storage using memristor and [27] describe a memristor based neuromorphic circuit capable of learning which is tolerant of error. Suri, M. at el. [28] demonstrate a unique energy efficient methodology that uses PCM as synapse in ultra-dense large scale neuromorphic systems. In [28] the demonstration of complex visual pattern extraction from real world data using PCM synapses in a 2-layer spiking neural network is shown.

To the best of our knowledge, our proposed architecture is the first that covers both the pattern matching layer and probabilistic inference layer in neuromorphic computing. Neither have the implementation on state-of-the-art multicore processor nor the projected acceleration using memristor crossbar array been addressed in previous works.

III. 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:

$$\boldsymbol{x}(t+1) = S(\alpha \cdot \boldsymbol{A}\boldsymbol{x}(t) + \beta \cdot \boldsymbol{x}(t)) \quad (1)$$

where **x** is an N dimensional real vector and **A** is an N-by-N connection matrix, which is trained using the extended Delta rule. $\mathbf{Ax}(t)$ 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:

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

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'[14].

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):

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

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 [8]. 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

Inspired by human cognitive process, cogent confabulation [1] mimics human information processing including Hebbian learning, correlation of conceptual symbols and recall action of brain. Based on the theory, the cognitive information process consists of two steps: learning and recall. 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.

In learning process, matrices storing posterior probabilities between neurons of two features are captured and referred as the *knowledge link*s *(KL)*.A KL stores weighted directed edges from symbols in source lexicon to symbols in target lexicon. The (i, j) th entry of a KL, quantified as the conditional probability $P(s_i|t_j)$, represents the Hebbian plasticity of the synapse between *i*th symbol in source lexicon *s* and *j*th symbol in target lexicon *t*. The knowledge links are constructed during learning process by extracting and associating features from the inputs and collection of all knowledge links in the model forms its *knowledge base (KB)*.

During recall, the input is a noisy observation of the target. In this observation, certain features are observed with great ambiguity, therefore multiple symbols are assigned to the corresponding lexicons. The goal of the recall process is to resolve the ambiguity and select the set of symbols for maximum likelihood using the statistical information obtained during the learning process. This is achieved using a procedure similar to the integrate-and-fire mechanism in biological neural system. Each neuron in a target lexicon receives an excitation from neurons of other lexicons through KLs, which is the weighted sum of its incoming excitatory synapses. Among neurons in the same lexicon, those that are least excited will be suppressed and the rest will fire and become excitatory input of other neurons. Their firing strengths are normalized and proportional to their excitation levels. As neurons gradually being suppressed, eventually only the neuron that has the highest excitation remains firing in each lexicon and the ambiguity is thus resolved.

Let l denote a lexicon, F_l denote the set of lexicons that have knowledge links going into lexicon l, and S_l denote the set of symbols that belong to lexicon l. The excitation of a symbol t in lexicon l is calculated by summing up all incoming knowledge links:

$$el(t) = \sum_{k \in F_l} \left[\sum_{s \in S_k} el(s) \ln \left(\frac{P(s|t)}{p_0} \right) + B \right], t \in S_l \quad (3)$$

the function *el*(*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.

IV. System Architecture

A. Overview of the ITRS

The ITRS is divided into three layers as shown in Fig. 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 matched 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.

