# Economic LSTM Approach for Recurrent Neural Networks

Kasem Khalil, *Member, IEEE*, Omar Eldash, *Member, IEEE*, Ashok Kumar, *Senior Member, IEEE*, and Magdy Bayoumi, *Fellow, IEEE*

***Abstract*—Recurrent Neural Networks (RNNs) have become a popular method for learning sequences of data. It is sometimes tough to parallelize all RNN computations on conventional hardware due to its recurrent nature. One challenge of RNN is to find its optimal structure for RNN because of computing complex hidden units that exist. This brief presents a new approach to Long Short-Term Memory (LSTM) that aims to reduce the cost of the computation unit. The proposed Economic LSTM (ELSTM) is designed using a few hardware units to perform its functionality. ELSTM has fewer units compared to the existing LSTM versions which makes it very attractive in processing speed and hardware design cost. The proposed approach is tested using three datasets and compared with other methods. The simulation results show the proposed method has comparable accuracy with other methods. At the hardware level, the proposed method is implemented on Altera FPGA.**

***Index Terms*—LSTM, neural network, deep learning, image recognition, hardware recurrent neural networks, ELSTM.**

## I. Introduction

Artificial intelligence has been commonly used for solving many problems, in different domains. Machine Learning is a form of artificial intelligence, and it can be used to automate decision making and prediction making. It can be used for image classification, pattern recognition (character recognition, face recognition, etc), object detection, time series prediction, natural language processing, and speech recognition. Machine Learning (ML) has multiple structures: Convolutional Neural Networks (CNN) [1], Feed-Forward Deep Networks (FFDN) [2], and Recurrent Neural Networks (RNN) [3].

RNN has been efficiently used in many applications such as speech recognition [4], language translation, image captioning, and recognizing actions in videos [5]. RNN can be considered a deep model when it is unrolled along the time axis. One main advantage of RNN is that RNN can learn from previous data/information. The key point is what to remember and how far a model remembers. In the standard RNN, recent past information is used for learning [6]. The drawback is that it can not learn long-term expectations or dependencies due to vanishing or exploding gradients [7]. To overcome this deficiency, Long Short-Term Memory (LSTM) has been proposed [8]. LSTM is an architecture of RNN where it adds memory controllers for deciding when to forget, remember, and output. This allows an expansion of the training procedure, to learn long-term dependencies. Xue *et al.* [9] present hierarchical LSTM Model to consider both the scene layouts and influence of social neighborhood to predict pedestrians' future trajectory. In this approach Social-Scene-LSTM (SS-LSTM), three different LSTMs have been used to capture the social, personal, and scene scale information. circular shape of the neighborhood is used instead of a rectangular shape in the social scale. The approach is tested using three datasets, and the simulation result shows the prediction accuracy is improved due to using a circular shape neighborhood. In a hardware implementation, RNN hardware design is not done on all neural networks. Chang *et al.* [10] present a hardware implementation of RNN on FPGA. This hardware implementation is implemented on the programmable logic Zynq 7020 FPGA from Xilinx for LSTM. The implementation has 2 layers with 128 hidden units, the method is tested using a character level language model. Performance per unit power of different embedded platforms is studied.

LSTM block diagram consists of three gates and two activation functions as shown in Fig. 1. The first step of LSTM is to decide which information should be forgotten from the cell state and this is called "forget gate". The second step is to make a decision on which new information will be stored in the cell. This is done by "input gate" which decides values

Manuscript received May 3, 2019; revised May 29, 2019 and August 21, 2019; accepted September 1, 2019. Date of publication June 25, 2019; date of current version November 1, 2019. This brief was recommended by Associate Editor C. K. Tse. *(Corresponding author: Kasem Khalil.)*

The authors are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504 USA (e-mail: kmk8148@louisiana.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSII.2019.2924663

