ELC-ECG: Efficient LSTM Cell for ECG Classification based on Quantized Architecture

Seyed Ahmad Mirsalari*, Najmeh Nazari*, Seyed Ali Ansarmohammadi* Sima Sinaei+, Mostafa E. Salehi*, Masoud Daneshtalab+
*School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
+Division of Intelligent Future Technologies, Malardalen University, Vasteras, Sweden
{ahmad.mirsalari, najme.nazari, sansarmohammadi, mersali}@ut.ac.ir, {sima.sinaei, masoud.daneshtalab}@mdh.se

***Abstract*— Long Short-Term Memory (LSTM) is one of the most popular and effective Recurrent Neural Network (RNN) models used for sequence learning in applications such as ECG signal classification. Complex LSTMs could hardly be deployed on resource-limited bio-medical wearable devices due to the huge amount of computations and memory requirements. Binary LSTMs are introduced to cope with this problem. However, naive binarization leads to significant accuracy loss in ECG classification. In this paper, we propose an efficient LSTM cell along with a novel hardware architecture for ECG classification. By deploying 5-level binarized inputs and just 1-level binarization for weights, output, and in-memory cell activations, the delay of one LSTM cell operation is reduced 50x with about 0.004% accuracy loss in comparison with full precision design of ECG classification.**

Keywords—Long Short -Term Memory (LSTM), Wearable Devices, Electrocardiogram (ECG) Signal Classification.

I. INTRODUCTION

Cardiovascular diseases (CVDs) such as myocardial infarction, cardiomyopathy and myocarditis are the leading causes of death in the world. Myocardial Infarction (MI) is among the most important CVDs and early detection of myocardial infarction are of great significance, which can ensure the life safety of patients. Electrocardiogram (ECG) is the most commonly detection tools, and it is spontaneous to detect myocardial infarction using ECG with artificial intelligence method. Wearable devices provide a platform for continuous monitoring of patients' heartbeats in daily life [1]. Our approach is to locally execute the ECG classification algorithm on patients' personal wearable devices. Local execution allows for continuous operation regardless of the network speed and availability. In addition, it allows data to stay on the wearable device and hence avoids privacy issues of cloud assisted processing. Continuous monitoring on wearable devices require the automated ECG classification algorithm to be both accurate and light-weight at the same time. This forms our main focus in this work. Different methods to detect and locate myocardial infarction have been widely employed. Most of the literatures were based on the procedure including the feature extraction, feature dimensionality and classification. Support vector machine (SVM) [2], Naive Bayes (NB) [3], and other machine learning (ML) techniques have been widely adopted as the classification methods.

To extract the features automatically and increase the heartbeat classification accuracy, deep-learning based algorithms have recently been proposed [4]–[7]. Deep learning-based algorithms, including deep Convolutional Neural Networks (CNNs) [8][9][10] and Recurrent Neural Networks (RNNs) [11][12][13], have recently been proposed to increase the heart signal classification accuracy. One of the most popular and effective RNN models used for sequence learning is called Long Short-Term Memory (LSTM) [14]. LSTM is designed to model the long-range dependencies and has achieved remarkable success in various applications on sequence data because of its memory and gate mechanism. However, high accuracy comes at the cost of high computation, and more storage and memory bandwidth requirements. These requirements make deployment of high accurate networks challenging, especially for resource-limited platforms such as portable devices. Compared to feed-forward Neural Networks (such as CNNs), RNN deployment has more challenges as it requires keeping the state between processing steps. As a result, multi-core general-purpose computing platforms may be inefficient for implementing RNN algorithms. This paper focuses on the ECG classification algorithm based on the LSTM networks.

To accelerate the inference of LSTMs on resource-limited wearable devices, many techniques have been presented in the literature. [15] and [16] propose techniques for pruning weight to reduce the computation. Quantization techniques as one of the most popular and efficient techniques are deployed to reduce not only the massive amount of computations but also memory storage and access in neural networks [17][18]. Binarization is the most extreme approach of the quantization model that constrains weights and/or inputs to +1 or -1 values represented by a single 1 or 0 bit, respectively. Binarized neural network (BNN) eliminates most of the multiply-accumulate (MAC) operations, and hence computations are remarkably reduced [19][20]. Therefore, they are extremely suitable for real-time and resource-limited embedded and wearable devices. However, deploying the conventional BNN model to some kind of datasets such as ECG leads to remarkable accuracy loss. In this work, our effort is to cope with the accuracy loss while reducing computation complexity and memory storage. To the best of our knowledge, our proposed method is the first effort to fully binarize LSTM for ECG classification with remarkable yield in complexity reduction. Our focus is on multi-level binarization of LSTM cells in each time step, for all weights, inputs, internal parameters, and outputs of the activation functions.