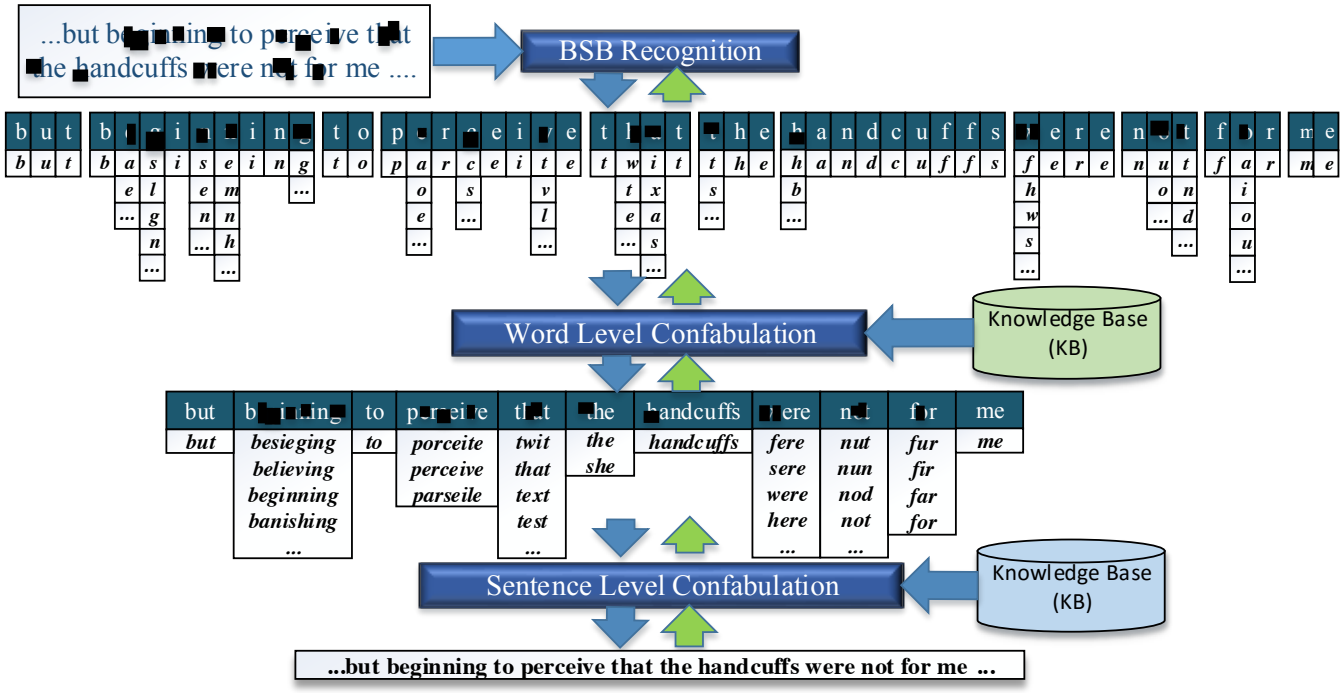

Fig. 1 Overall architecture of the models and algorithmic flow

Fig. 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.

B. Character Level Image Recognition

The initial image processing consists of six major steps performed in a sequence. These steps corrects the distortion and extract characters for further pattern recognition. To optimize performance these stages are designed as a pipeline as shown in Fig. 2.

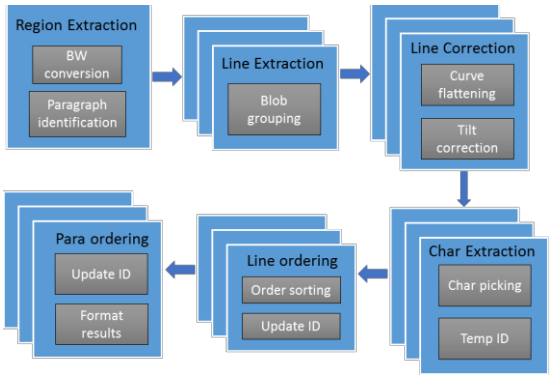

Fig. 2 Image processing pipeline

The region extraction operates at the page level. In this stage pages are broken down to paragraphs. The line extraction operates at paragraph level, which extracts the text lines from a paragraph. The line correction is the next step that corrects all deformations due to warping and rotation. Characters are then extracted and scaled in order to remove perspective distortion. Correct order of text lines in paragraph and correct order of paragraphs in a page are determined in line ordering and paragraph ordering stages. Each character image is labeled with these orders and sent to BSB model for pattern recognition.

We designed a new "racing" algorithm for BSB recalls to implement the multi-answer character recognition process. 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 $52 \times M$ patterns to remember during training and to recognize during recall.

A BSB model is trained for each character in S. Therefore there will be a set of 52 BSB models and 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 characters in K "fastest" converged processes as the final output to word confabulation model. Fig. 3 gives an example of how the racing mechanism works.

C. Word Level Confabulation

Word level confabulation interfaces between BSB and sentence confabulation, which collects ambiguous character inputs from BSB layer and generate valid combinations to form meaningful words. The word confabulation use the ambiguous letter candidates and create valid word combinations. The dictionary database is loaded as a trie data structure during initialization. An example of trie data structure is shown in the right of Fig. 4.

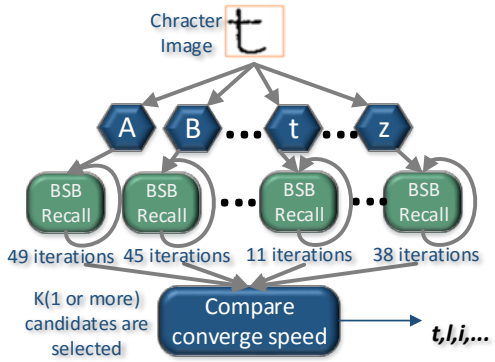

Fig. 3 Example for "racing" mechanism based on BSB model. Hand-written "t" is compared against each model storing patterns of each character in S, and initiate 52 recall processes. K fastest converged process are selected to output its corresponding character as candidates to next level, i.e. word confabulation.

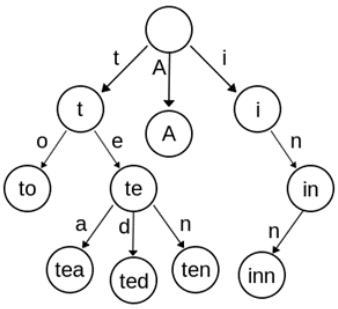

Fig. 4 Trie data structure used in Word Confabulation

The combinations based on letter candidates are validated against the trie. For example let's consider a word "dog". Its candidates for each letter position are [d o b] [o a g] [g a y]. Word confabulation will traverse through the trie using these candidates to search for the valid words presented in the trie. The valid words will be pushed onto a stack. In this example, these valid words would be: dog, day, boy, bag. Since the letter candidates were passed with their relative confidence level, the confidence level for each word will be the product of the letters it contains.