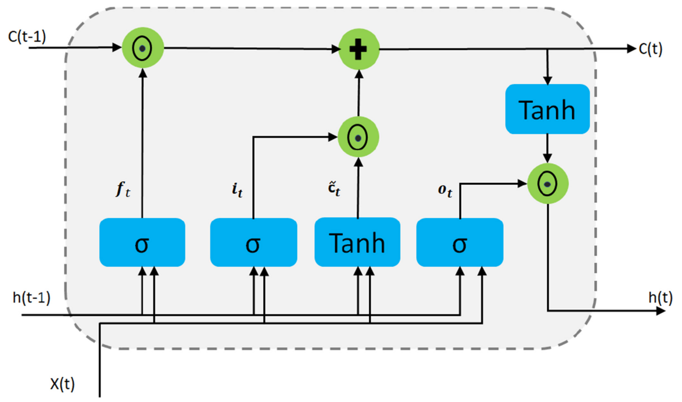

Fig. 1. Long Short-Term Memory (LSTM) structure.

1549-7747 © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY. Downloaded on November 17,2022 at 15:42:11 UTC from IEEE Xplore. Restrictions apply.

to be updated and tanh which creates new candidate values. Finally, the output layer which decides data will go to output. The equation of each part is calculated as follows [11].

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (1)$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (3)$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \quad (4)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (5)$$
$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

where the matrix multiplication $W_f[h_{t-1}, x_t] = W_h h_{t-1} + W_x x_t$, $f_t$ is the result of the forget gate, $i_t$ is the input gate result, and $o_t$ is the output gate result. The new state memory is $\tilde{c}_t$, the final state memory is $c_t$, and the cell output is $h_t$. The weights of the forget gate, input gate, output gate are $W_f$, $W_i$ $W_o$, respectively. The biases are $b_f$, $b_i$, and $b_o$ for the forget, input, and output layer, respectively. The symbol of $\odot$ represents the elementwise (Hadamard) multiplication, $\sigma$ is the logistic sigmoid function, and *tanh* is the hyperbolic tangent function.

LSTM has been proposed with variations to make it simple and improve its performance. Greff *et al.* [12] present a coupled-gate LSTM, in which the forget and input gate are coupled into one. Therefore, the structure has one gate less which makes it simpler than LSTM. The consequence is that the coupled-gate LSTM leads to reduced computational complexity and slightly higher accuracy. Cho *et al.* [13] present another LSTM variation which is called Gated Recurrent Unit (GRU) architecture. Instead of using three gates in LSTM, GRU includes two gates: update gate and rest gate. The update gate operation combines the forget and input gate while the rest gate has the same functionality as the output layer. GRU model simplified LSTM by eliminating the memory unit and the output activation function. Zhou *et al.* [14] simplified LSTM by using only one gate, named as Minimal Gated Unit (MGU). MGU does not include the memory cell which is similar to GRU. GRU model has faster training, higher accuracy, and fewer trainable parameters compared to LSTM. Elsayed *et al.* [15] present a reduced-gate convolutional LSTM (rgcLSTM) architecture which is another one-gate method. It uses a memory cell, and it has a peephole connection from the cell state to the network gate.

A variation of LSTM structure is adopted in this brief to reduce training parameter and increase the training speed while retaining or increasing the performance. The results of all versions of LSTM show that models with fewer parameters may provide higher accuracy. The focus of this brief is to propose and discuss a new LSTM structure which utilizes one gate and two activation functions. The proposed method has a simple structure and acceptable performance comparable to the previous versions of LSTM. The remainder of this brief is organized as follows: Section II presents the proposed architecture of the ELSTM. Section III presents the implementation and the experimental results of the proposed method followed by the conclusion in Section IV.