In this paper, we propose ELC-ECG, a compute optimized LSTM cell for ECG signal classification along with a novel hardware architecture for LSTMs, as an efficient solution to address the problem of high-performance LSTM deployment in resource-limited platforms. Based on the binarization of all parameters, we propose an XNOR based multiplier for performing matrix and point-wise multiplications. Our proposed method significantly reduces computations and delays yet with a negligible accuracy loss compared to the conventional LSTM cell for ECG classification. At the hardware level, our system consists of an efficient hardware

architecture that exploits the inherent parallelism of the LSTM and is parametrized with respect to the level of binarization. By optimizing the level of binarization, the proposed architecture generates a system tailored to the target application, the available hardware resources, and the computation time constraints. The paper is structured as follows: In Section II, we briefly overview related work. Section III presents a background on the LSTM. Our proposed method is discussed in Section IV. Section V presents the experimental results. Finally, Section VI concludes the paper.

II. Related Work

Different types of neural networks such as CNN [21], LSTM [22], RNN [23], and the combination of CNN and LSTM [24] are deployed for ECG signal classification. All of the mentioned works have attempted to design a neural network model with high accuracy on ECG signal classification that plays a crucial role in healthcare applications. However, the amount of computation required for such highly accurate neural networks is a challenge for low power and resource-limited healthcare wearable devices [25]. Quantization is one of the most common techniques that alleviates the compute and storage challenges of deep neural networks by reducing the precision of operations [26][27]. Vigno at el. proposed quantized CNNs for ECG classification to be used on mobile devices. They substitute floating-point weights with 8 bits precision, while both input and output remain floating points [28]. The most severe quantization approach is binarization that results in BNNs. XNOR-Net [20] has binarized both weights and inputs and substituted MAC operations with bitwise operations. Therefore, by eliminating expensive arithmetic operation, BNNs are appropriate for efficient hardware implementation. Wu at el. present binarized CNNs for ECG signal classification. Since accuracy drops significantly with binarization, they use the knowledge distillation technique to cope with this challenge. A full-precision model (teacher) is deployed for regularization of the training of binarized model (student) through knowledge distillation [25].

As LSTM has been developed to process the sequence of data, it is more suitable for ECG signal classification compared to CNN. Since the quantization of LSTM stands even more challenges, most of the prior works on quantization have focused on CNNs, and less attention has been paid to RNNs. Binarized LSTMs (binarizing both weights and inputs) have been recently used in natural language processing and human activity recognition[29][30]. However, binarized LSTM suffers from accuracy loss in more complicated applications such as ECG classification. Our observations on ECG classification has demonstrated that binary LSTM leads to considerable accuracy loss. In this paper, we present an efficient LSTM cell for ECG signal classification that is suitable for healthcare wearable devices. For this purpose, we reduce both computation complexity and memory storage and still cope with the accuracy loss for ECG signal classification. To the best of our knowledge, there is no work on quantizing LSTM for ECG signal classification.

III. LSTM Theory

A recurrent neural network typically processes inputs and produces an output at each time step. The network has recurrent connections from the output at a one-time step to the hidden parts at the next time step, which enables it to capture sequential patterns. The LSTM model differs from RNNs in that it comprises control units named gates, instead of layers. A typical LSTM is composed of several recurrently connected "memory cells". An LSTM cell is shown in Figure 1. Equation (1) to (6) summarize formulas of the LSTM network forward pass.