D. Sentence Level Confabulation

Sentence level confabulation model defines three levels of lexicons. The first and second level lexicons represent single words and pairs of adjacent words; while the third level of lexicons represent the *parts-of-speech* (POS) tags of the

corresponding word. During recall, those word and word-pair symbols corresponding to the outputs from word level confabulation are set as active, and all POS tag symbols are also set as active. If a lexicon has more than one active symbol, it is said to have ambiguity. The goal of sentence confabulation is to resolve the ambiguity iteratively through a recall procedure similar to belief propagation and finally form a meaningful sentence. The general confabulation recall algorithm can is described as follows in Algorithm.2.

As Algorithm 2 shows, 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. This procedure will continue until the activated symbols in all lexicons do not change anymore. If the convergence cannot be reached after a given number of iterations, then we will force the procedure to converge. Then value of N will be reduced by 1 and we repeat the above procedure. At last N is reduced to 0 which means there is only one active symbol in each lexicon. Then ambiguity is eliminated in all lexicons.

Algorithm 2. Confabulation recall algorithm

```
for each known lexicon*
   set symbol to be active
end for
for N from MAX_AMBIGUIOUS downto 1
   converged = false;
   iteration_count =0;
   while not converged
      for each unknown lexicon
         for each symbol associated to the lexicon
            calculate the excitation level of the symbol;
         end for
         select N highest excited symbols and set them to be active;
      end for
      iteration_count++;
      if activated set does not change since last iteration
         or iteration_count >= MAX_ITERATION
         converged = true;
      end if
   N--;
end for
```

*lexicons who has only one symbol candidate are denoted as known lexicons, others are unknown lexicons.

E. Improving Sentence Confabulation

In sentence confabulation, the excitation level of a candidate is the weighted sum of excitation levels of active symbols in other lexicons. Intuitively, however, different source lexicons do not contribute equally to a target lexicon. For example, the lexicon right next to an unknown word obviously gives more information in determining the unknown word than the lexicon that is five words away. Thus the significance of a KL can be measured by weight and quantified by the *mutual information*(MI)[9]. Mutual information of two random variables is a measure of variables' mutual independence, calculated as

$$I(A;B) = \sum_{b\in B}\sum_{a\in A} p(a,b) log\left(\frac{p(a,b)}{p(a)p(b)}\right) \quad (4)$$

where *A* is the source lexicon and *a* represents symbols in *A*; *B* is the target lexicon and *b* represents symbols in *B*. $p(a,b)$ is the joint probability of symbol *a* and *b*; $p(a)$ and $p(b)$ are the margin probability of symbol *a* and *b* respectively. $I(A;B)$ is nonnegative. The value of $I(A;B)$ will increase when the correlation of symbols in lexicon *A* and *B* get stronger. We defined the weight of KL (i.e. w_{kl}) from A to B as positive linear function of MI of A and B.

The sentence confabulation model in Algorithm 2 considers all initial symbols equally possible. In reality, we know that some words are more likely than others from the given image. To incorporate the image information with sentence confabulation, we consider the BSB convergence speed during the confabulation process, and modify the excitation level calculation of a word symbol *t* as follows,

$$el(t) = \alpha P_{BSB}(t) + \beta \sum_{k\in F_l}\left[w_{kl}\sum_{s\in S_k} el(s)\ln\left(\frac{P(S|t)}{p_0}\right) + B\right] \quad (5)$$

In (5), variable $P_{BSB}(t)$ is the excitation to *t* from the BSB layer, which is calculated as: $P_{BSB}(t) = \frac{1/(N_{BSB}(t)-N_{min})}{\sum_t 1/(N_{BSB}(t)-N_{min})}$, where $N_{BSB}(t)$ is the BSB convergence speed of *t*, N_{min} is the minimum convergence number that is possible for BSB engines, α and β are coefficients that adjust the weight of BSB (i.e. image) information and confabulation (i.e. language) information, $\alpha + \beta = 1$. In general, we should increase the value of α and decrease the value of β when the image has high quality and vice versa.

V. 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 TiO2 thin-film [10]. The overall memristance can be expressed as:

$$M(p) = p \cdot R_H + (1-p)\cdot R_L \quad (6)$$

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:

$$v(t) = \frac{dp(t)}{dt} = \mu_v \cdot \frac{R_L}{h^2}\cdot\frac{V(t)}{M(p)} \quad (7)$$

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.

Crossbar array illustrated in Fig. 5 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].

