Energy-Efficient LSTM Inference Accelerator for Real-Time Causal Prediction

ZHE CHEN, Computer Science Department, UCLA
HUGH T. BLAIR, Department of Psychology, UCLA
JASON CONG, Computer Science Department, UCLA

Ever-growing edge applications often require short processing latency and high energy efficiency to meet strict timing and power budget. In this work, we propose that the compact long short-term memory (LSTM) model can approximate conventional *acausal* algorithms with reduced latency and improved efficiency for real-time causal prediction, especially for the neural signal processing in closed-loop feedback applications. We design an LSTM inference accelerator by taking advantage of the fine-grained parallelism and pipelined feedforward and recurrent updates. We also propose a bit-sparse quantization method that can reduce the circuit area and power consumption by replacing the multipliers with the bit-shift operators. We explore different combinations of pruning and quantization methods for energy-efficient LSTM inference on datasets collected from the electroencephalogram (EEG) and calcium image processing applications. Evaluation results show that our proposed LSTM inference accelerator can achieve 1.19 GOPS/mW energy efficiency. The LSTM accelerator with 2-sbit/16-bit sparse quantization and 60% sparsity can reduce the circuit area and power consumption by 54.1% and 56.3%, respectively, compared with a 16-bit baseline implementation.

CCS Concepts: • **Hardware → Logic circuits**; • **Computing methodologies → Neural networks;**

Additional Key Words and Phrases: Calcium imaging, EEG, energy efficiency, long short-term memory (LSTM), quantization

ACM Reference format:
Zhe Chen, Hugh T. Blair, and Jason Cong. 2022. Energy-Efficient LSTM Inference Accelerator for Real-Time Causal Prediction. *ACM Trans. Des. Autom. Electron. Syst.* 27, 5, Article 44 (June 2022), 19 pages.
https://doi.org/10.1145/3495006

1 INTRODUCTION

Near-sensor edge computing often requires short processing latency and high energy efficiency, which are important for a broad range of emerging applications, including brain-machine interfaces, autonomous driving, robotics and drones, and more, especially when the closed-loop feedback control is in demand. In this article, we introduce our approaches that can achieve short-latency and energy-efficient neural signal processing for implantable electrophysiology and calcium-imaging sensors.

This work is supported by the National Science Foundation under Grant No. DBI-1707408.
Authors' addresses: Z. Chen, Computer Science Department, UCLA, Engineering VI, 405 Hilgard Avenue, Los Angeles, CA, 90095, USA; email: zhechen@ucla.edu; H. T. Blair, Department of Psychology, UCLA, Pritzker Hall, 405 Hilgard Avenue, Los Angeles, CA, 90095, USA; email: tadblair@ucla.edu; J. Cong, Computer Science Department, UCLA, Engineering VI, 405 Hilgard Avenue, Los Angeles, CA, 90095, USA; email: cong@cs.ucla.edu.

1084-4309/2022/06-ART44 $15.00
https://doi.org/10.1145/3495006

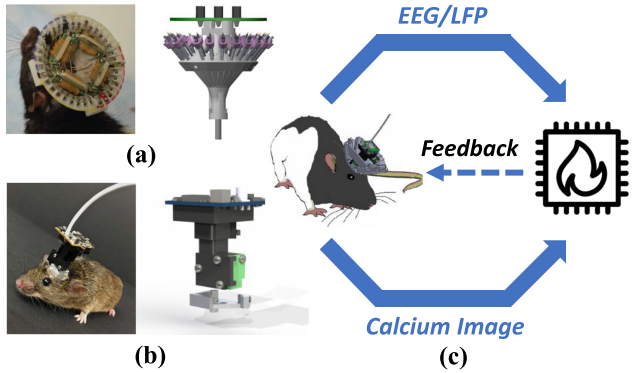

Fig. 1. (a) 128-channel electrophysiological signal-recording sensor [29]. (b) Miniaturized calcium imaging device [1]. (c) Closed-loop neurofeedback control based on real-time EEG and calcium image processing.