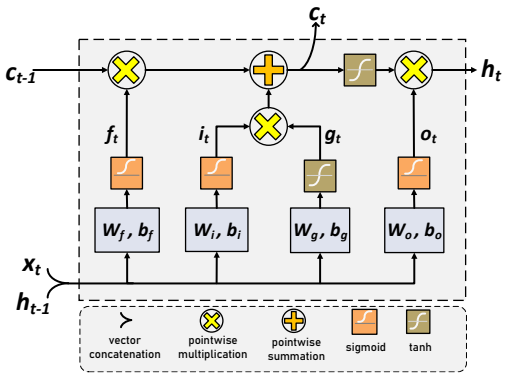


Figure 1. Long Short-Term Memory (LSTM) Cell

$$i_t = sig\left(W_{i_x} x_t + W_{i_h} h_{t-1} + b_i\right) \quad (1)$$

$$f_t = sig\left(W_{f_x} x_t + W_{f_h} h_{t-1} + b_f\right) \quad (2)$$

$$g_t = tanh\left(W_{g_x} x_t + W_{g_h} h_{t-1} + b_g\right) \quad (3)$$

$$o_t = sig\left(W_{o_x} x_t + W_{o_h} h_{t-1} + b_o\right) \quad (4)$$

$$c_t = f_t \times c_{t-1} + i_t \times g_t \quad (5)$$

$$h_t = o_t \times tanh(c_t) \quad (6)$$

The weight matrices are shown by W_x and W_h. x is the input vector, b refers to bias vectors, and t denotes the time (*t-1* refers to the previous time step). Activation functions are point-wise non-linear functions, that is *logistic sigmoid* for the gates and *hyperbolic tangent* for input to and output from the node. i_t, f_t and o_t are the input, forget and output gates respectively, c_t is the current state of the LSTM, h_{t-1} is the previous output, x_t is the current input at time t. Finally, all the W matrices denote the weight matrices that contain the trainable parameters of the model. In the full-precision LSTM, the lengths of all vectors are considered to be 32 bits.

IV. ELC-ECG: Proposed Binarization Method

In this section, we first discuss the conventional binarization technique. Then, introduce the multi-level binarized LSTM. Finally, we present the hardware accelerator for our proposed method.

A. Conventional binarization technique

In BNN, full precision inputs (i) and weights (w) values are substituted with -1 or +1 represented by a single 0 or 1 bit, respectively. Sign function as a deterministic binarization function is mostly used for binarizing parameters according to Equation (7), where *x* parameter can be input or weight [19].

$$sign(x_b) = \begin{cases} +1 \text{ (mapped to 1)} & if\ x \geq 0, \\ -1 \text{ (mapped to 0)} & otherwise \end{cases} \quad (7)$$

By deploying binary values instead of full precision ones, MAC operations are substituted with bitwise operations. Scaling factors are also employed for binarized values to reduce the accuracy loss caused by binarization especially in large datasets. Let $\vec{x} = \alpha_x \vec{s}_x$ and $\vec{w} = \alpha_w \vec{s}_w$ be inputs and weights, where $\vec{s}_x$ and $\vec{s}_w$ are sign vectors and α_x and α_w denote the scaling factors of inputs and weights, respectively. Therefore, the dot product between weights and inputs can be approximated using *XnorPopcount* operations according to Equation (8).

$$dot(\vec{w}, \vec{x}) = \alpha_x \alpha_w\ dot\ (\vec{s}_w, \vec{s}_x) = \alpha_x \alpha_w XnorPopcount\left(\vec{b}_w, \vec{b}_x\right) \quad (8)$$

Where $\{\vec{b}_x, \vec{b}_w\}$ is the binary encoding corresponding to $\{\vec{s}_x, \vec{s}_w\}$, respectively. If we consider N and p as the size of the vector and set-bits, respectively, *Popcount* operation is calculated as 2p-N.

B. *Proposed Binarization Method for LSTM*

Although binarized LSTMs decrease the number of computations, resources, and memory footprint, they suffer from significant accuracy loss in some applications like ECG classification compared to their full precision counterparts. We have proposed ELC-ECG, which uses binary weights and multi-level binarized inputs to improve accuracy. In fact, we have presented a specific binarization method based on [31] which efficiently quantizes inputs and also substitutes all weight parameters of LSTM with binary values. Moreover, to improve the accuracy, we have exploited scaling factors which are learned during the training phase. MLB Algorithm shows proposed multi-level binarization for both inputs and weights. It should be mentioned that the area overhead of multi-level binarization is negligible.

MLB Algorithm Multi-Level Binarization

Inputs: $\alpha_1, \ldots, \alpha_M$ *(scaling factors), x (Input of MLB)*
Output: $l_1, l_2, \ldots, l_n$

1: $r = x$
2: **for** i =1 **to** M **do**
3: $l_i \leftarrow Binarize((Sign(r))$
4: $r \leftarrow r - Sign(r) \times \alpha_i$
5: **end for**

This procedure is repeated for *M* times, where *M* denotes the number of binarization levels for the input. Note that in M-level binarized input, each layer needs M separated scaling factors for inputs. Besides, we have considered a distinguished scaling factor for each of the weights and biases based on their values.

