Eur. Phys. J. Special Topics **228**, 2313–2324 (2019)

https://doi.org/10.1140/epjst/e2019-900046-x

THE EUROPEAN PHYSICAL JOURNAL SPECIAL TOPICS

Review

A survey on LSTM memristive neural network architectures and applications

Kamilya Smagulova and Alex Pappachen James[a]

Nazarbayev University, 53 Kabanbay Batyr Ave., Nur-Sultan, Kazakhstan

Received 28 February 2019 / Received in final form 4 may 2019
Published online 14 October 2019

Abstract. The recurrent neural networks (RNN) found to be an effective tool for approximating dynamic systems dealing with time and order dependent data such as video, audio and others. Long short-term memory (LSTM) is a recurrent neural network with a state memory and multilayer cell structure. Hardware acceleration of LSTM using memristor circuit is an emerging topic of study. In this work, we look at history and reasons why LSTM neural network has been developed. We provide a tutorial survey on the existing LSTM methods and highlight the recent developments in memristive LSTM architectures.

1 Introduction

The volume, veracity and velocity of data from edge devices in Internet of things framework puts the need for near-edge smarter memories and information processing. One of the efficient tools for real-time contextual information is a recurrent neural network (RNN). The idea of using neural networks for data processing is not new but is increasingly becoming a reality with rapid device scaling and emerging technologies such as in-memory computing and neuromorphic chips.

Unlike feedforward neural networks, RNN has feedback connections between nodes and layers that can process input sequences of arbitrary length. However, training the simplistic RNN can be challenging task. The algorithms used for weight update in RNN are mainly gradient based and this lead to either vanishing or exploding gradient problems which is proved to overcome with the development of "Long short-term memory" (LSTM). LSTM is a special type of RNN that possesses an internal memory and multiplicative gates. Variety of LSTM cell configurations have been described since the first LSTM introduction in 1997 [8].

To the date, more than 1000 works dedicated to Long Short-term Memory Recurrent Neural Networks and their variants are published. Around 900 of them were published after 2015. The increase of interest to LSTM might have been caused by a new approach in explaining of LSTM functionality [12] and its effectiveness [9]. The LSTM contributed to the development of well-known tools as Google Translate, Siri, Cortana, Google voice assistant, Alexa. The motivation of this work is to give an overview of LSTM architectures and its applications.

[a] e-mail: apj@ieee.org

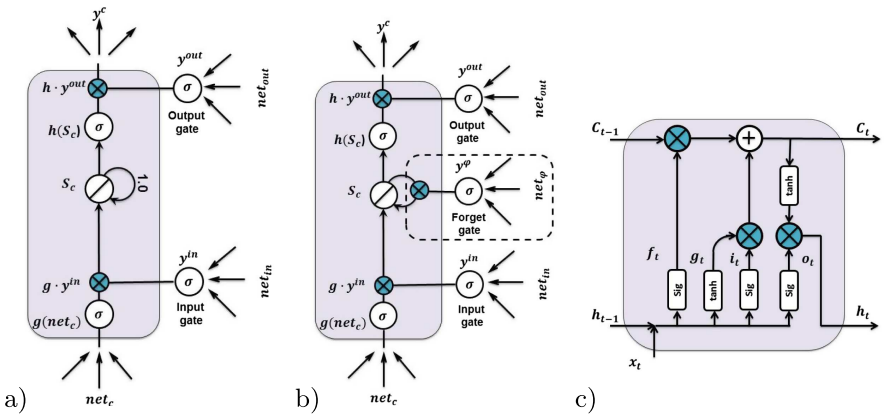


Fig. 1. (a) An original LSTM unit acrhitecture: a memory cell and two gates; (b) LSTM cell with forget gate; (c) modern representation of LSTM with forget gate.

2 LSTM cell architecture

2.1 An original LSTM: No Forget Gate (NFG)

The original paper presenting standard LSTM cell concept was published in 1997 [8]. A simple RNN cell (Fig. 1a) was extended by adding a memory block which is controlled by input and output multiplicative gates. Figure 1b demonstrates LSTM architecture of the jth cell c_j.The heart of a memory block is a self-connected linear unit s_c also called "constant error carousel" (CEC). CEC protects LSTM from vanishing and exploding gradient problems of traditional RNNs. An input gate and output gates consist of corresponding weight matrices and activation functions. The input gate with weighted input net_{in} and output y^{in} is capable of blocking irrelevant data from entering the cell. Similarly, the ouput gate with weighted input net_{out} and y^{out} shapes the output of the cell y^c.