Recent advancements in neural recording technology have unprecedentedly enabled monitoring of cell activity from hundreds to thousands of neurons *in vivo* from rats or mice [1, 11]. Figure 1(a) shows an assembly of tetrodes [29] that can sample 128 channels of electroencephalogram (EEG) signals at a 32-kHz rate. Figure 1(b) presents a miniaturized calcium-imaging device (Miniscope) [1] that can monitor dynamic fluorescence changes coupled with cell firings from hundreds of neurons in real time. Extensive neuroscientific research has taken place by leveraging these two types of neural recording techniques. However, most of the existing works rely on offline data analysis, and the closed-loop feedback for the EEG and/or the calcium imaging as illustrated in Figure 1(c) remains a challenge.

There are two main obstacles preventing a computing device from working effectively in a closed loop. First, the EEG and calcium-image processing require short and deterministic latency for the closed-loop feedback, especially considering the spike timing precision of 1 ms [24]. Second, the processing demands high energy efficiency, because the device needs to be worn or surgically implanted into the experimental subject and it must support long battery life when frequent charging is impossible. Moreover, the heat generated by the processor should not cause a temperature increase surpassing $2^{\circ}C$, which is a limit for general neural implants [25].

Many neural signal processing algorithms have been proposed. However, they mainly target offline data analysis. For the EEG signal processing, the conventional flow is based on finite impulse response (FIR) or infinite impulse response (IIR) digital filters. Such FIR/IIR filters have an *acausal* nature; thus, it will cause an undesirable delay when deployed for real-time EEG phase detection. For calcium-image processing, several offline analysis pipelines have been proposed [12, 13, 28, 38]. The most popular one is based on the iterative constrained nonnegative matrix factorization (CNMF) method [38], which extracts cell contours and traces simultaneously. However, since it requires batch processing, it is hard to achieve a short latency for closed-loop feedback applications.

In this work, we take inspiration from the recurrent nature of the brain, and come up with a solution based on the compact long short-term memory (LSTM) model. We first introduce the algorithm and the evaluation. Then, we design a customized accelerator for it. We consider both the quantization and the pruning methods that can further improve the energy efficiency of the accelerator while not degrading the accuracy.

This article is an extension of our previous works [5–7]. We propose a combination of the bit-sparse quantization and pruning methods for the energy-efficient LSTM inference, and evaluate the accuracy across the EEG and the calcium-imaging datasets. We also compare the energy and

area efficiency of the LSTM accelerator under various pruning and quantization optimizations. We summarize the contributions of this article as follows.

- We propose that the compact LSTM inference can approximate conventional acausal algorithms for both EEG and calcium-image processing and largely reduce the latency and improve the efficiency.
- We design an LSTM inference accelerator that takes advantage of the fine-grained parallelism and pipelined feedforward and recurrent updates inherent in the compact LSTM model.
- We combine bit-sparse quantization with pruning to reduce the circuit area and improve the energy efficiency of the LSTM accelerator by 54.1% and 56.3%, respectively, while not degrading the accuracy.

This article is organized as follows. Section 2 presents the background on neural signal processing. Sections 3 and 4 introduce the LSTM-based methods and the corresponding accelerator design, respectively. Section 5 discusses the hardware-friendly quantization and pruning methods. Section 6 shows the evaluation and experimental results. Section 7 summarizes related works and Section 8 draws conclusions.

2 RESEARCH BACKGROUND

2.1 EEG Signal Processing

EEG is a widely used technique for sensing the electrical activity generated by the brain at submillisecond time resolution, either invasively from electrodes implanted in the brain or noninvasively from scalp electrodes. In humans and other mammals, the brain generates rhythmic oscillations that can be detected in the EEG across a range of different frequencies. Major EEG frequency bands include Delta rhythm (δ; <4 Hz), Theta rhythm (θ; 4–10 Hz), Alpha rhythm (α; 10–15 Hz), Beta rhythm (β; 15–30 Hz) and Gamma rhythm (γ; >30 Hz). In addition to these oscillatory rhythms, transient EEG signals such as K-complexes, P-300 waves, and sharp-wave ripple (SWR) events are also known to occur in specific brain regions [2]. Research has shown that certain brain rhythms and transient EEG events can be used as biomarkers of abnormal brain activity, such as seizures or anxiety attacks [2]. We focused on a processing flow that can be generally used to detect such EEG rhythms or transient signals and deliver closed-loop stimulation that is precisely synchronized to the detected signals.