In this paper, we target Diagnosing Myocardial Infarction using LSTM networks on PTB Diagnostic ECG Database[32]. We have used the LSTM network presented in [33][34] as a basic implementation, then we have binarized the LSTM model. As the ECG data of each subject is divided into smaller sequences, we first need to predict a label for each sequence/window. As soon as we have done the predictions, we can group them for each patient and make a patient-level prediction. When the average is lower than 0.5, it means that more than half of the sequences are classified as healthy, then we classify the patient as healthy. When the average is more than 0.5, the patient is diagnosed with a Myocardial infarction.

TABLE I demonstrates the effect of multi-level binarization on inputs (M=1 to M=5) on accuracies of Healthy Control and Myocardial infarct. I and W stand for bit widths of input and weight, and FP stands for 32-bit full-precision. As shown in this table, increasing *M* improves the final achievable accuracy.

TABLE I. The obtained accuracies by multi-level binarization for Input

(I,W)	(1,1)	(2,1)	(3,1)	(4,1)	(5,1)	(FP,FP)
Control	43.25	68.2	69.34	74.09	75.16	75.69
Infarct	89.12	90	91.52	93.61	94.09	94.57

As shown in TABLE I the network achieves pretty close accuracy to full precision ones with (5,1). Therefore, we consider only 1-bit for weights and multi-level binarization for inputs. Also, to eliminate point-wise multiplications, we have considered the output of the activation functions to be one bit.

In our method, we can still use the *XnorPopcount* instead of the multiplication. In the proposed method, the dot product between an M-level residual-binarized input vector and a binary weight vector is performed using M subsequent *XnorPopcount* operations. Let $\vec{x} = \sum_{i=1}^{M} \alpha_{x_i} \vec{s}_{x_i}$ and $\vec{w} = \alpha_w \vec{s}_w$ where α_{x_i} denotes the i[th] scaling factors of inputs, $\vec{s}_x$ and $\vec{s}_w$ are sign vectors and α_w denotes the scaling factors of weights. The dot product of multi-level input and 1-bit weight is shown in Equation 9.

$$dot(\vec{x}, \vec{w}) = dot\left(\sum_{i=1}^{M} \alpha_{x_i} \vec{s}_{x_i}, \alpha_w \vec{s}_w\right) = \sum_{i=1}^{M} \alpha_{x_i} \alpha_w dot(\vec{s}_{x_i}, \vec{s}_w) = \sum_{i=1}^{M} \alpha_{x_i} \alpha_w XnorPopcount(\vec{b}_{x_i}, \vec{b}_w) \quad (9)$$

Since the scaling factors have been primarily obtained statically during the training phase, we can directly calculate γ instead of $\alpha_x * \alpha_w$ and store it in memory and avoid $\alpha_x * \alpha_w$ multiplication.

C. Hardware Accelerator

In this paper, all of the matrix-vector multiplications, activation functions, pointwise multiplications, and pointwise summations of one LSTM layer are performed by a unit called Processing LSTM Layer (PL). As illustrated in Figure 2(a), the PL unit is comprised of P Processing Cell (PCs) that each performs one LSTM cell operation. Two-levels of parallelism are offered by the PL unit. First, we assume *h* as the number of outputs of an LSTM layer which are fed into the PCs in *h/P* groups. The operations of all PCs run in parallel to compute multiple output cells simultaneously. Second, in each PC, same SIMD sub-operations are performed in parallel on the input vectors. For more clarification, we assume N_x as the size of the input vector in each PC unit in which the operations on S of N_x are performed simultaneously. It is also worth noting that the number of PCs (P) and SIMD lanes (S) are configurable to provide a tradeoff between throughput and area. In the LSTM cell, first, the matrix-vector multiplications must be calculated. We present Matrix-Vector-Activation (MVA) unit to perform the matrix and vector arithmetic and activation function operation demonstrated in Equation (1) to (4).