Overall, it can be concluded that LSTM cell consist of one input layer, one output layer and one self-connected hidden layer. The hidden unit may contain "conventional" units that can be fed into subsequent LSTM cells. The mathematical description of LSTM output of the jth cell at time t is:

$$y^{c_j}(t) = y^{\text{out}_j}(t)h(s_{c_j}(t)) \tag{1}$$

where $s_{c_j(t)}$ is an internal state:

$$s_{c_j(t)} = s_{c_j(t-1)} + y^{\text{in}_j}(t)g(\text{net}_{c_j}(t)). \tag{2}$$

In equation (1) a differentiable function h scales s_c and in equation (2) function g squashes weighted input of the cell $\text{net}_{c_j}(t)$. The output values of input and output gates are:

$$y^{\text{out}_j}(t) = f_{\text{out}_j}(\text{net}_{\text{out}_j}) \tag{3}$$

$$y^{\text{in}_j}(t) = f_{\text{in}_j}(\text{net}_{\text{in}_j}) \tag{4}$$

and net inputs of a cell:

$$\mathrm{net}_{\mathrm{out}_j}(t) = \sum_u w_{\mathrm{out}_j u} y^u(t-1) \tag{5}$$

$$\mathrm{net}_{\mathrm{in}_j}(t) = \sum_u w_{\mathrm{in}_j u} y^u(t-1) \tag{6}$$

$$\mathrm{net}_{c_j}(t) = \sum_u w_{c_j u} y_u(t-1). \tag{7}$$

In equations (5)–(7), indices u represent any units, including conventional ones.

2.2 LSTM with forget gate

Nevertheless, a standard LSTM cell also met some constraints due to a linear nature of s_c. It was identified that its constant growth may cause saturation of the function h and convert into ordinary unit. Therefore an additional forget gate layer was included [4]. A new gate allows unneeded information to be erased and forgotten. The Figure 1b shows a new cell architecture. The behaviour of LSTM cell with forget gate is similar to standard LSTM with the exception of s_c as it includes an impact of y^φ:

$$s_{c_j(t)} = y^{\varphi_j}(t) s_{c_j(t-1)} + y^{\mathrm{in}_j}(t) g(\mathrm{net}_{c_j}(t)) \tag{8}$$

where y^φ is an output of forget gate:

$$y^{\varphi_j}(t) = f_{\varphi_j}(\mathrm{net}_{\varphi_j}). \tag{9}$$

Over the last twenty years, a variety of different LSTM configurations were proposed. Moreover, other notations for cell description were adopted as well [11,16]. Majority of recent papers use enumeration listed in the Table 1. In addition, in late works gates are included to the cell. LSTM architecture of frequent occurrence is demonstrated in the Figure 1c. For the sake of consistency, we will stick to in the subsequent sections.

Based on the Table 1, the feedforward behaviour of the most commonly used configuration can be described by equations (10)–(15). The cell input x_t at time t concatenates with output of a cell h_{t-1} at previous time step $t-1$. The resulting vector goes through input node and forget, input and output gates:

$$g_t = \tilde{C}_t = \tanh(W^{(g)} x_t + U^{(g)} h_{t-1} + b^{(g)}) \tag{10}$$

$$f_t = \sigma(W^{(f)} x_t + U^{(f)} h_{t-1} + b^{(f)}) \tag{11}$$

$$i_t = \sigma(W^{(i)} x_t + U^{(i)} h_{t-1} + b^{(i)}) \tag{12}$$

$$o_t = \sigma(W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)}). \tag{13}$$

Then, forget gate decides whether to keep cell data C_{t-1} from a previous time step or block it. The current cell memory state and output of the cell are defined by:

$$C_t = g_t \bigodot i_t + f_t \bigodot C_{t-1} \tag{14}$$

$$h_t = o_t \bigodot \tanh(C_t), \tag{15}$$

where a symbol $\bigodot$ denotes a pointwise or Hadamard multiplication.

Table 1. LSTM cell notations: original and modern.

Parameter (at time t)	Initial notation	Common notation
Net input of a cell	net_{c_j}	–
Net input of an input gate	net_{in_j}	–
Net input of an output gate	$\text{net}_{\text{out}_j}$	–
Net input of a forget gate	net_{φ_j}	–
Input of a cell		x
Output of a cell	y^{c_j}	h
Input gate	y^{in}	i_t
Input node	g^{in}	$g_t/\tilde{C}_t$
Output gate	y^{out}	o_t
Forget gate	y^{φ}	f_t
Cell state/internal memory	$s_c(t)$	C_t
Activation function – sigmoid (0,1)	f_{in},f_{out},f_{φ}	σ
Centered logistic sigmoid function (−2,2)	g	–
Centered sigmoid (−1,1)	h	–
Hyperbolic tangent	–	tanh
Hidden layer weight matrix	w_*	$U^{(*)}$
Input weight matrix		$W^{(*)}$

Weight increment during backpropagation can be found from the equation below:

$$W^{\text{new}} = W^{\text{old}} - \lambda \cdot \delta W^{\text{old}}, \tag{16}$$

where λ is a Stochastic Gradient Descent (SGD) coefficient and deltas $\delta W = \sum_{t=1}^{T}$, $\delta\text{gate}S_t \cdot x_t$, $\delta U = \sum_{t=1}^{T} \delta\text{gate}S_{t+1} \cdot h_t$, and $\delta b = \sum_{t=1}^{T} \delta\text{gate}S_{t+1}$.