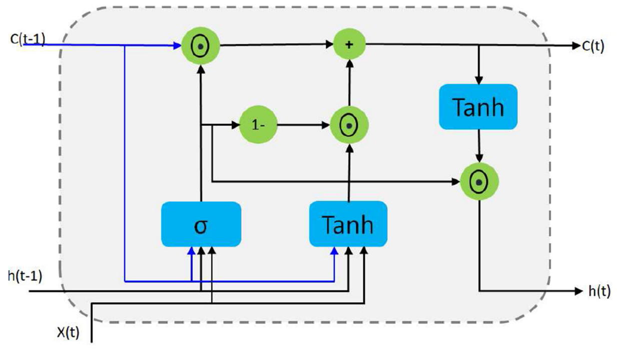

Fig. 2. The block diagram of the proposed approach in weight update flow.

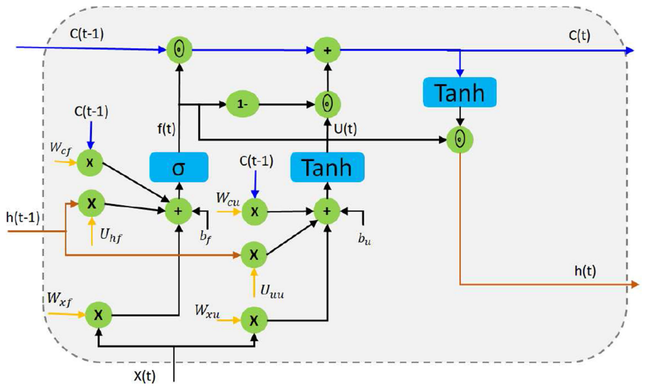

Fig. 3. The conceptual schematic diagram of the proposed method.

## II. The Proposed LSTM Structure

The proposed ELSTM provides a new architecture of LSTM using one gate. This approach is simple in terms of components which makes it more suitable in low-cost hardware design. The proposed method is shown in Fig. 2, in the structure, the gate provides the mechanism of deleting and updating data. The output of this gate feeds three parts: the memory layer, update layer, and output layer. In the memory layer, using $x(t), h(t-1), and\ c(t-1)$, the output provides values of this gate $f_t$ which is multiplied with the memory state for a forgetting step. This gate couples the forget (update) gate and the input (reset) gate. The forget gate $f_t$ is generated, and then the elementwise product between $1-f_t$ and the output of *tanh* which is calculated by $x_t$, $h_{t-1}$, and $c_{t-1}$. The memory state is used in the calculation to get accurate performance and stability in terms of forgetting and updating to improve the learning performance compared to MGU. Fig. 3 shows the structural conceptual details of arithmetic, weights, and biases. The output of the gate is given by

$$f(t) = \sigma\left(W_f.I_f + b_f\right) \quad (7)$$
$$f(t) = \sigma\left([W_{cf}, W_{xf}, U_{hf}].[x(t), c(t-1), h(t-1)] + b_f\right) \quad (8)$$

where $I_f$ is a general input in this case, $f(t) \in R^{d\times h\times n}$ while d is the width, h is the height, and n is the number of channels of $f_t$. x(t) $\in R^{d\times h\times r}$ is the input which may be an image, speech, etc, and r is the number of input channels. $h(t-1)$ is the output of the block at the time of $(t-1)$, the stack representing the internal state at the time of $(t-1)$ is called $c(t-1)$. The same as $f(t)$, $h(t-1)$ and $c(t-1) \in R^{d\times h\times n}$. For

Authorized licensed use limited to: MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY. Downloaded on November 17,2022 at 15:42:11 UTC from IEEE Xplore. Restrictions apply.

the weights, $W_{xf}$, $W_{cf}$, and $U_{hf}$ are the convolutional weights, and they have dimension $(m \times m)$ for all the kernels, and $b_f$ is the bias which has a dimension of $n \times 1$. The input update equation is obtained by.

$$u(t) = \tanh(W_u.I_u + b_u) \tag{9}$$

$$u(t) = \tanh([W_{cu}, W_{xu}, U_{uu}].[x(t), c(t-1), h(t-1)] + b_u) \tag{10}$$

where $I_u$ is a general input, $u(t) \in R^{d\times h\times n}$, and it matches the dimension of $f_t$. $b_u \in R^{n\times 1}$ also matches the dimension of $b_f \in R^{n\times 1}$. The output $U_t$ of *tanh* is multiplied by $1 - f_t$, the multiplication result will be added with the memory state to generate the updated memory state. The new state is used to produce the desired output using *tanh* and the output gate $f_t$. The equations of the memory state and the output are finalized by the following equations.