It should be noted that we use the *XnorPopcount* in the MVA unit to multiply the M-level binarized input with 1-bit weight based on Equation 9. The internal architecture of MVA

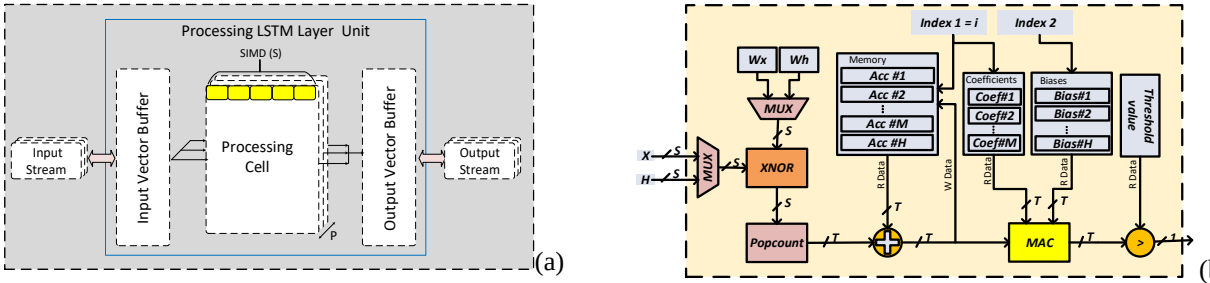


Fig 2. (a) Processing Lstm Layer (PL) architecture: "PC" Processing Cell performing "S"-width SIMD operations. (b) the internal architecture of MVA

is shown in Figure 2(b). The Multiplexers' selection lines determine the type of matrix multiplication which can be $W_x x_t$ or $W_h h_{t-1}$ and then multiplication is done by *XnorPopcount* unit. As mentioned before, in Equation (1) to (4) W, h, and b are represented with 1-level binarized values and inputs represented by x are M-level binarized values. Therefore, a simple dual-port memory with M+1 words depth has been considered to store the Popcount values in MVA unit; since each word stores the Popcount value corresponding to each M binarization level, M words are needed to perform $\sum_{i=1}^{M} XnorPopcount(\vec{b}_{x_i}, \vec{b}_{wx})$ and one word is needed to perform $XnorPopcount(\vec{b}_{\mathrm{h}}, \vec{b}_{w})$. According to Equation 9, we must apply the coefficients when all Popcount values are calculated. Since the weight and input coefficients are obtained offline during the training phase, we can calculate the γ instead of $\alpha_x * \alpha_w$ and store it in memory to avoid the multiplication. Hence, the accumulator values are multiplied by the corresponding scaling factors and are summed together by the MAC unit. Afterward, the bias values are added to the MAC results (Eq (1) to Eq (4)). Since we use the binary activation function, which approximates an input x by the sign of x, the activation function is implemented using a comparator that compares the MAC outputs with a threshold value. The next step is to update the old cell state C_{t-1}, with the new cell state C_t based on Equation (5). As mentioned before, the outputs of Equation (1) to (4) are binarized with 1-level values. In this way, we have the following formula for computing Equation (5):

$$C_t = f_t * C_{t-1} + i_t * g_t = S_{f_t}\alpha_{f_t} * S_{C_{t-1}}\alpha_{C_{t-1}} + S_{i_t}\alpha_{i_t} * S_{g_t}\alpha_{g_t} \\ = S_{f_t}S_{C_{t-1}}(\alpha_{f_t} * \alpha_{C_{t-1}}) + S_{i_t}S_{g_t}(\alpha_{i_t} * \alpha_{g_t}) \quad (10)$$

Where S and α denote the sign (-1 and +1) and scaling factor of each parameter, respectively. Based on$\{S_{f_t}, S_{C_{t-1}}, S_{i_t}, S_{g_t}\}$, there are sixteen possible values (four input variables are combined in 16 various ways) for $Equation\ (10)$. As illustrated in Figure 1, C_t is used for two purposes: 1) as C_{t-1} in the next time step 2) the input of *tanh* for computing h_t. Since C_{t-1} is represented with 1-level binarization in the next time step, we need only the sign of C_t to be used as C_{t-1} for the next time step. In addition, since the scaling factors are obtained statically during the training phase, we would store the sign of sixteen possible values in the memory as a look-up table. Therefore, three pointwise operations are replaced with LUT1, as shown in Figure 3(a). Finally, we need to calculate h_t similarly:

$$h_t = tanh(C_t) \times o_t \\ = tanh\left(S_{f_t}S_{C_{t-1}}(\alpha_{f_t} * \alpha_{C_{t-1}}) + S_{i_t}S_{g_t}(\alpha_{i_t} * \alpha_{g_t})\right) * S_{o_t}\alpha_{o_t} \quad (11)$$