For the sake of simplicity following notations were used:

$$\text{gate}S_t = \begin{bmatrix} g_t \\ i_t \\ f_t \\ o_t \end{bmatrix}, \; W = \begin{bmatrix} W^{(g)} \\ W^{(i)} \\ W^{(f)} \\ W^{(o)} \end{bmatrix}, \; U = \begin{bmatrix} U^{(g)} \\ U^{(i)} \\ U^{(f)} \\ U^{(o)} \end{bmatrix}, \; b = \begin{bmatrix} b^{(g)} \\ b^{(i)} \\ b^{(f)} \\ b^{(o)} \end{bmatrix}. \tag{17}$$

Deltas of $\text{gate}S_t$ are to be found using following equations [6]:

$$\delta h_t = \Delta_t + \Delta h_t \tag{18}$$

$$\delta C_t = \delta h_t \bigodot o_t \bigodot (1 - \tanh^2(C_t)) + \delta C_{t+1} \bigodot f_{t+1} \tag{19}$$

$$\delta g_t = \delta C_t \bigodot i_t \bigodot (1 - {g_t}^2) \tag{20}$$

$$\delta i_t = \delta C_t \bigodot g_t \bigodot (1 - i_t) \tag{21}$$

$$\delta f_t = \delta C_t \bigodot C_{t-1} \bigodot f_t \bigodot (1 - f_t) \tag{22}$$

$$\delta o_t = \delta h_t \bigodot \tanh(C_t) \bigodot o_t \bigodot (1 - o_t) \tag{23}$$

$$\delta x_t = W^T \cdot \delta\text{gate}S_t \tag{24}$$

$$\Delta h_{-1} = U^T \cdot \delta\text{gate}S_t \tag{25}$$

2.3 Other LSTM variants

Apart from variants in Table 2, other types of LSTM also include following configurations: No Input Gate (NIG),No Output Gate (NOG), No Input Activation Function (NIAF), No Output Activation Function (NOAF), Coupled Input and Forget Gate (CIFG).

2.4 LSTM models and applications

2.4.1 LSTM applications

LSTM neural networks can be used to implement various tasks such as prediction, pattern classification, different types of recognition, analysis and even sequence generation. Due to capability to process sequential data, LSTM is an efficient tool in many different fields including statistics, linguistics, medicine, transportation, computer science and others.

Since 1997 more than 1000 works were presented at conferences and journals (Fig. 2). Surface analysis of the Mendeley Web Catalog has shown that more than one third of LSTM works are on recognition. Almost half of publications are on classification and prediction problems. One fifth of works aimed sequence generation and other problems (Fig. 3).

2.4.2 LSTM models

In a neural network, LSTM cells can have multiple directions and dimensions. Most commonly used networks are unidirectional LSTM. In BiLSTM two LSTM cells share common input and output. Having two direction allows to learn different features from input data. Stacking multiple LSTM cells results in a hierarchical LSTM. A stacked LSTM, as a particular type of hierarchical LSTM, allows storing more information. Another type is a tree-LSTM. In a tree-structured LSTM, a single cell can reflect information from parents and child cells. This feature resembles human speech. Unidirectional, bidirectional and tree-LSTM are shown in the Figure 4. Table 3 demonstrates other models of LSTM connection and corresponding possible tasks.

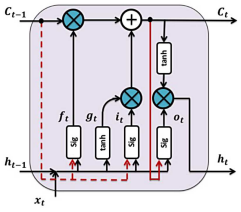
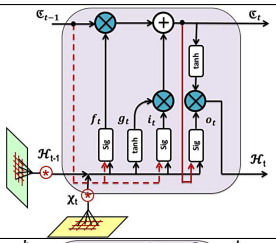
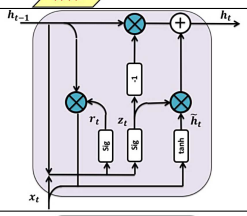
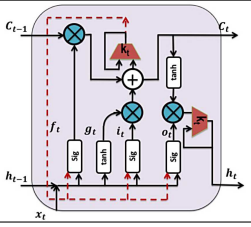
3 LSTM hardware accelerators

The high cost in computation leads to research on hardware accelerators for LSTM implementation.