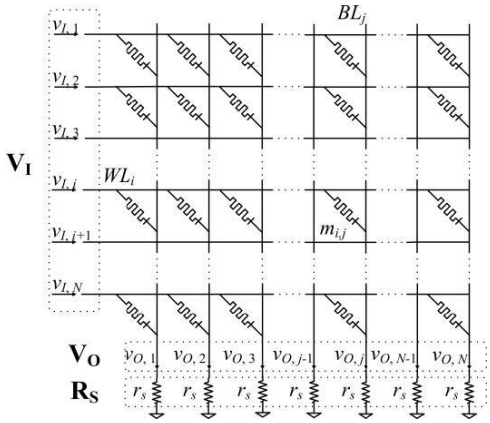

Fig. 5 A memristor crossbar array

B. Matrix Multiplication using Memristor Crossbar

In order to use the *N*-by-*N* memristor crossbar array illustrated in Fig. 5 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:

$$V_o = CV_I \tag{8}$$

Here, **C** can be represented by the memristors' conductance and the load resistors as:

$$\boldsymbol{C} = \boldsymbol{D}\boldsymbol{G}^T = diag(d_1, \cdots, d_N)\begin{bmatrix} g_{11} & \cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{N,1} & \cdots & g_{N,N} \end{bmatrix} \tag{9}$$

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 (8) indicates that a trained memristor crossbar array can be used to construct the weight matrix **C**, and transfer the input vector $\mathbf{V_I}$ to the output vector V_O. However, **C** is not a direct one-to-one mapping of conductance matrix **G** as indicated in equation (9). Though we can use a numerical iteration method to obtain the exact mathematical solution of **G**, it is too complex and hence impractical when frequent updates are needed.

For simplification, assume $g_{i,j}\in\mathbf{G}$ satisfies $g_{min}\leq g_{i,j}\leq g_{max}$, 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:

$$g_{j,i} = c_{i,j} \cdot (g_{max} - g_{min}) + g_{min} \tag{10}$$

With the proposed fast approximation function (10), the memristor crossbar array performs as a decayed matrix $\hat{C}$ between the input and output voltage signals, where $\hat{c}_{i,j} = c_{i,j} \cdot g_{max}/g_s$.

C. 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 *i*th row and the *j*th 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:

$$\Delta w_{ij} = \alpha \cdot sign(t_j - y_j) \cdot sign(x_i) \tag{11}$$

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.

D. 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 (9). 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:

$$a_{i,j}^{+} = \begin{cases} a_{i,j}, & if\ a_{i,j} > 0 \\ 0, & if\ a_{i,j} \leq 0 \end{cases} \text{ and } a_{i,j}^{-} = \begin{cases} 0, & if\ a_{i,j} > 0 \\ -a_{i,j}, & if\ a_{i,j} \leq 0 \end{cases} \tag{12}$$

As such, (1) becomes

$$\boldsymbol{x}(t+1) = S(\boldsymbol{A}^{+}\boldsymbol{x}(t) - \boldsymbol{A}^{-} \cdot \boldsymbol{x}(t) + x(t)) \tag{13}$$

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 (10).

VI. Experimental Results

In this section, we present several independent experiments carried out on different layers of the ITRS system. Each experiment is specifically designed to demonstrate our improvements on that particular layer over the previous works. Their configuration and results are discussed in detail in the following sections. We also report the accuracy and confidence level of the entire ITRS system when applied to recognize document images with different qualities. At the end, we demonstrate the recall quality of memristor crossbar array based BSB, and analyze its performance and cost.

A. Performance improvement in word confabulation layer

Instead of the hash table, which is originally used to store the dictionary, the trie data structure is applied as a new implementation to significantly reduce the search time for checking all character combinations against dictionary. Three sets of images with different qualities are used as inputs. The first set of images are clean scanned document images; the second set of images are scanned document image with 10% of characters completely occluded; and the third set of images are camera images with the same amount of occlusions. Each set consists of 8 document images. The average Signal-to-Noise Ratio (SNR) and average Peak Signal-to-Noise Ratio (PSNR) of the images in each set are given in TABLE 1. The clean image has the highest quality while the camera occluded image has the lowest quality.

TABLE 1. Quality of Input Images

Image sets	Scanned Clean	Scanned Occluded	Camera Occluded
Avg. SNR	5.1204	4.3756	3.8116
Avg. PSNR	8.095	7.3515	6.7904

TABLE 2. Improvement in word confabulation layer

	Word Confabulation Time (sec)		
	Clean image	Scan Occluded	Camera Occluded
Original implementation	310	2997	2483
New implementation	0.3	1.28	1.71

TABLE 2. compares the word confabulation time of old implementation to that of the new implementation when processing input images with different qualities. As we can see, the lower quality input image leads to higher ambiguity in pattern matching. As the number of letter candidates increases, the complexity of the original implementation of word confabulation increases exponentially as it has to check all the combinations of the letter candidates. The new implementation has much lower complexity because it pruned many invalid combinations in advance. Furthermore, the hash table based dictionary storage in the original implementation has very poor memory locality, which degrades the performance even more.