$$C(t) = f(t) \odot C(t-1) + (1 - f(t)) \odot U(t) \tag{11}$$

$$h(t) = f(t) \odot \tanh(C(t)) \tag{12}$$

where C(t) is the final memory state, h(t) is the final output, and the $\odot$ symbol represents elementwise multiplication. The comparison of computation components for LSTM, coupled-gate LSTM, MGU, GRU, and ELSTM is shown in Table I. This comparison shows the computation components of each structure in terms of the state memory cell, number of gates, number of activation function, number of elementwise multiplication, number of elementwise summation, and number of weight matrices. The proposed ELSTM has a minimum number of gates (one gate) and two activation function which is lower than LSTM and coupled-gate LSTM. The number of elementwise multiplication, number of elementwise summation, and number of weight matrices are comparable to the other methods. The benefit of using fewer components in the proposed LSTM increases the computation speed. It also reduces the cost of designing in hardware level.

## III. The Implementation and Experimental Results

In this section, an evaluation of the proposed ELSTM will be discussed. ELSTM is implemented and tested using three datasets: MNIST, IMDB, and ImageNet datasets. Firstly, the MNIST dataset contains handwritten digits images (0-9) [16]. It includes 10,000 images in the test set phase and 60,000 images in the training set phase. These images are preprocessed to make the center of these digits' mass to be at the central position of image size $28 \times 28$. MNIST has been commonly used for deep neural network classification. The testing is done by using each row (28 pixels) as a single input. We use 100 hidden units with the batch size of 100, and the learning rate is $10^{-10}$ with a momentum of 0.99. ELSTM's accuracy is higher than that of LSTM as shown in Fig. 4. ELSTM achieves an accuracy of 90.89% while the LSTM is 87.21% at the end of 20,000 epochs.

ELSTM is tested again by MNIST using another different method which takes every pixel as one component in the input sequence. Therefore, an image spreads to become a sequence of length 784. The pixels scanning runs from left to right and top to bottom. The aim of this task is to test ELSTM performance in long sequence length. The simulation result shows ELSTM has an accuracy of 84.91% after 20,000 epochs and LSTM has an accuracy of 65% after 900,000 epochs.

The second testing of ELSTM is used to study the classification of sentiment in IMDB movies reviews. It separates the status of the reviews into a positive and negative review. This dataset contains 25,000 movie reviews for testing and another 25,000 for training. The sequence length has a maximum length of 128. The proposed LSTM is implemented using 100 hidden units with a batch size of 16, and $10^{-8}$ learning rate with 0.99 momenta. The simulation result shows the ELSTM has an accuracy of 65.03% while LSTM has an accuracy of 61.43% after 20,000 epochs as shown in Fig. 5. Therefore,

TABLE I
Computation Components Comparison

| Metrics | LSTM | Coupled-Gate LSTM | MGU | GRU | ELSTM |
|---|---|---|---|---|---|
| State memory cell | Yes | Yes | No | No | Yes |
| Number of gates | 3 | 2 | 1 | 2 | 1 |
| Number of activation function | 2 | 2 | 1 | 1 | 2 |
| Number of elementwise multiplication | 3 | 4 | 3 | 3 | 3 |
| Number of elementwise summation | 1 | 1 | 1 | 1 | 1 |
| Number of weight matrices | 8 | 6 | 6 | 6 | 6 |

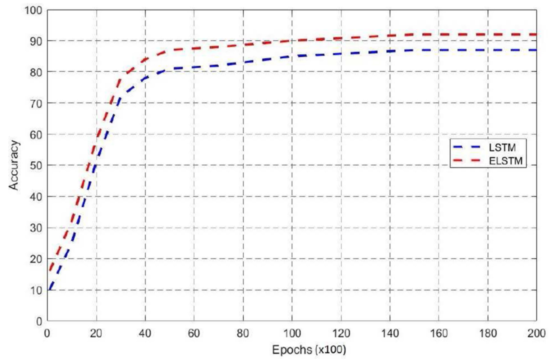

Fig. 4. The accuracy simulation result using MNIST dataset.