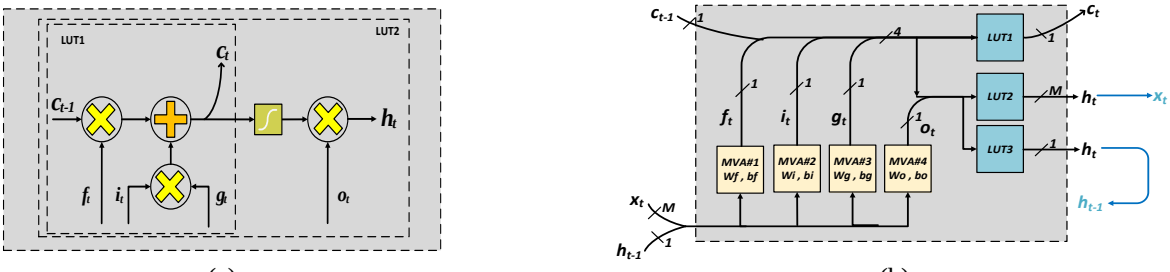
As mentioned in the prior subsections, C_t is used as *tanh* input and has only sixteen values. Thus, the *tanh* of these sixteen values can also be calculated offline. On the other hand, h_t is used as h_{t-1} in the next time step with 1-level binarization, and is used as x_t for the next layer with M-level binarization. Therefore, h_t is implemented by two look-up tables; one of them has 1-bit output, and another has M output bits, as shown in Figure 3-(b) as the overall proposed architecture for one LSTM cell.

V. Experimental Results

We have implemented a three-layer LSTM network for ECG signals classification using Keras [35] library. We have implemented our FPGA accelerator in Vivado [36] to calculate resource utilization and latency. The proposed architecture has been implemented on XC7K160 from kintex7 FPGA family. We have compared our method with the full precision LSTM in terms of accuracy, resource utilization, and inference time on the FPGA accelerator. In this paper, we have evaluated our method on ECG signal classification used in several different heart disease analysis. For this purpose, the PTB diagnostic dataset [32] from the Physionet database has been used. It contains ECG records of 290 subjects, of which 52 are normal, 148 have had a Myocardial infarction (more commonly known as a heart attack). The remaining 90 subjects suffer from other type of heart diseases.

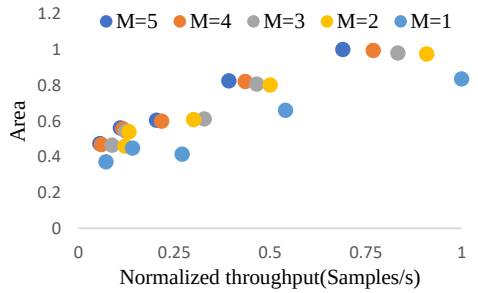
Each layer of the LSTM network has its own corresponding parallelism factors (PC-count and SIMD-width). These two parallelism factors only affect hardware performance and have no effect on the LSTM accuracy. In other words, these two factors are configurable, and provide a a tradeoff between throughput and area. Figure 4-(a) shows the area in terms of throughput for few states of these two factors (P, S) for M=1 to M=5. It should be noted that to select those two factors in each case, we have two assumptions: 1) these two factors are selected in a way that the latency of all layers are almost balanced so we can optimally use the pipeline structure. 2) Given the sampling rate in this dataset, the architecture should be able to accommodate a range of 1KHz to 10KHz. This requirement can be met by the proper selection of these two factors (P, S).

Fig 5-(a) illustrates that by increasing M to achieve better accuracy, the latency is increased and throughput is decreased as the PCs would require more accumulators. For each M from left to right of the graph, the two parallelism factors increase, which in turn decreases the delay and increases the throughput. Figure 4-(b) compares the area of our method with the full precision design in terms of throughput for different values of (P, S), considering the two mentioned assumptions. As shown, we can achieve better throughput with less area. Figure 5 compares the latency of our proposed method with the full precision design in the same area.

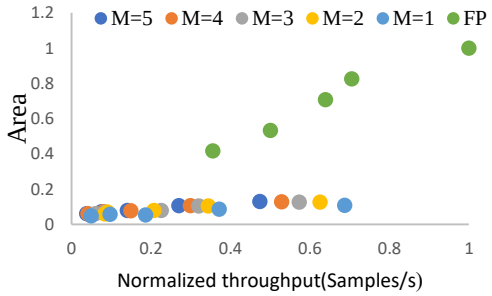


(a) (b)

Fig 3. (a) Removing point wise operation with LUT1. (b) The overall architecture for one LSTM cell.