A typical closed-loop neurofeedback stimulation case with conventional EEG signal processing flow is illustrated in Figure 2. EEG signals are typically sampled at a high sampling rate, such as 32 kHz. To analyze a specific frequency band, raw EEG data are often downsampled to the lowest allowable rate correspondingly. For example, to analyze the Theta rhythm, the raw EEG data need to be downsampled to 160 Hz, which is about ten times the upper cutoff frequency of the bandpass. A bandpass filter is then applied to this downsampled signal, and a standard method for extracting the phase information to synchronize the neurofeedback stimulation involves taking the Hilbert transform, which shifts the phase of the bandpass filtered signal by 90° [22].

The bandpass and Hilbert transform filters can be implemented by FIR or IIR digital filters. However, these filters are acausal and can incur unacceptably long delay; thus, they are more useful for offline analysis. In real-time applications, FIR or IIR implementations can impose an unwanted delay longer than 10 ms in a neurofeedback closed loop, which can cause the stimulation to not work properly.

The last step in the EEG signal processing flow is the phase and envelope calculation. We denote the real and imaginary components of the analytic signal as the u_r and the u_i, which are computed

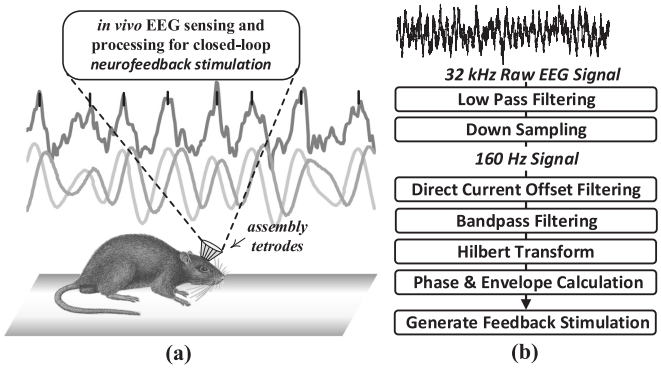

Fig. 2. (a) EEG neurofeedback application on a freely behaving rat and (b) a conventional flow for EEG signal processing for neurofeedback stimulation.

from the bandpass filter and Hilbert transform, respectively. Finally, the phase and envelope are calculated by the following equations:

$$\begin{aligned} \phi(n) &= \arctan(u_i(n)/u_r(n)) \\ u_A(n) &= \sqrt{u_r^2(n) + u_i^2(n)} \end{aligned} \tag{1}$$

where $\phi(n)$ and $u_A(n)$ represent the calculated phase and envelope for the timestep n, respectively. The phase and envelope can be used to detect specified events in the EEG, and the event detection can be used to trigger closed-loop neurofeedback stimulation to accomplish therapies or research tasks.

2.2 Calcium-Image Processing

In vivo calcium imaging is an emerging technique for monitoring activity from large populations of neurons in the brain of a freely behaving animal, such as a mouse or rat. One common approach for using this technique is to mount a miniaturized fluorescence microscope, for example, the Miniscope, onto the animal's head to obtain video recordings of calcium activity in the brain. In such recordings, a punctate flash of fluorescence is observed at a specific location in the video when a particular neuron becomes active. The Miniscope can record the activity from hundreds of neurons simultaneously over weeks or months as the animal engages in various behavioral tasks.

Several offline calcium-image analysis pipelines are currently in wide use by research labs, including the CNMF and the MIN1PIPE [12, 13, 28, 38]. These algorithms share similar analysis flows. As Figure 3(a) shows, the first step is motion correction, which is necessary because as the brain moves around inside the skull of a freely behaving animal, it produces motion artifacts that must be canceled out to stabilize the images for further analysis. Second, the motion-corrected image is enhanced for a higher signal-to-noise ratio (SNR), and locations of individual neurons are identified from the enhanced image, as illustrated in Figure 3(b). Third, calcium fluorescence traces need to be disentangled and extracted from the individual neurons, as shown in Figure 3(c). Hundreds or even thousands of different neurons can reside within the Miniscope's field of view. Thus, extracting traces for each neuron can be a computationally intensive task.