3.1 Memristor-based LSTM

With advances in technology and development of memristive devices new architectures for LSTM implementation are being proposed. The core of idea of such configurations is utilization of memristor crossbar arrays to perform vector-matrix multiplication. Each memristor's conductance G_{ij} in a crossbar is programmed to retain LSTM weight values w_{ij}. When, input voltages V_i are applied to the rows of a crossbar, according to Kirchoff's Law, the resulting current values I_j of a crossbar columns give a dot product of w_{ij} and V_i: $I_j = \sum_{i=1}^{\infty} G_{ij} \cdot V_i$.

Table 2. LSTM cell configurations: ConvLSTM, GRU, Phased LSTM.

LSTM variant	Description
	Peephole LSTM LSTM with peephole connections proved to be efficient for the tasks that require precise timing. It is also called a "Vanilla" LSTM [5,7]. $i_t = \sigma(W^{(i)} \cdot x_t + U^{(i)} \cdot h_{t-1} + V^{(i)} \cdot C_{t-1} + b^{(i)})$ $f_t = \sigma(W^{(f)} \cdot x_t + U^{(f)} \cdot h_{t-1} + V^{(f)} \cdot C_{t-1} + b^{(f)})$ $g_t = \sigma(W^{(g)} \cdot x_g + U^{(g)} \cdot h_{t-1} + b^{(g)})$ $C_t = g_t \bigodot i_t + f_t \bigodot C_{t-1}$ $o_t = \sigma(W^{(o)} \cdot x_t + U^{(o)} \cdot h_{t-1} + V^{(o)} \cdot C_t + b^{(o)})$ $h_t = o_t \bigodot \tanh(C_t)$
	ConvLSTM The input of ConvLSTM is 3D data. $f_t = \sigma(W^{(f)} * \chi_t + U^{(f)} * \mathcal{H}_{t-1} + V^{(f)} \cdot \mathcal{C}_{t-1} + b^{(f)})$ $g_t = \tanh(W^{(g)} * \chi_t + U^{(g)} * \mathcal{H}_{t-1} + b^{(g)})$ $i_t = \sigma(W^{(i)} * \chi_t + U^{(i)} * \mathcal{H}_{t-1} + V^{(i)} \cdot \mathcal{C}_{t-1} + b^{(i)})$ $o_t = \sigma(W^{(o)} * \chi_t + U^{(o)} * \mathcal{H}_{t-1} + V^{(o)} \cdot \mathcal{C}_{t-1} + b^{(o)})$
	GRU GRU is a simplified LSTM cell. $z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1} + b^{(z)})$ $r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1} + b^{(r)})$ $\tilde{h}_t = \tanh(W^{(\tilde{h})} x_t + U^{(\tilde{h})}(h_{t-1} \bigodot r_t + b^{\tilde{h}})$ $h_t = (1 - z_t) \bigodot h_{t-1} + z_t \bigodot \tilde{h}_t$
	Phased LSTM at time t_j: $\tilde{C}_j = f_j \bigodot C_{j-1} + i_j \bigodot \sigma(W^{(c)} x_j + U^{(c)} h_{j-1} + b^{(c)})$ $C_j = k_j \bigodot \tilde{C}_{j-1} + (1 - k_j) \bigodot C_{j-1}$ $\tilde{h}_j = o_j \bigodot \sigma(\tilde{C}_j)$ $h_j = k_j \bigodot \tilde{h}_j + (1 - k_j) \bigodot h_{j-1}$

3.1.1 Design of analog LSTM circuit

Based on above, in [14] authors proposed a conceptual idea of analog CMOS-memristor circuit for LSTM architectures which is represented in the Figure 5. Particularly, the whole LSTM circuit is comprised of several circuits to perform matrix-vector multiplication (MVM), Hadamard (pointwise) multiplication and replicate hyperbolic tangent and sigmoid functions behaviour. In this circuit, a single weight value is set by a conductance difference of two memristors, e.g. $w_{i-1,1} = G_{i-1,1} - G_{i-1,2}$. This is done to achieve wider range of resulting conductance of different polarity. Activation layers were composed of CMOS-memristor circuits of the same architecture for both sigmoid and hyperbolic tangent functions. The output characteristics of activation functions were tuned by bulk voltages V_{p1}, V_{p2}, V_{n1} and V_{n2} of MOSFET transistors P_1, P_2, N_1 and N_2. A pointwise multiplication was performed by a single transistor T_1. Since this transistor could perform only in Ohmic region, several aspects were taken into account. For instance, voltages applied to the drain and gate of T_1 could take any values within the ranges of $(-0.45; 0)$ V and $(0; 0.45)$ V respectively. Therefore, differential amplifiers and inverters with switches S_1

