Image Based HTM Word Recognizer for Language processing

Aidana Irmanova, Olga Krestinskaya, Alex Pappachen James
Nazarbayev University, Astana, Kazakhstan
apj@ieee.org

Abstract

The hardware implementation of neuro-inspired machine learning algorithms for near sensor processing on edge devices is an open problem. In this work, we propose a solution to written word recognition problem related to sequence learning tasks with images. Applying a theoretical framework of neocortex functionality as a sequence learning algorithm on a hardware implementation of Hierarchical Temporal Memory (HTM), we test the potential use of HTM in near-sensor on-chip natural language processing for text/symbol recognition.

Keywords: HTM, natural language processing, text recognition, symbol recognition.

1. Introduction

Natural language is a connected system of symbols with several levels of symbol dependencies. The language comprehension and language production abilities require the realization of these levels in a sequential order. The ability to learn these sequences is accomplished through the activation of cortical structures in a sequence determined by the evolution and an individual development [1]. HTM is a machine learning algorithm that mimics the functionality of pyramidal neurons of human brain neocortex responsible for sequence learning. The analog hardware implementation of such algorithms compatible with edge devices is an open research problem.

In this paper, we show that HTM algorithm can be used for symbol order recognition and learning sequences from character images. To test the performance of the method, we perform system-level simulations by comparing the sequences of character images. Testing this similarity of character sequences from images, that we refer as word recognition, is the step towards using the HTM algorithm for solving sequence learning tasks such as spell checking.

Current HTM hardware implementations based on CMOS-memristor hybrid circuits are proven to be useful for Face Recognition and Automatic Speech Recognition tasks [2,3]. In these architectures of HTM, pixel values are taken as the sensory input for both face and speech recognition (extracted MFCC images) tasks and deterministic learning approach is used for algorithm implementation. Following the algorithm described in [2,3], we extend this work to the sequence (word) recognition task. The description of the proposed HTM architecture and the estimation of overall power and area consumption specific to image-based word recognition task are provided in Section 3.

2. HTM for sequence learning

The HTM algorithm for sequence learning is divided into two main parts: Spatial Pooler (SP) and Temporal Memory (TM). The HTM SP is used for the identification of spatial patterns of input sensory data and encoding them to the sparse distributed representations (SDRs) via activation of neuron columns. Originally, spatial pooling is implemented with four steps: (1) initialization, (2) overlap, (3) inhibition, and (4) learning [4]. The spatial pooler (SP) design is implemented based on the learning rules and algorithm proposed in [2]. The HTM TM stores learned synaptic values over time and the patterns that are likely to follow each other are memorized. The synaptic weights of HTM TM are adjusted according to the spatial variations reflected in the training set.

The block diagram of HTM algorithm used in this work for sequence learning is illustrated in Fig. 1. The character images are applied as an input to the data controller, where the initial pre-processing is performed for forming the image sequence representing words. These words are further processed by HTM SP for feature extraction, encoding and converting the images to SDRs. The output data controller along with the HTM TM processes SDRs, to either update the weights in TM during the training stage or compare the SP outputs of words during the testing stage.

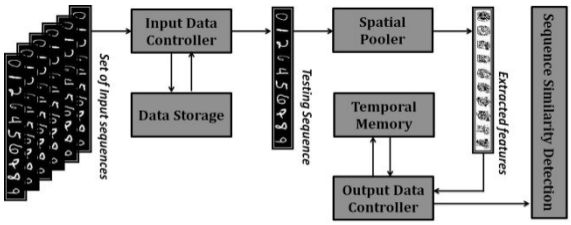


Fig. 1. System level diagram of SP and TM for sequence matching

978-1-5386-5807-9/18/$31.00 ©2018 IEEE

The sequence of digits "0123456789" is used to represent the sequence of letters, which is equivalent to a word that HTM learns to recognize. The image sequences forming the words used in this paper are created from MNIST database [5] and an example set of generated word sequences are shown in Fig. 2(a). To create the set of words for HTM training, 100 words are created using images from MNIST. HTM is trained to recognize the words considering spatial variance and differences in writing styles.

In the testing stage, the accuracy of word sequence recognition is tested on the word sequences with the different order of the characters shown in Fig. 2(b). The testing set of images consists of six categories of a word sequence belonging to the same class with each category having 60 samples representing different levels of sequence errors, leading to the total of 360-word sequences with variable errors. First 60 samples from the first category represent the ideal word sequence order without errors, while other categories of word sequences have one, two, three, four, and five character sequence errors. To introduce the error into the sequence several characters are replaced randomly with other characters, and this changes the original order of the sequence.