2.3 LSTM Model

LSTM [21] is a type of recurrent neural network (RNN) that has been successfully used for many temporal signal prediction tasks, such as handwriting recognition [15] and speech recognition [16].

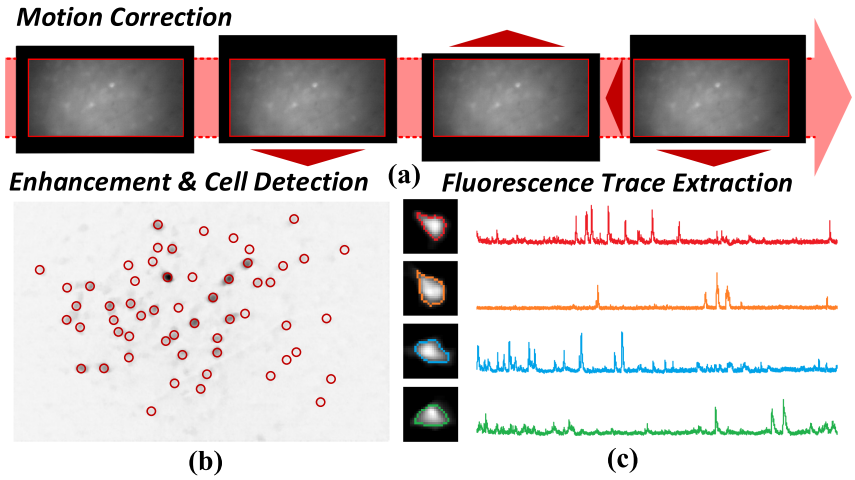

Fig. 3. Conventional calcium image analysis flow includes (a) motion correction, (b) image enhancement and cell detection and (c) fluorescence trace extraction.

A typical LSTM network architecture is made up of a hidden layer and an output layer, as Figure 4 shows. The hidden layer consists of four types of gates: the input gate $\boldsymbol{I}$, the forget gate $\boldsymbol{F}$, the cell gate $\boldsymbol{G}$, and the output gate $\boldsymbol{O}$; and two types of nodes: the cell node $\boldsymbol{C}$ and the hidden node $\boldsymbol{H}$. For each time step n, the LSTM takes the input $In(n)$, updates the values of all of the gates and nodes, and then generates the output $Out(n)$ based on the value of hidden nodes. The LSTM inference is defined by:

$$\begin{aligned}
\boldsymbol{I}(n) &= \sigma\left(In(n)\boldsymbol{W}_{Ii} + \boldsymbol{H}(n-1)\boldsymbol{W}_{IH} + \boldsymbol{B}_I\right)\\
\boldsymbol{F}(n) &= \sigma\left(In(n)\boldsymbol{W}_{Fi} + \boldsymbol{H}(n-1)\boldsymbol{W}_{FH} + \boldsymbol{B}_F\right)\\
\boldsymbol{G}(n) &= \tanh\left(In(n)\boldsymbol{W}_{Gi} + \boldsymbol{H}(n-1)\boldsymbol{W}_{GH} + \boldsymbol{B}_G\right)\\
\boldsymbol{O}(n) &= \sigma\left(In(n)\boldsymbol{W}_{Oi} + \boldsymbol{H}(n-1)\boldsymbol{W}_{OH} + \boldsymbol{B}_O\right)
\end{aligned} \tag{2}$$

$$\begin{aligned}
\boldsymbol{C}(n) &= \boldsymbol{F}(n) \odot \boldsymbol{C}(n-1) + \boldsymbol{G}(n) \odot \boldsymbol{I}(n)\\
\boldsymbol{H}(n) &= \boldsymbol{O}(n) \odot \tanh\left(\boldsymbol{C}(n)\right)\\
Out(n) &= \boldsymbol{H}(n)\boldsymbol{W}_{OUT} + B_{OUT}
\end{aligned} \tag{3}$$