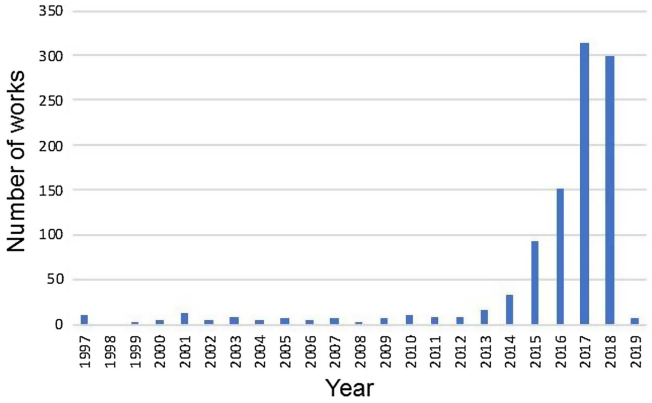


Fig. 2. Number of publications on LSTM during 1997–2019 (accessed on February, 2019).

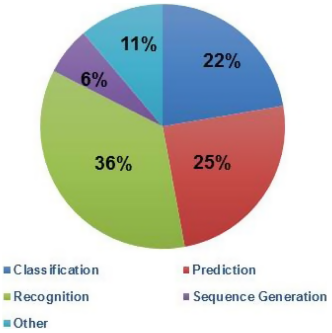


Fig. 3. LSTM Applications during 1997–2019 (accessed on February, 2019).

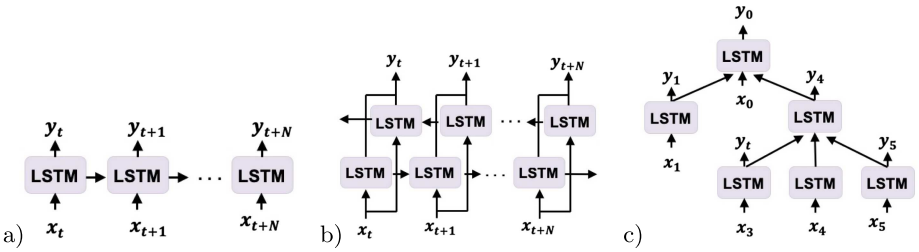
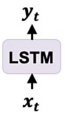
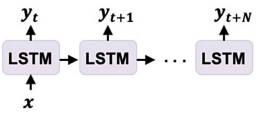
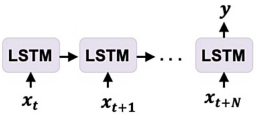
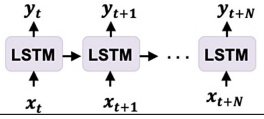


Fig. 4. (a) Unidirectional LSTM; (b) BiLSTM; and (c) tree-structured LSTM.

and S_2 were used to control the amplitude and polarity of input and output signals. All simulations were done for 180 nm CMOS technology and HP memristor.

Later, in order to avoid current-to-voltage and voltage-to-current conversions between circuits, CMOS-memristor circuit of activation layer was replaced by CMOS circuit that works in current domain. The proposed current-based architecture was validated in [13] for airplane passengers number *prediction* problem. Due to limitations of the transistor used for elementwise multiplication, in [1] T_1 was substituted by a voltage-based CMOS circuit. The new circuit was tested for the same prediction task and the Table 4 compares performance of the proposed designs. As it can be seen a voltage-based LSTM demonstrates higher accuracy than a current-based LSTM. Moreover, in [2] a voltage-based LSTM circuit outperformed a memristor-based single

Table 3. LSTM models.

Model	Schematic	Description
One-to-One		One input and one output model is suitable for classification tasks. The networks can be unrolled for several timesteps.
One-to-Many		This model allows to convert a single input to sequences.
Many-to-One		Prediction; Classification.
Many-to-Many		Sequence-to-Sequence generation (video, text, music); Speech Recognition; Machine translation; Video-to-Text; Image-to-Text.