B. Performance improvement in sentence confabulation layer

As the most important layer of ITRS system, more optimizations are proposed on sentence confabulation layer. In order to focus only on the performance of sentence confabulation, for all experiments in this subsection, we set α and β in Equation (5) to 0 and 1 respectively, in order to decouple the image information from sentence confabulation. We will discuss the impact of parameters α and β in the next subsection.

Original sentence confabulation model maintains a separate knowledge link for each pair of source and target lexicons, which generates redundancy. A new implementation called circular model is proposed in [16] to merge all knowledge links between source and target lexicons that have same relative position. For example, knowledge links between any pair of adjacent lexicons will be merged as one. The new implementation not only reduces training and recall time, but also improves the accuracy of sentence completion. In this experiment, we cover random number of words completely in a sentence so that all words in dictionary are taken as candidates for the missing words. TABLE 3. shows that circular model gives 23.99% accuracy improvement, with 70.4% less effort of training and 17.5% less effort of recall.

TABLE 3. Comparasion of non-circular and circular model

	Non-circular	Circular	Improvement(%)
Training time(sec)	489180	144540	70.45%
Recall time(sec)	6317.22	5207.83	17.56%
Accuracy	54.95%	68.13%	23.99%

We also reduced the initialization time of sentence confabulation by loading the knowledge base in parallel. The size of sentence confabulation knowledge base is more than 7GB. Loading the knowledge base sequentially takes more than 83.9 seconds. A multi-thread implementation that loads the knowledge base in parallel can reduce the initialization time to 31 seconds and provides 2.7x speedups.

Integrating the POS tag in confabulation model significantly improves the sentence confabulation accuracy [15]. To evaluate the impact, the tag-assisted confabulation method is compared with no-tag confabulation at various noise levels. In this experiment, we randomly select input character image and add 3 pixel wide horizontal strikes. The noise level percentage means the ratio of characters in text with a 3-pixel wide horizontal strike. Note that the size of original character is 15x15 pixels, a 3-pixel wide strike is almost equivalent to 20% distortion.

Fig. 6 shows that no-tag sentence confabulation quickly collapse as noise level increases. This is because each test sentence contains on average 28 characters and we only consider the sentence correct if all of its characters are correct. The noise level at character level is compounded into character and word level ambiguity. Without semantic information, which provides an overall structure for each sentence, the success rate is expected to drop exponentially as noise level increase. Tag-assisted confabulation shows clear improvements over no-tag confabulation at all noise levels. The improvement

is minor at low noise level, but significant at high noise level. Overall, tag-assisted confabulation improves success rate by 33% in average.

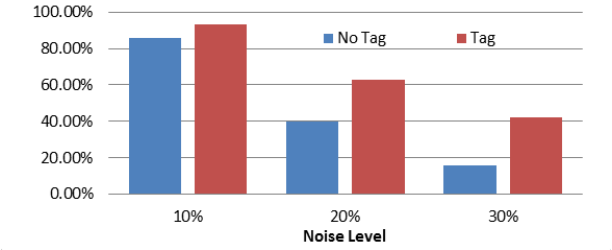

Fig. 6 Accuracy of sentence confabulation with/without POS tag

The next set of experiments is to show the impact of weighting knowledge link of sentence level confabulation using mutual information between the source and target lexicons.

This experiment is based on *Microsoft Research Sentence Completion (MRSC)* challenge. The MRSC challenge intended to stimulate research into language modeling techniques which are sensitive to overall sentence coherence [20]. The challenge consists of fill-in-the-blank questions similar to those widely used in the Scholastic Aptitude Test. We use partial training set provided by MRSC project to train our confabulation model due to the limited time. And we run recall function based on sentence confabulation model with and without weighting knowledge links to fill in the blank words for 1,040 tests in the challenge. Fig. 7 shows the recall accuracy of the two different of confabulation models. For each model, the Bandgap is varied from 1 to 1000. As we can see, when bandgap value is 10 or less, assigning weight to KL provides little improvement. However, when the bandgap value exceeds 100, assigning weight to KLs brings visible benefits; it improves accuracy by about 4%. The recall accuracy becomes saturated after the bandgap exceeds 100. We also observe that, without weighted KL, changing the bandgap value has almost no impact on the recall accuracy. Please note in this experiment, the condition is equivalent to that words are completely covered, sentence level confabulation cannot get any clue from word confabulation. And since we train incomplete training set to save time, some words appear in the tests are not stored in dictionary. An unrecognized word will never be recalled correctly by the confabulation model, thus if we train complete training set, sentence accuracy will be increased. The same testing set was evaluated in [21], our weighted confabulation model gives a slightly higher recall accuracy of 48.30% than 47% accuracy based on *recurrent neural network (RNN)* model. Please note that the RNN model identifies the missing word from the list of candidates by evaluating the probability of the sentence that they could make. Therefore, it has to create a sentence for each combination of the candidates and calculate its probability. The complexity of the RNN is an exponential function of the number of missing words, while the complexity of confabulation model is a linear function of the number of missing words.