It describes the update of the four types of gates based on the input from the current timestep n and the value of hidden nodes from the previous timestep $n-1$, and the update of all cell and hidden nodes based on the updated value of gates and the value of cell nodes from the previous timestep $n-1$ through the recurrent connection, as shown in Figure 4. The output $Out(n)$ is computed from the updated value of the hidden nodes. In Equations (2) and (3), σ and $tanh$ are non-linear *sigmoid* and *hyperbolic tangent* functions, and $\odot$ represents the dot product operation. Supposing that a simple LSTM model with one single layer has N_H hidden nodes, the input weights $\{\boldsymbol{W}_{Ii}, \boldsymbol{W}_{Fi}, \boldsymbol{W}_{Gi}, \boldsymbol{W}_{Oi}\}$ and the hidden layer biases $\{\boldsymbol{B}_I, \boldsymbol{B}_F, \boldsymbol{B}_G, \boldsymbol{B}_O\}$ are N_H-dimensional vectors, and the internal weights $\{\boldsymbol{W}_{IH}, \boldsymbol{W}_{FH}, \boldsymbol{W}_{GH}, \boldsymbol{W}_{OH}\}$ are $N_H \times N_H$ matrices. The output weight $\boldsymbol{W}_{OUT}$ is an N_H-dimensional vector and the output layer bias B_{OUT} is a scalar value. The computation complexity measured in the number of operations for an N_H-node LSTM model is

$$C(N_H) = 8N_H^2 + 14N_H. \tag{4}$$

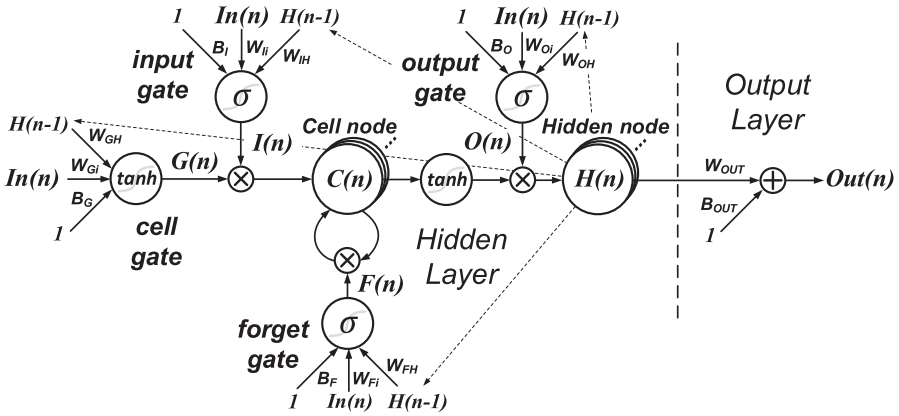

Fig. 4. Typical LSTM network architecture with single hidden layer.

The number of operations on the feedforward path is

$$C_{forward}(N_H) = 8N_H^2 + 8N_H. \tag{5}$$

Considering that $N_H = 5$, the operations executed on the feedforward path occupy 89% of the total operations.

3 LSTM-BASED METHOD

As both the EEG and the calcium images have rich temporal features, they provide good opportunities for using the LSTM to approximate *acausal* offline algorithms for efficient real-time causal prediction. In this section, we introduce three case studies demonstrating the LSTM advantage in real-time neural signal processing. Section 3.1 describes its usage for EEG phase detection; Sections 3.2 and 3.3 show its application in calcium-image motion correction and trace extraction.

3.1 LSTM-Based EEG Signal Processing

We proposed using LSTM inference to reduce both computation cost and processing latency for the EEG signal processing. As shown in Figure 5, we trained a pair of LSTMs to generate predictions of the u_R and u_I described in Section 2.1. Each LSTM network has one single hidden layer, which contains 5 nodes. During offline training, both LSTMs take the same downsampled and direct current offset (DCO) filtered EEG signal as the training input, and they take the bandpass-filtered signal u_R and the phase-shifted signal u_I as the training targets. Once the training is accomplished, the well-trained LSTMs are used to generate predictions on the test input data and approximate to the learned bandpass filter and Hilbert transform functions. In the final step, the phase and envelope are computed based on the inference results and used for event detection.