(a)



(b)

Fig 4. (a) comparing the area and throughput of our method with M=1 to M=5. (b) comparing the area and throughput of our method with the Full Precision (FP) design for different states of (P,S). The area and throughput are normalized to maximum values. M is the number of levels for inputs.

In summary, the latency of our method grows with *M*, because the number of *XnorPopcount* operations is equal to *M* for a single dot product based on Equation (9). Compared to the full precision design, our proposed method can reduce latency up to 50x without any area overhead.

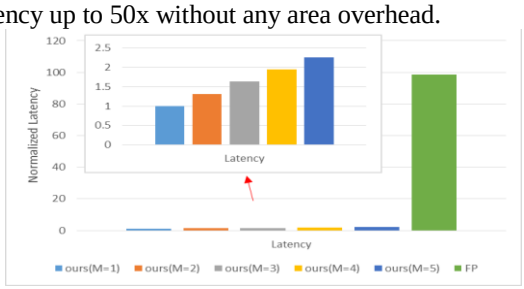


Fig 5. Normalized latency of our method compared to Full Precision (FP). The latencies are normalized to the latency of M=1.

VI. Conclusion

Binarized LSTM is an appropriate approach to realize the deployment of LSTM on embedded systems. However, conventional binarization approaches lead to significant accuracy loss especially for ECG classification. In this paper, a novel binarized LSTM (ELC-ECG) was proposed for the ECG classification that not only has remarkably decreased computations in terms of delay and area compared to full precision design but also has almost achieved the accuracy close to full precision. Moreover, we can achieve the same throughput in a much smaller area than the full precision. Finally, while our accuracy is quite close to full precision, our method reduces latency by 50x without any area overhead.

References

[1] J. M. Bote, J. Recas, F. Rincn, D. Atienza, and R. Hermida, "A modular low-complexity ecg delineation algorithm for real-time embedded systems," IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 2, pp. 429–441, March 2018.

[2] A. K. Dohare, V. Kumar, & R. Kumar, "Detection of myocardial infarction in 12 lead ECG using support vector machine". *Applied Soft Computing, 64*, 138-147, 2018.

[3] N. Priyanka and P. RaviKumar, "Usage of data mining techniques in predicting the heart diseases—Naïve Bayes & decision tree," In *International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pp. 1-7, 2017.

[4] O. S. Lih, V. Jahmunah, T. R. San, E. J. Ciaccio, T. Yamakawa, M. Tanabe, M., ... & U. R. Acharya, "Comprehensive electrocardiographic diagnosis based on deep learning", *Artificial Intelligence in Medicine*, 103, 101789, 2020.

[5] C. Han, & L. Shi, "ML–ResNet: A novel network to detect and locate myocardial infarction using 12 leads ECG", *Computer methods and programs in biomedicine*, 185, 105138, 2020.

[6] A. Darmawahyuni, S. Nurmaini, W. Caesarendra, V. Bhayyu, & M. N. Rachmatullah, "Deep Learning with a Recurrent Network Structure in the Sequence Modeling of Imbalanced Data for ECG-Rhythm Classifier", *Algorithms*, 12(6), 118, 2019.

[7] R. K. Tripathy, A. Bhattacharyya, & R. B. Pachori, "A novel approach for detection of myocardial infarction from ECG signals of multiple electrodes, *In IEEE Sensors Journal*, 19(12), 4509-4517, 2019.

[8] U. B. Baloglu, M. Talo, O. Yildirim, R. San Tan, & U. R. Acharya, "Classification of myocardial infarction with multi-lead ECG signals and deep CNN", *Pattern Recognition Letters*, 122, 23-30, 2019.

[9] H. Makimoto, M. Höckmann, T. Lin, D. Glöckner, S. Gerguri, L. Clasen,... & S. Angendohr, "Performance of a convolutional neural network derived from an ecG database in recognizing myocardial infarction," *Scientific Reports*, 10(1), 1-9, 2020.

[10] P. Natesan & E. Gothai, "Classification of Multi-Lead ECG Signals to Predict Myocardial Infarction Using CNN," *In International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1029-1033, 2020.