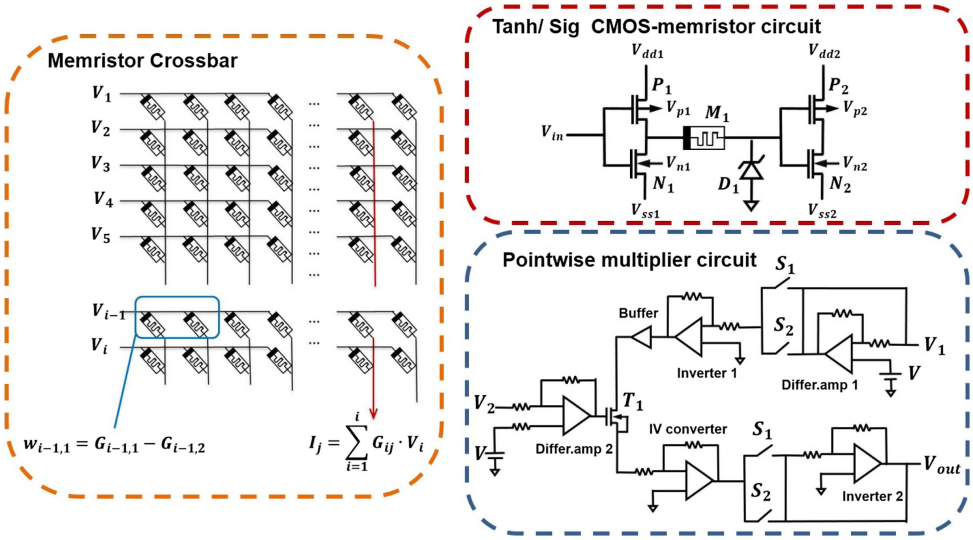


Fig. 5. Design of LSTM circuit by [14]: a memristor crossbar array circuit, activation function circuit and pointwise multiplier circuit.

Table 4. A comparison table of current-based and voltage-based circuit designs for airplane passengers prediction task.

LSTM architecture	Power consumption	Area	RMSE (software)	RMSE (circuit)
Current-based LSTM [13]	105.9 mW	77.33 μm^2	55.26%	47.33%
Voltage-based LSTM [2]	225.67 mW	108.60 μm^2	10.05%	10.99%

perceptron, DNN, ANN and modified HTM architectures in *classification* of normal and abnormal wafers with accuracy more than 95%.

LSTM weight matrix values to implement prediction and classification tasks were extracted via simulation using Python Keras. Circuit simulations for prediction and classification problems were performed in LTSpice.

3.1.2 LSTM using Ta/HfO_2 memristor crossbar array

Figure 7 shows the data flow in the system proposed by [10]. In this work, LSTM and fully-connected layers were implemented in situ in a 1T1R crossbar array with Ta/HfO_2 memristors on top of it. Multilevel memristors in a crossbar array were programmed with predefined conductance values using write-and-verify method. At time t input x_t was applied to the rows of a LSTM layer crossbar and output voltages h_t were read from crossbar columns using virtual ground. Afterwards, h_t is fed into fully-connected layer and back to LSTM as recurrent input h_{t-1}. If size of x_t is N and size of h_{t-1} is M, then including biases the size of a single LSTM unit crossbar is $[(M + N + 1), (4M)]$. Each matrix weight was encoded with conductance difference of two memristors. Therefore, the size of a crossbar doubles: $[2(M + N + 1), 2(4M)]$. Gated and nonlinear units in the architecture were implemented in software. So off-chip operations include $3M$ sigmoid, $2M$ hyperbolic tangent, $3M$ elementwise multiplications and $2M$ additions. The proposed architecture was tested for regression and classification problems. More detailed description of on-chip and off-chip operations is given below.

The size of a memristor crossbar used to implement LSTM layer on the chip is 128×64. For each problem, crossbar conductance values were extracted individually using Backpropagation through time (BPTT) algorithm in software because hardware training is still challenging and computationally expensive task. 2 μm transistors in a crossbar serve as selector devices when $V_{\text{gate}} = 5V$. The weights were updated using two-pulse scheme as follows:

1. Decreasing memristor conductance required two consecutive cycles, e.g. "Reset" and "Set".
   RESET cycle: $V_{\text{reset}} = 1.7\,\text{V}$ (about 1 s) is applied to a bottom electrode of a memristor.
   SET cycle: $V_{\text{reset}} = 2.5\,\text{V}$ (about 2 s) is applied to a top electrode of a memristor.
2. Increasing memristor conductance involves only SET cycle.

An individual memristor conductance switches in around 5 ns. In this work, the process of switching memristors conductance in array was a serial communication between microcontroller and off-chip part described below. Authors assume possibility of faster switching via more sensitive measurement equipment and on-chip parallel weight update.

Figure 8 shows the off-chip measurement system which is controlled by microcontoller. Eight 16-bit digital-to-analog converters (DACs) allow to supply voltage to all 128 rows of a crossbar at the same time. The dot-product values are equal to current readings from each column. Before being fed into analog-to-digital converter, current values are converted into voltages. This system requires further optimization to speed up the process and decrease power consumption.