(a) (b)

Fig. 2. Sample images from training (a) and testing (b) sequences of digits

Algorithm 1 shows the proposed design of HTM word recognizer implemented in MATLAB. Lines 2-19 of the algorithm refer to feature extraction of the HTM SP. The algorithm generates a random weight matrix w of size N × N bits having a range from 0 to 1. Dimensions of initially generated random weights matrix w define the dimensions of each column within the HTM SP. Lines 5-8 define the initial connectivity of each synapse in HTM. Overlap value is calculated by *column.overlap()* for each column m, which is represented as the sum of the products of synaptic weight matrix w and $N \times N$ bits of the image within the region m. Lines 15-19 refer to the inhibition stage, where a windowing operation of $M \times M$ columns is performed. The inhibition output depends on the overlap values of the columns within that inhibition region *inhibition.block()*. The inhibition is performed by comparing individual overlap values of columns with the maximum overlap θ that is detected within that particular inhibition region. Finally, the columns with overlap value greater than or equal to the threshold θ are activated. Otherwise, columns are will stay inactive with logical 0 value. As a result, the binary output image SP.*image* is formed by concatenating all inhibition regions.

```
1:  Image based HTM word recognizer
2:  >HTM SP
3:  Create random matrix w of N × N size
4:   for all n in w do
5:        if w(n) > γ then
6:             w(n) = 1
7:        else
8:             w(n) = 0
9:  Divide image into blocks of N × N pixels
10:  for m image blocks do
11:       column.overlap(m) = sum(w × image.block(m))
12:  Divide image into inhibition blocks of M × M columns
13:  for c columns within inhibition.block(M × M) do
14:       θ = max(column.overlap(c.columns))
15:       if column.overlap(c) ≥ θ then
16:            inhibition.block(c) = 1
17:       else
18:            inhibition.block(c) = 0
19:  SP.image = inhibition.block
20: > HTM TM
21:  if training stage then
22:       Define number of train images z per sequence
23:       for all j in z train images do
24:            for all i pixels in SP.image do
25:                 if SP.image(j,i) = 1 then
26:                      sequence(i) = sequence(i) + Δ
27:                 else if SP.image(j,i) = 0 then
28:                      sequence(i) = sequence(i) − Δ
29:       for all i pixels in sequence do
30:            if sequence(i) ≥ σ then
31:                 sequence(i) = 1
32:            else if sequence(i) < σ then
33:                 sequence(i) = 0
34:  > Match Calculation
35:  else if testing stage then
36:  for i image pixels do
37:       score = sum(XOR(sequence(i),SP.image(i)))
38:  match = ((i − score)/i) • 100%
```

Algorithm 1. Proposed image based HTM word recognizer

The HTM TM stage is shown in lines 24-33. The HTM TM stage has a learning mechanism that is activated during the training stage when binary features extracted by HTM SP SP.image is moved by the output data controller to the HTM TM block. The HTM TM creates sequence matrix for a given sequence representation, that reflects temporal variations of its spatial features. The HTM TM is updated, and when newly extracted features are applied to the HTM TM block. The features in a class template class.map are increased by Δ value if the training image features processed by HTM SP are 1, and decreased if it equals 0. At the end of the training stage, according to lines 29-33, the sequence of each class is binarized.

Pattern matching process is active during the testing stage when binary image *SP.image* is transmitted by the output data controller block to the Sequence Matching block. The similarity score between the extracted features SP.*image* of the testing and the trained word sequence representation

978-1-5386-5807-9/18/$31.00 ©2018 IEEE

Table 1. Mean values of sequence similarities of each category to the original sequence (seq.)

	Sequence similarity					
	original sequence	seq. with 1 error	seq. with 2 errors	seq. with 3 errors	seq. with 4 errors	seq. with 5 errors
Mean	80.8720238 ± 1.35	80.52275 ± 1.36	80.39647 ± 1.46	80.05974 ± 1.19	79.81399 ± 1.27	79.36246 ± 1.11

stored within HTM TM is defined in lines 37-38 with the sum of XOR logic outputs.