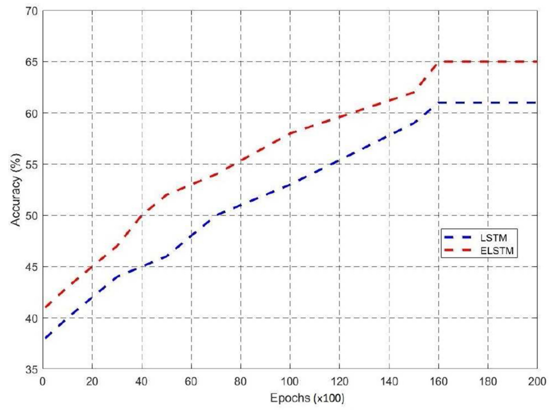

Fig. 5. The accuracy simulation result using IMBD dataset.

Authorized licensed use limited to: MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY. Downloaded on November 17,2022 at 15:42:11 UTC from IEEE Xplore. Restrictions apply.

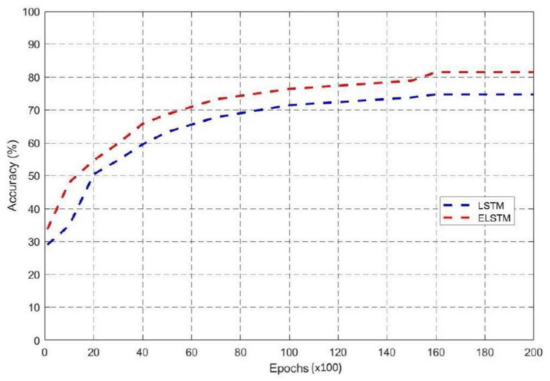

Fig. 6. The accuracy simulation result using ImageNet dataset.

TABLE II
ACCURACY COMPARISON IN DIFFERENT STRUCTURES USING MULTIPLE DATASETS

| Method | MNIST | IMDB | ImageNet |
|---|---|---|---|
| LSTM | 87.21% | 61.43% | 75.12% |
| Coupled-gate LSTM | 88.15% | 62.82% | 76.79% |
| GRU | 88.27% | 63.23% | 77.83% |
| MGU | 89.06% | 63.96% | 79.23% |
| ELSTM | 90.89% | 65.03% | 82.39% |

ELSTM has higher accuracy than LSTM, and ELSTM is faster than LSTM due to a few computation components.

Thirdly, ELSTM is tested using ImageNet dataset. ImageNet includes 3.2 million cleanly labeled full resolution images with 12 subtrees with 5247 synonym set or synsets [17]. The ELSTM is implemented using 100 hidden units with a batch size of 16, and $10^{-8}$ learning rate with 0.99 momenta. The simulation result shows the ELSTM has 82.39% accuracy while LSTM has an accuracy of 75.12% after 20,000 epochs as shown in Fig. 6. In this test, the proposed ELSTM has better performance than the traditional LSTM.

The simulation result of all tests using different datasets shows the proposed ELSTM has better performance than LSTM. For deep evaluation, the proposed method is compared with multiple LSTM structure such as coupled-gate LSTM, MGU, and GRU using the three datasets (MNIST, IMDB, and ImageNet) as shown in Table II. This comparison shows the proposed method has a comparable performance with these LSTM structures. Therefore, the evaluation of ELSTM is presented using different structures and multiple datasets.