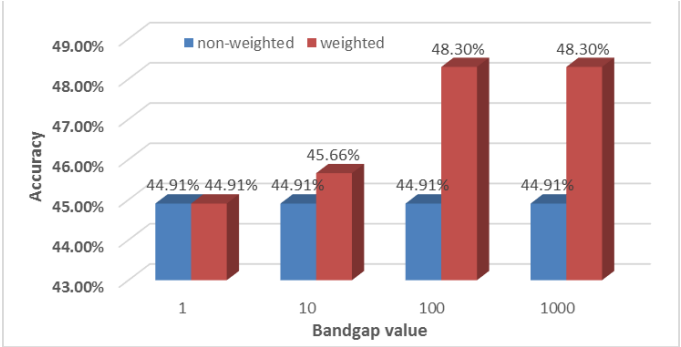

Fig. 7 Comparison of accuracy for weighted/non-weighted KL model with different bandgap value

C. *Performance improvement of overall ITR system*

To evaluate the impact of weighting image and language information. We assign w_{kl} as 1, and α varies from 1 to 0 at step of 0.1, β varies from 0 to 1 at step of 0.1. In this experiment, we run the complete ITRS to show the overall performance.

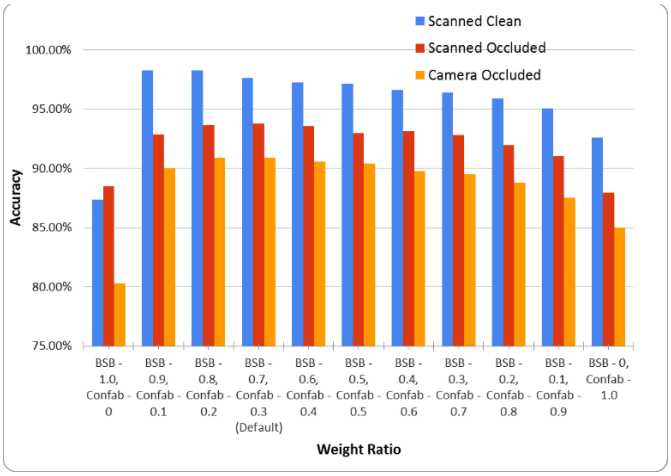

Fig. 8 Adjusting the weight of image and language information affects the accuracy of ITRS

As shown in (5), the excitation level of a word in the sentence confabulation layer is a weighted sum of two components. One of them represents the likelihood of the word based on the input image; the other represents the likelihood of word based on the language context. The parameters α and β control the weight of image information and language knowledge. Adjusting the value of α and β affects the accuracy of ITRs. Fig. 8 shows how the word accuracy changes as we vary the value of α and β. In this experiment, we take three sets of images as input, scanned clean images, scanned occluded images and occluded images taken by camera. As we can see, completely ignore either the image inform or language information will lead to poor accuracy. Furthermore, for a clean image, we can rely more on the image information, and the best quality recognition happens when α and β are set at (0.9, 0.1); while for a low quality image, we should rely more on language information, and the best quality recognition happens when α and β are set at (0.7, 0.3).

We further assign confidence level to the words recognized by ITRS. The confidence level is calculated as the normalized excitation difference between the selected candidate and its final competitor in the last round of confabulation, $c(t_1) = min[1, \frac{I(t_1)-I(t_2)}{I(t_2)}]$, where t_1 is the selected word and t_2 is its only competitor in the last round of confabulation. Under this definition, 100% confidence means that there was only 1 candidate for the lexicon while 0% confidence means that the excitation level for the two remaining candidates are the same and in that case, the program just chooses the first candidate.

TABLE 4. CONFIDENCE LEVEL OF SCANNED OCCLUDED IMAGES

File Name	Test-1	Test-2	Test-3	Test-4	Total
Total Words	761	737	745	613	2856
Total Right Words	731	703	700	575	2709
Total Wrong Words	30	34	45	38	147
Average Confidence of right words(%)	88.86	85.66	88.63	88.96	87.99
Average Confidence of wrong words(%)	16.83	20.87	15.37	24.53	19.31
Total Average Confidence(%)	86.03	82.67	84.20	84.96	84.46
Total Accuracy(%)	96.06	95.39	93.96	93.80	94.85

TABLE 4. shows the recall results for scanned occluded images as an example. Correctly recalled words have around 90% confidence compared to around 20% confidence of wrongly recalled words,. The overall average confidence is pretty high around 85%, which means the ITRS system can always eliminate the ambiguity for multiple candidates effectively and achieve a high accuracy.