3. Hardware implementation

The overall hardware implementation of the proposed HTM word recognizer consists of HTM SP, HTM TM, and Memristive Pattern Matcher. The analog hardware implementation of the HTM SP is shown in Fig.4(a). Receptor Block (RB) performs the selection of randomly generated synaptic weights represented as memristors with the random state. In RB, the input sequence of character image features represented as pixels is applied to 10 different sets of the random weights using the averaging method. The output of each set V_{R1} is fetched to the Receptor Block Mean circuit, which produces the final value of the RB V_{RB}. This corresponds to the calculation of the overlap value in each HTM column. In the next step, the calculation of threshold value for Inhibition Block (IB) is performed by the averaging memristive circuit and V_{AVG} is obtained. The inhibition of the RBs is performed by the comparator that compares the output of each RB with the obtained threshold value. The comparator output is inverted and the final binary output of a single IB is produced. With this operation, important spatial features of the image are extracted and irrelevant features are inhibited.

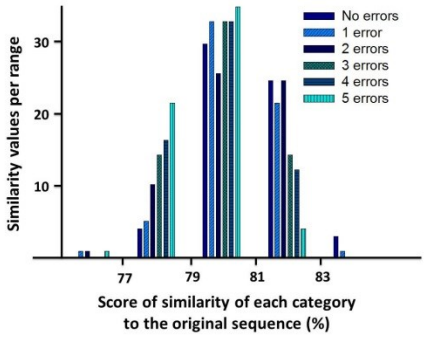
Figure 4(b) presents the analog hardware implementation of the HTM TM. As shown in the first block of the figure the output of the HTM SP is applied to the comparator that consists of two inverters. The comparator performs the comparison of the pixel as a feature of a new training image with a stored training sequence samples. The comparator decides whether to increase the pixel value of the sample training image (if the pixel of the new image is white, 1V) or to reduce it (the pixel of the new image is black, 0V). The comparator output is either $+\Delta$ or $-\Delta$ value. The summation of the pixel of the previous image and the Δ value is performed by the summing multiplier. The summing multiplier consists of the averaging circuit, that calculates the mean of the inputs, and amplifier, which multiplies the mean value by 2 to perform the summation operation. The new temporal pattern is saved in a multi-level memory that stores discrete analog values. When the training cycle is complete, the output of the HTM TM is binarized by the thresholding circuit to perform the word recognition.

In the word recognition stage, the matching of the HTM TM pattern and new HTM SP output is performed by the Memristive Pattern Matcher shown in Fig.4(c). The Memristive Pattern Matcher consists of the Memristive XNOR gate (Pattern Matcher cell) comprised of 2 averaging circuits and 3 inverters. Each pattern matcher cell performs the matching of a single pixel. Then, the similarity score is calculated based on average outputs of the Memristive Pattern Matcher cells for each image pixel.

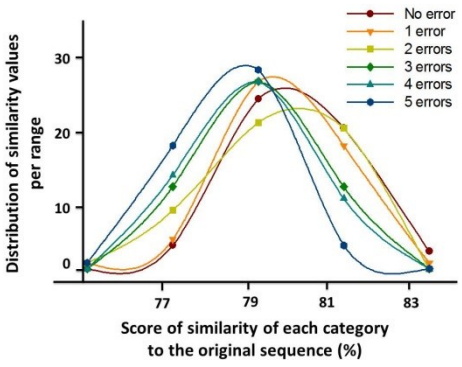
4. Results

To present the functionality of the proposed HTM system, system level simulation as in Algorithm 1 and hardware level simulation for SP and TM were conducted. Similarity of the original sequence to the sequences with errors was evaluated at the system level, and with hardware simulation tools area and power consumption of HTM were calculated.

Mean values of the sequence similarities for each category to the original sequence are provided in Table 1. The mean similarity to the original sequence gradually decreases with the increase of error in the sequence. Fig. 3(a) shows the histogram of similarity score for each category of testing images, while Fig. 3(b) shows the distribution curve of the histogram values. The curve of the category with 5 errors is skewed to the left side representing the decreased level of similarity score, while the curve of the sequence with no errors is most skewed to the right, representing a higher level of similarity scores.