We studied the error evaluation using mean squared error (MSE) and mean absolute error (MAE) [18]. The MSE measures the average of the squares of the errors, or is the average squared difference between the desired value and what is estimated. The MAE is a measure of the difference between two continuous variables. Each one is calculated using the following equations.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y})^2 \tag{13}$$

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \tilde{y} \mid}{n} \tag{14}$$

TABLE III
ERROR COMPARISON

| Method | MSE | MAE |
|---|---|---|
| LSTM | 1.834 | 2.062 |
| Coupled-gate LSTM | 1.534 | 2.201 |
| GRU | 1.189 | 2.193 |
| MGU | 1.13 | 2.15 |
| ELSTM | 1.012 | 1.190 |

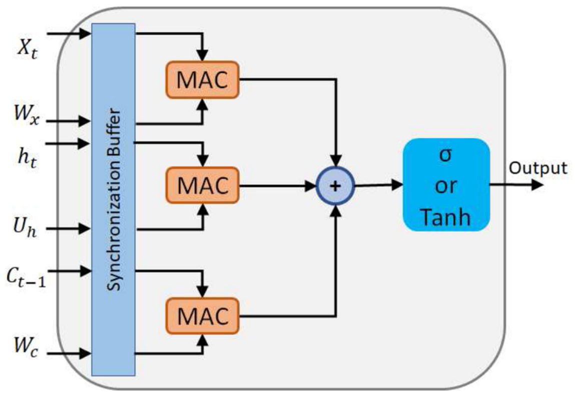

Fig. 7. The hardware gate module.

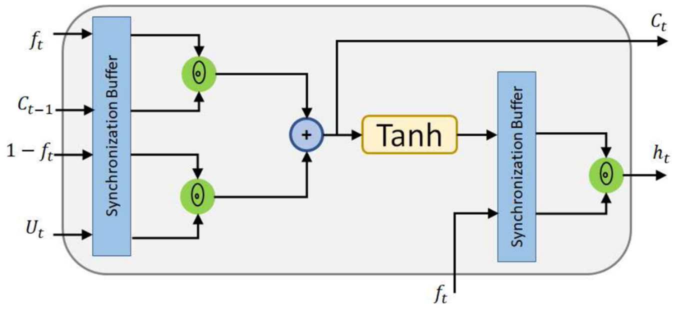

Fig. 8. The hardware module of the ELSTM final stage.

where $y_i$ is the resulted value, $\tilde{y}$ is the estimated value, and n is the number of results. The measurements of both MSE and MAE are shown in Table III. These results are obtained using ImageNet dataset, the result shows the proposed method has a comparable error with the other models. The proposed method provides comparable accuracy with LSTM because it has few factors to tune and memory state has been used for forgetting and updating inputs, therefore, it will be easier to find the best performance for the proposed method. ELSTM also has a low error due to these factors, and it has faster training which allows it to achieve high accuracy faster.

In hardware design, hardware module of the gate is shown in Fig. 7. The input streams may not be synchronized even if the module triggers the ports at the same time. Consequently, a stream synchronization is required, the synchronization buffer is used to cache streaming data until all streaming ports are finished. This operation is needed to ensure that the matrices are aligned to feed Multiplier Accumulator (MAC) which performs multiplication and addition operation on the inputs. The final block can be for sigmoid ($\sigma$) function or *tanh* function. The Module of obtaining $C_t$ and $h_t$, from the results of the gates, is presented in Fig. 8. Two synchronization buffers are