[11] A. Darmawahyuni and S. Nurmaini, "Deep Learning with Long Short-Term Memory for Enhancement Myocardial Infarction Classification", In *6th International Conference on Instrumentation, Control, and Automation (ICA)*, pp. 19-23, 2019

[12] E. Prabhakararao, and S. Dandapat, "Myocardial infarction severity stages classification from ecg signals using attentional recurrent neural network", *IEEE Sensors Journal*, *20*(15), 8711-8720, 2020.

[13] F. Karisik, and M. Baumert, "A Long Short-Term Memory Network to Classify Myocardial Infarction Using Vectorcardiographic Ventricular Depolarization and Repolarization", In *Computing in Cardiology (CinC)*, 2019.

[14] K. Greff, R. K. Srivastava, R. K., Koutník, B. R. Steunebrink, & J. Schmidhuber, "LSTM: A search space odyssey", *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222-2232, 2017.

[15] Cao, Shijie, et al. "Efficient and effective sparse LSTM on FPGA with Bank-Balanced Sparsity." In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 63-72. ACM, 2019.

[16] Park, Junki, et al. "Balancing Computation Loads and Optimizing Input Vector Loading in LSTM Accelerators." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019).

[17] Li, Jiajun, et al. "Simulate-the-hardware: training accurate binarized neural networks for low-precision neural accelerators." In *Proceedings of the 24th ASP-DAC*, pp. 323-328. ACM, 2019.

[18] Wang, Meiqi, et al. "E-LSTM: An Efficient Hardware Architecture for Long Short-Term Memory." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* (2019).

[19] I. Hubara, et al. "Binarized neural networks," in *NIPS*, 2016, pp. 4107–4115.

[20] M. Rastegari, et al. "Xnor-net: Imagenet classification using binary convolutional neural networks," in *ECCV*. Springer, 2016, pp. 525– 542.

[21] Xiong, Zhaohan, Martin K. Stiles, and Jichao Zhao. "Robust ECG signal classification for detection of atrial fibrillation using a novel neural network." In *2017 Computing in Cardiology (CinC)*, pp. 1-4. IEEE, 2017.

[22] Mostayed, Ahmed, Junye Luo, Xingliang Shu, and William Wee. "Classification of 12-lead ECG signals with Bi-directional LSTM network." *arXiv preprint arXiv:1811.02090* (2018).

[23] Xie, Pengwei, Guijin Wang, Chenshuang Zhang, Ming Chen, Huazhong Yang, Tingting Lv, Zhenhua Sang, and Ping Zhang. "Bidirectional recurrent neural network and convolutional neural network (BiRCNN) for ECG beat classification." In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2555-2558. IEEE, 2018.

[24] Oh, Shu Lih, Eddie YK Ng, Ru San Tan, and U. Rajendra Acharya. "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats." *Computers in biology and medicine* 102 (2018): 278-287.

[25] Wu, Qing, Yangfan Sun, Hui Yan, and Xundong Wu. "Ecg signal classification with binarized convolutional neural network." *Computers in Biology and Medicine* (2020): 103800.

[26] I. Hubara, et al. "Quantized neural networks: Training neural networks with low precision weights and activations," *arXiv preprint arXiv:1609.07061*, 2016.

[27] Zhou, Shuchang, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients." *arXiv preprint arXiv:1606.06160* (2016).

[28] Moura, Vigno, Vilson Almeida, Domingos Bruno Sousa Santos, Nator Costa, Luciano Lopes Sousa, and Pedro Cunha Pimentel. "Mobile Device ECG Classification using quantized Neural Networks." (2020).

[29] Liu, Xuan, et al. "Binarized lstm language model." In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pp. 2113-2121. 2018.

[30] Edel, Marcus, and Enrico Köppe. "Binarized-blstm-rnn based human activity recognition." In *IPIN*, pp. 1-7. IEEE, 2016.

[31] Ghasemzadeh, M, et al. ReBNet: Residual binarized neural network. In *FCCM*, pp. 57-64, 2018

[32] R. Bousseljot, D. Kreiseler, A. Schnabel, A. Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet. Biomedizinische Technik, Band 40, Ergänzungsband 1 (1995) S 317

[33] https://www.orikami.nl/

[34] https://github.com/Lucnies/ecg-classification

[35] F. Chollet et al., “Keras.” https://github.com/fchollet/keras, 2015.

[36] Xilinx, “Vivado.” https://www.xilinx.com/products/design-tools/vivado.html, 2017.