Regression task for predicting airplane passengers number was deployed using a two-layer recurrent neural network (RNN). The first layer unrolls LSTM unit 15 times and form of Many-to-Many model. Each LSTM unit requires a (34×60) memristor crossbar. The second layer is a fully-connected neural network (32×1). Human gait *classification* problem uses Many-to-One model. Its first layer is comprised of 14

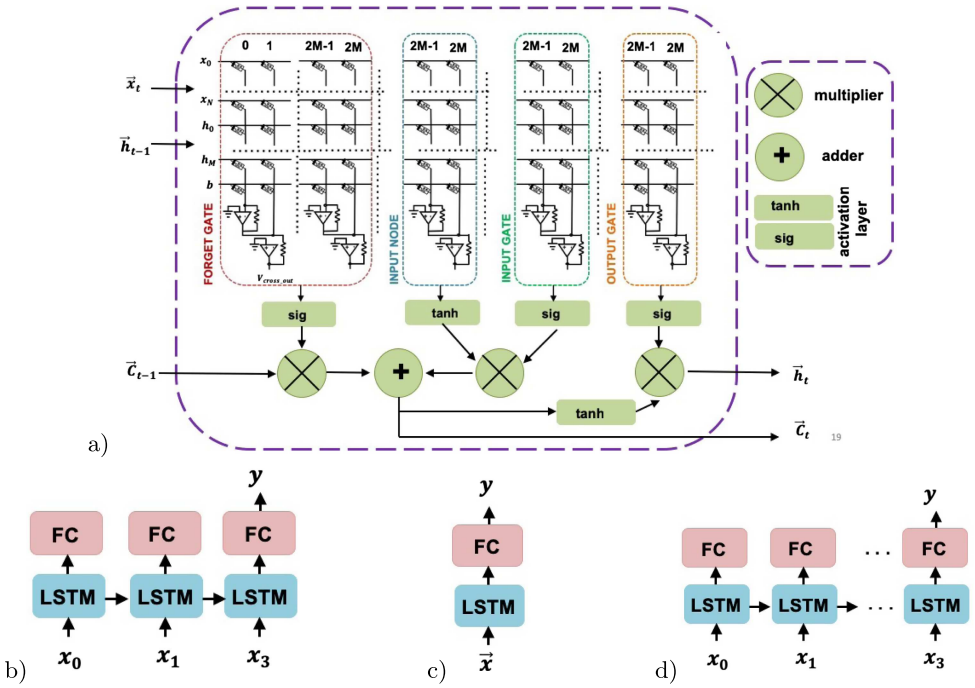


Fig. 6. Implementation of prediction and classification problems by [1,2]: (a) schematic of LSTM unit; (b) model used for prediction of airplane passenger number (based 144 observations during 1949–1960): LSTM is untolled for 3 timesteps, the length of x_t $N = 1$, the length of h_t $M = 4$; (c) model used for wafer quality classification to normal and abnormal classes: 1 timestep; $N = 152$; $M = 4$; (d) model used for wafer quality classification to normal and abnormal classes: $N = 1$, $M = 1$, 152 timesteps.

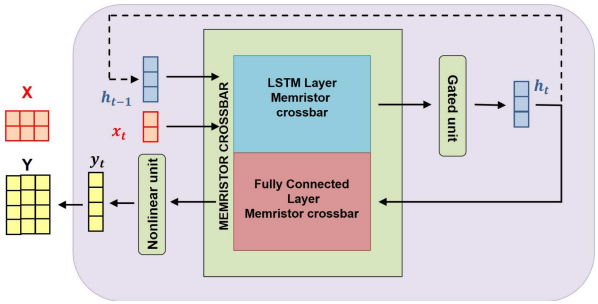


Fig. 7. Data flow in the architecture proposed by [10]: Ta/HfO_2 memristor corssbar array and software-based gated and nonlinear units.

LSTM units. The size of a crossbar used by one LSTM cell is (128×56). The second layer is also fully-connected layer (28×8).

Talking about scalability of the system, authors suggest two ways to deal with the large memristor crossbar arrays which sizes constrained by their output current. One of the approaches is to decrease input signal amplitudes at condition of acceptable signal-to-noise ratio rates. Another method is to maintain conductance at low level which is controlled via memristor materials.

All works above have shown the consistency in the implementation of LSTM on memristor crossbar array. The main limitation of such architectures is utilization of

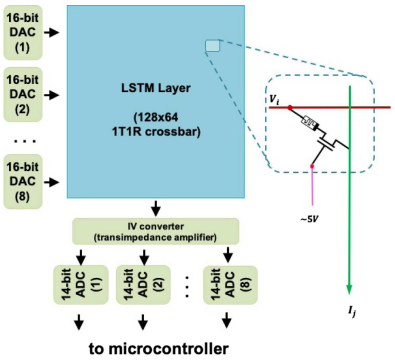


Fig. 8. Measurement system of Ta/HfO_2 memristor corssbar array proposed by [10].