In the last experiment, we compare the accuracy of ITRS with that of Tesseract on processing the same three sets of testing images. Developed initially at HP lab and now at Google, Tesseract is claimed to be the most accurate open source OCR engine available. The word accuracy of both ITRS and Tesseract are given in TABLE 5. As we can see, with the reduced image quality, the accuracy of Tesseract degrades rapidly, while the performance of ITRS is more robust. Although Tesseract produces perfect recognition with given clean image, the ITRS is more reliable under noisy environment for low quality images.

TABLE 5. COMPARISION BETWEEN ITRS AND TESSERACT

Input quality	Scanned clean images	Scanned images with occlusions	Camera images with occlusions
Tesseract	100%	93.1%	88.6%
ITRS (default)	97.6%	93.5%	90.9%
ITRS (best)	99.0%	94.8%	91.9%

Please note that, unlike Tesseract which recognize words and sentences solely based on image information, ITRS cannot guarantee the recognition of any word that is not in its dictionary. This is because the known words will always receive higher excitation than unknown words during sentence confabulation, which is analogy to human cognition process. If we exclude all proper nouns, such as the name of characters and locations, the word level accuracy of ITRS can be further increased.

D. Performance evaluation on Memristor based BSB circuit

The robustness of the BSB recall circuit was analyzed based on Monte-Carlo simulations at the component level. Memristor device parameters are taken from [10]. We tested 26 BSB circuits corresponding to the 26 lower case letters from "a" to "z". The character imaging data was taken from [17]. Each character image consists of 16×16 points and can be converted to a (−1, +1) binary vector with n=256. Accordingly, each BSB recall matrix has a dimension of 256×256. The training set of each character consists of 20 prototype patterns representing different size-font-style combinations. In each test, we created 500 design samples for each BSB circuit and ran 13,000 Monte-Carlo simulations. We use the *probability of failed recognitions (P_F)* to measure the performance of a BSB circuit.

Fig. 9 shows the comparison of P_F of each input character pattern without considering any noise sources ("Ideal") and under the scenario including all the process variations and signal fluctuations ("All noises"). In the figure, "within 3" stands for the failure rate test that the correct answer is within the top 3 recognized patterns, and "1st hit" stands for the failure rate test that the first recognized pattern is the correct answer.

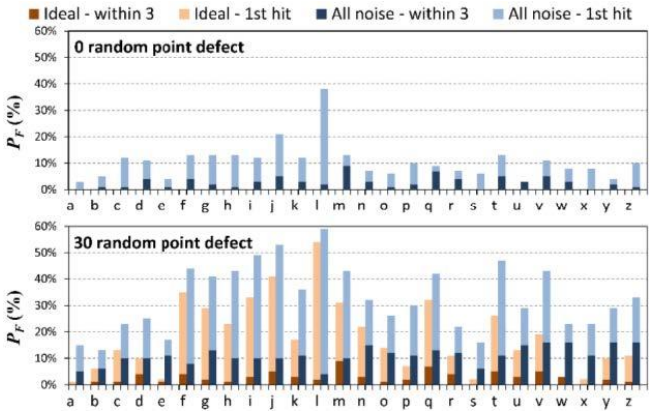

Fig. 9 P_F for each character pattern

The simulation shows that the performance degradation induced by process variations and signal fluctuations have a constant impact on all of the BSB circuits in the case of "within 3". When processing a perfect image under ideal conditions, no BSB circuits fail and hence P_F=0. After including all static and dynamic noise, P_F (within 3) ranges from 1% to 7% for different input characters. When increasing the random point defects to 30 for input images, the range of P_F (within 3) increase from 0–10% under ideal conditions to 4–16% after including the noise sources. When considering only the "1st hit" case, the P_F of most characters, both in "Ideal" or "All noise", dramatically increases as defects number goes to 30, implying that the input image defects rather than noise dominates the failure rate. Only a few characters, such as "j" and "l", are more sensitive to noise than defects as they suffer from the high failure rates even without input pattern defects.

Besides accuracy, BSB model emphasize more on speed of calculation, for ambiguity can be eliminated on word confabulation level. We created a Verilog-A memristor model by adopting the device parameters from [18] and scaling them to 65nm node based on the resistance and device area relation given in [19]. To achieve high-speed and small form factor, we adopt the flash analog-digital converter (ADC) and current steering digital analog converter (DAC) [29] in our design. For more detailed design of the peripheral circuitry of the crossbar array, please refer to [30].

We implemented the layout and schematic of the 64x64 memristor crossbar array under Cadence Virtuoso environment and extracted its area. The delay and power consumption of the crossbar array is obtained through simulation. The area, delay and power consumption of the peripheral circuits (e.g. AD/DA converter, op-amps, etc.) are estimated using published data [29]. We then scale the results to obtain an estimation of the *Neuromorphic Computing Accelerator (NCA)* with size 256x256. 93 NCAs are used and each of them implement one BSB model. TABLE 6. 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.