Authorized licensed use limited to: MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY. Downloaded on November 17,2022 at 15:42:11 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
HARDWARE IMPLEMENTATION RESULTS

| | ELSTM | | LSTM | |
|---|---|---|---|---|
| Logic Utilizing | Used | Utilization | Used | Utilization |
| Number of slice registers | 6321 | 6.12% | 8610 | 8.34% |
| Number of Slice LUTs | 3258 | 7.18% | 4043 | 8.92% |
| Number of BUFs | 3 | 2.66% | 5 | 4.3% |
| Number of DSPs | 20 | 275 | 45 | 16.36% |
| Number of Block RAM | 11 | 8.59% | 16 | 12.5% |
| Memory LUTs | 312 | 2.49% | 496 | 3.98% |
| FF | 832 | 1.06% | 2552 | 3.26% |
| Operating frequency (MHZ) | 120 | | 120 | |
| Time latency (ms) | 23 | | 35 | |
| Power consumption (W) | 1.192 | | 1.847 | |
| Area reduction | 34% | | | |

used, one is in the input side and the second one is in the output side. The ELSTM architecture will be implemented using three modules: two modules as Fig. 7 (One of them includes sigmoid block and the other one includes the *tanh* block) and one module as Fig. 8. The proposed method is tested and implemented using VHDL and Altera Arria 10 GX FPGA 10AX115N2F45E1SG. The operating frequency is 120 MHZ. The simulation results of the hardware implementation are shown in Table IV in terms of registers, LUTs, DSPs, Buffers, block RAM, Flip Flop (FF), etc. These results present the used resources which are the consumed resources and utilization which is the ratio of used resources to the total available resources. The resources utilization for MGU in terms of registers, LUTs, Buffers, DSPs, block RAM, LUTs, and FFs are 8.37%, 8.96%, 4.5%, 16.42%, 12.7%, 4%, and 3.31, respectively. Thus, the consumed resources for the proposed method is lower than the state-of-the-art which represents the reduction of consumer hardware. The hardware result also shows that the proposed method has a lower area by 34% compared to the original LSTM. The proposed method has a latency of 23 ms which is lower than the latency (35 ms) of original LSTM. The ELSTM also has a throughput of 258.4 MOPS while the LSTM has a throughput of 173.5 MOPS. Furthermore, the power consumption of ELSTM is 1.192 W, and LSTM has a 1.847 W power consumption. Thus, the proposed method is attractive to the hardware level due to the lower hardware cost of designing the proposed method. In addition, the proposed method is faster than the traditional LSTM due to using few components compared to LSTM.

## IV. CONCLUSION

The cost of LSTM units and the processing speed direct researchers to optimize the LSTM method. This brief presents a new approach ELSTM which uses fewer units for processing. The proposed method has the ability to retain memory and learn data sequences using one gate. The proposed method is tested using three datasets: MNIST, IMDB, and ImageNet. The simulation results show the proposed method has a valuable accuracy. For deep evaluation, the proposed method is compared with the current method using the three datasets. The results show the proposed method has a comparable accuracy compared to the other method while the proposed method has a few units. The benefit of the proposed method is performing a learning process at a faster speed due to the lower number of computation units. The proposed method design is also attractive in low-cost hardware design. The proposed method is tested and implemented using Altera Arria 10 GX FPGA. The proposed method saves 34% of the area and 35% of the power consumption compared to LSTM.

## REFERENCES

[1] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1725–1732.

[2] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.

[3] D. P. Mandic and J. Chambers, *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Chichester, U.K.: Wiley, 2001.

[4] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 6645–6649.

[5] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[7] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *Proc. IEEE Win. Conf. Appl. Comput. Vis. (WACV)*, 2018, pp. 1186–1194.

[10] A. X. M. Chang, B. Martini, and E. Culurciello, "Recurrent neural networks hardware implementation on FPGA," *arXiv preprint arXiv:1511.05552*, 2015.

[11] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, 2005.

[12] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[13] K. Cho *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[14] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *Int. J. Autom. Comput.*, vol. 13, no. 3, pp. 226–234, 2016.

[15] N. Elsayed, A. S. Maida, and M. Bayoumi, "Reduced-gate convolutional lstm using predictive coding for spatiotemporal prediction," *arXiv preprint arXiv:1810.07251*, 2018.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[17] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[18] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

Authorized licensed use limited to: MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY. Downloaded on November 17,2022 at 15:42:11 UTC from IEEE Xplore. Restrictions apply.