(a)



(b)

Fig. 3. Histogram values of sequence similarity scores

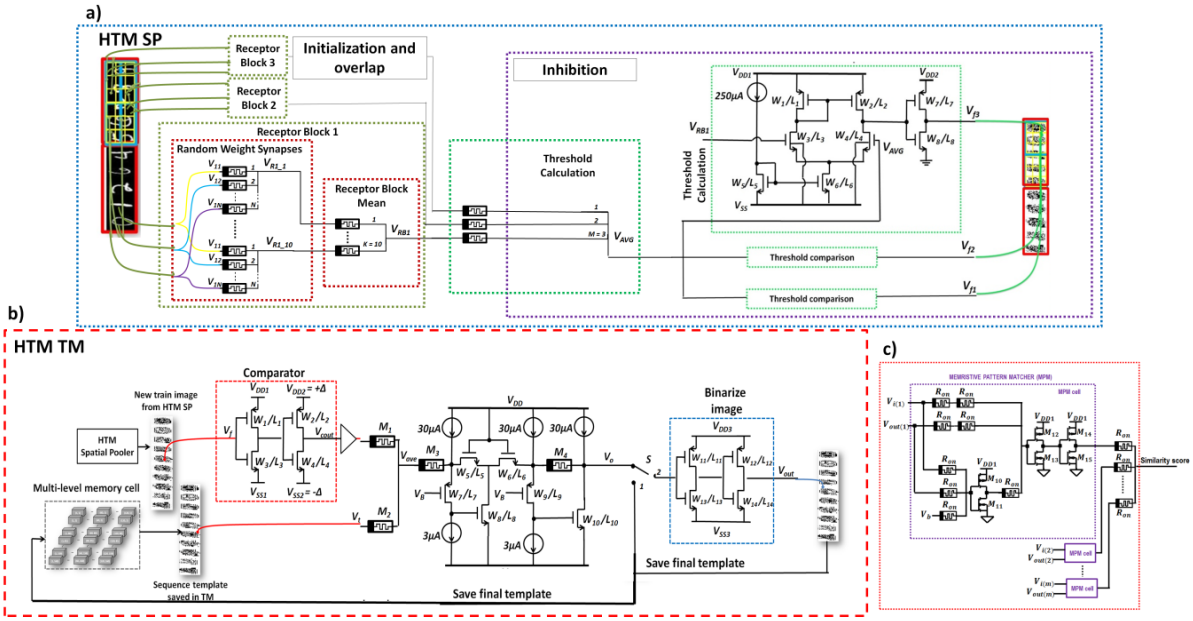


Fig. 4. HTM architecture: (a) Spatial Pooler circuit implementation that consists of Initialization, Overlap and Inhibition Blocks, (b)Temporal Memory circuits incorporating multi-level memories for discrete analog data storage of synaptic weights and synaptic weight update circuits and (c) Memristive Pattern Matcher

Table 2 shows the on-chip area and power consumption for sequential and parallel processing of HTM configuration blocks. Parallel processing involves the concurrent computation and simultaneous execution of the SP and TM operations for all input pixels. This optimizes the performance of HTM in terms of speed, but it is not efficient in terms of area and power consumption. Sequential processing in turn ensures the reduction of on-chip area and power computation with the limitation of reduced processing speeds.

Table 2. On-chip area and power consumption

Configuration	Area (µm2)	Maximum Power (mW)
Sequential processing		
HTM SP	19.96	365.88
HTM TM	23.85	442.26
Memristive PM	1.18	69.44
Parallel processing		
HTM SP	39121.6	717.12
HTM TM	46746	866.83
Memristive PM	2312.8	136.10

5. Conclusion

In this paper, we demonstrated a practical application of HTM circuits for sequence learning from handwritten character images. The system level simulation of HTM circuits illustrated the possibility to differentiate between the correct sequences of digits and its mismatch. The proposed application with HTM circuits can be used to perform intelligent back-plane information processing in CMOS image pixel arrays.

References

[1] Grossman, M., "Neurolinguistics and linguistic aphasiology: An introduction", *Annals of Neurology*, vol. 24, no. 6, pp. 800–801, 1988

[2] T. Ibrayev, U. Myrzakhan, O. Krestinskaya, A. Irmanova, and A. P. James, "On-chip face recognition system design with memristive hierarchical temporal memory," *CoRR*, vol. abs/1709.08184, 2017. [Online]. Available: *Arxiv.org/abs/1709.08184*

[3] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor logic circuits for pattern recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2017.

[4] J. Hawkins, S. Ahmad, and D. Dubinsky, "Hierarchical temporal memory including htm cortical learning algorithms," *Techical report, Numenta, Inc, Palto Alto http://www. numenta. com/htmoverview/education/HTM CorticalLearningAlgorithms. pdf*, 2010.

[5] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/