With the scaling of memristor devices, the programming energy will be further reduced [31] [32]. For example, Pi et al. demonstrated cross point arrays of memristive devices with a lateral dimension of 8 nm [32]. The 8 nm device arrays made required a programming current of 600 pA, and it only needed 3 nanowatts to power the operation. Moreover, memristance has an inverse proportional relationship with the device area. Thus, memristance will increase with the shrinking of device sizes, resulting in lower operation power consumption of crossbar array.

TABLE 6. COMPARISON OF MEMRISTOR AND XEON PROCESSOR

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

VII. CONCLUSIONS

This paper presents our work and optimization in neuromorphic computing with performance improvement. 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. With optimization on each layer of the framework, local and global accuracy are both increased. 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 memristor based BSB recall circuit has high resilience to process variations and signal fluctuations and NCA based on memristor crossbar array 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, "Resistance switching memories are memristors," *Applied Physics A: Materials Science& Processing*, vol. 102, no. 4, pp. 765–783, 2011.

[7] 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.

[8] 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.

[9] Z. R. Yang, M. Zwolinski, "Mutual information theory for adaptive mixture models," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, PP 396-403, April, 2001

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

[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. Ahmed, Qinru Qiu, P. Malani, M. Tamhankar, "Accelerating Pattern Matching in Neuromorphic Intelligent Text Recognition System Using Intel Xeon Phi Coprocessor," *Proc. International Joint Conference on Neural Networks* (IJCNN), 2014.

[15] F. Yang, Qinru Qiu, M. Bishop, and Q. Wu, "Tag-assisted Sentence Confabulation for Intelligent Text Recognition," *Proc. Of Computational Intelligence for Security and Defense Applications* (CISDA), May, 2012.

[16] Z. Li, Qinru Qiu, "Completion and Parsing Chinese Sentences Using Cogent Confabulation," on *Proc. Of IEEE Symposium Series on Computational Intelligence* (SSCI), 2014.

[17] Q. Wu, M. Bishop, R. Pino, R. Linderman, and Q. Qiu, "A multi-answer character recognition method and its implementation on a high-performance computing cluster," in *3rd International Conference on Future Computational Technologies and Applications*, 2011, pp. 7–13.

[18] 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.

[19] 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.

[20] Geoffrey Zweig and Chris JC Burges. A challenge set for advancing language modeling. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 29–36. Association for Computational Linguistics, 2012.

[21] B. Li, E. Zhou, B. Huang, J. Duan, Y. Wang, N. Xu, J. Zhang and H. Yang, "Large Scale Recurrent Neural Network on GPU," in *Neural Networks (IJCNN), 2014 International Joint Conference on*, pp. 4062-4069, 2014

[22] R.C. Voorhies, L. Elazary, L. Itti, "Neuromorphic Bayesian Surprise for Far-Range Event Detection," on Advanced Video and Signal-Based Surveillance (AVSS), 2012 IEEE Ninth International Conference on, pp 1-6, 18-21 Sept. 2012

[23] E. Neftci, S.Das, B.Pedroni, K. Kreutz-Delgado, G. Cauwenberghs, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in neuroscience*, vol 7, 2014

[24] Schmuker, Michael, Thomas Pfeil, and Martin Paul Nawrot. "A neuromorphic network for generic multivariate data classification." *Proceedings of the National Academy of Sciences* 111.6 (2014): 2081-2086.

[25] Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., ... & Modha, D. S. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197), 668-673.

[26] Kim, K. H., Gaba, S., Wheeler, D., Cruz-Albrecht, J. M., Hussain, T., Srinivasa, N., & Lu, W. (2011). A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications. *Nano letters*, 12(1), 389-395.

[27] Yakopcic, C., Hasan, R., & Taha, T. M. (2014, June). Tolerance to defective memristors in a neuromorphic learning circuit. In *Aerospace and Electronics Conference, NAECON 2014-IEEE National* (pp. 243-249). IEEE.

[28] Bichler, O., Suri, M., Querlioz, D., Vuillaume, D., DeSalvo, B., & Gamrat, C. (2012). Visual pattern extraction using energy-efficient "2-PCM synapse" neuromorphic architecture. Electron Devices, IEEE Transactions on, 59(8), 2206-2214.

[29] Gustavsson, M., Wikner, J. J., & Tan, N. (2000). CMOS data converters for communications. *Springer Science & Business Media*.

[30] X. Liu, M. Mao, B. Liu, H. Li, Y. Chen, B. Li, Y. Wang, "RENO: A High-efficient Reconfigurable Neuromorphic Computing Accelerator Design," *Proc. Of Design Automation Conference*, June, 2015.

[31] V. V. Zhirnov, R. Meade, R. K. Cavin, and G. Sandhu, " Scaling limits of resistive memories," *Nanotechnology*, vol. 22, no. 25, 2011.

[32] S. Pi, P. Lin, and Q. Xia, "Cross point arrays of 8 nm38 nm memristive devices fabricated with nanoimprint lithography," *Journal of Vacuum Science & Technology*, B 31, 06FA02 (2013).