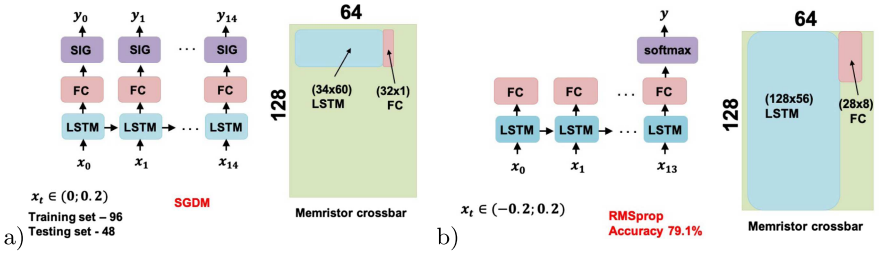


Fig. 9. Implementation of regression and classification problems by [10]. (a) Prediction of airplane passenger number (based 144 observations during 1949–1960); (b) human gait classification on USF-NIST dataset.

pre-trained networks' weight values. Control circuit for real-time weight learning and updating is required.

3.2 Other accelerators

Between 2015–2018 several research on FPGA-based LSTM was conducted. One of the most efficient peak performances of FPGA LSTM is 13.45 GOP/s which is 28.76× acceleration compared to CPU [15,17]. In [3] authors built LSTM hardware accelerator with systolic array for matrix-vector multiplication. It was named "Chipmunk" and the peak performance has been shown 3.08 GOP/s.

4 Conclusion

Information storage and processing in the human brain memory is a distributive system. LSTM is a state-of-the-art tool for processing various sequential and temporal data such as speech, video, stock data and others. Presence of internal memory in LSTM allows to maintain long-term dependencies. In this work we discussed the most popular configurations of LSTM and its capacity and potential. Despite advantages, LSTM neural networks are slow due to large parallelism and sequential nature. Hopefully this will be solved by presenting a reliable hardware accelerator in near future. The utilization of memristors may help to overcome a "bottleneck" of modern computers to implement vector-matrix multiplication and achieve better non-linearity for computation of various tasks.

References

1. K. Adam, K. Smagulova, A.P. James, Memristive LSTM network hardware architecture for time-series predictive modeling problems, in *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)* (IEEE, 2018), pp. 459–462
2. K. Adam, K. Smagulova, O. Krestinskaya, A.P. James, *Wafer quality inspection using memristive LSTM, ANN, DNN, and HTM*, arXiv:11809.10438 (2018)
3. F. Conti, L. Cavigelli, G. Paulin, I. Susmelj, L. Benini, Chipmunk: A systolically scalable 0.9 mm 2, 3.08 Gop/s/mW at 1.2 mW accelerator for near-sensor recurrent neural network inference, in *Custom Integrated Circuits Conference (CICC)* (IEEE, 2018), pp. 1–4
4. F.A. Gers, J. Schmidhuber, F. Cummins, *Learning to forget: Continual prediction with LSTM* (IEEE, London, 1999), pp. 850–855
5. F.A. Gers, N.N. Schraudolph, J. Schmidhuber, J. Mach. Learn. Res. **3**, 115 (2002)
6. A. Gomez, Backpropogating an LSTM: a numerical example, Aidan Gomez blog at Medium, 2016
7. K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, IEEE Trans. Neural Netw. Learn. Syst. **28**, 2222 (2017)
8. S. Hochreiter, J. Schmidhuber, Neural Comput. **9**, 1735 (1997)
9. A. Karpathy, *The unreasonable effectiveness of recurrent neural networks, 2015* (2016), http://karpathy.github.io/2015/05/21/rnn-effectiveness
10. C. Li, Z. Wang, M. Rao, D. Belkin, W. Song, H. Jiang, P. Yan, Y. Li, P. Lin, M. Hu, N. Ge, Nat. Mach. Intell. **1**, 49 (2019)
11. Z.C. Lipton, J. Berkowitz, C. Elkan, *A critical review of recurrent neural networks for sequence learning*, arXiv:1506.00019 (2015)
12. C. Olah, *Understanding LSTM networks* (2015)
13. K. Smagulova, K. Adam, O. Krestinskaya, A.P. James, *Design of cmos-memristor circuits for lstm architecture*, arXiv:1806.02366 (2018)
14. K. Smagulova, O. Krestinskaya, A.P. James, Analog Integr. Circ. Sig. Process. **95**, 467 (2018)
15. Z. Sun, Y. Zhu, Y. Zheng, H. Wu, Z. Cao, P. Xiong, J. Hou, T. Huang, Z. Que, FPGA acceleration of lstm based on data for test flight, in *2018 IEEE International Conference on Smart Cloud (SmartCloud)* (IEEE, 2018), pp. 1–6
16. I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in *Advances in Neural Information Processing Systems* (2014), pp. 3104–3112
17. Z. Zhao, A. Srivastava, L. Peng, Q. Chen, ACM J. Emerg. Technol. Comput. Syst. **15**, 13 (2019)