After the training converges, we used the inference results on the test set to compute the phase and the envelope according to Equation (1). The mean phase error is within ±3° of zero after calibration, which is accepted as an accurate result for real-time neurofeedback stimulation. The main benefit of the method is that the LSTM inference largely reduces processing latency by getting rid of the acausal delay caused by conventional FIR/IIR digital filters. Table 1 shows the difference in processing latency between a 5-node LSTM and a 5-tap IIR filter, which suffers from a long acausal delay considering a downsampled rate of 200. As Figure 5 shows, the LSTM inference bypasses the low pass filter, which operates at a high sampling rate and contributes to the main portion of the

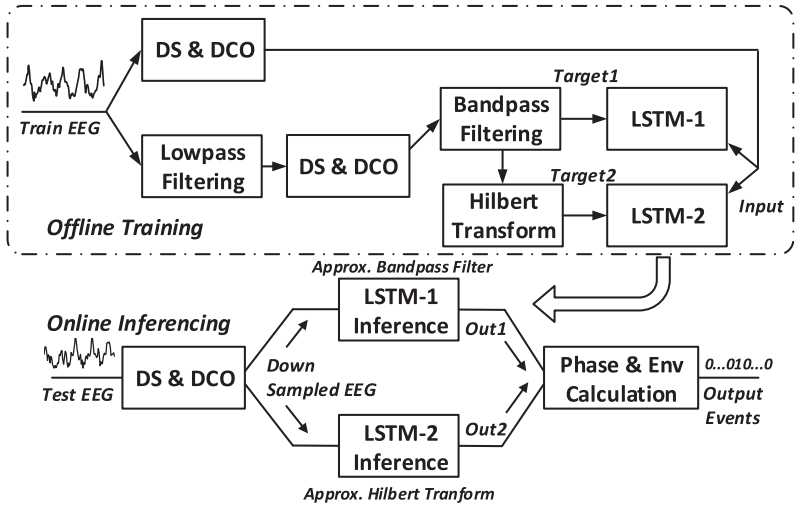

Fig. 5. EEG signal processing flow based on compact LSTM inference [7].

Table 1. Comparison of the EEG Signal Processing Methods

Methods	LSTM [7]	IIR
Implementation	5-node	5-tap
Latency@100 MHz	2.09 μs	12.5 ms
Overall # op/sample	860	4,094

computation. Considering the filtering for the Theta rhythm, the LSTM inference saves 79% of operations while maintaining acceptable accuracy.

3.2 LSTM-Based Non-rigid Motion Correction

Motion correction is a critical processing step in a variety of calcium-image analysis algorithms [13, 28, 30]. Recent work shows that piecewise rigid motion correction can effectively reduce non-rigid motion artifacts and outperforms other methods for calcium-image stabilization [30]. Piecewise rigid motion correction divides the field of view of the image into overlapping patches and performs rigid motion correction for each patch.

We propose an LSTM-based non-rigid motion correction algorithm to improve efficiency. As Figure 6(a) shows, instead of performing heavy motion calculation for each image patch, we evaluate motion only at a central region. Then, we use the calculation result to predict non-rigid motions of all image patches based on the LSTM inference. Figure 6(b) shows the motion extraction from the central region and all image patches throughout the calcium-image video session. The displacement of each patch is represented by a motion vector containing two values, which represent the rigid motion against the template along the horizontal and vertical axes with sub-pixel precision.

For the motion extraction at the central region, a fixed-point rigid motion correction is performed to reduce the computation cost. To estimate motion vectors at all image patches, a non-rigid motion correction algorithm NoRMCorre [30] is used. Figure 6(c) illustrates the LSTM-assisted method. During the offline training, the motion vector time series extracted at the central region is used as input, and the motion vector time series extracted at a specific patch $f(i_0, j_0)$ is used as

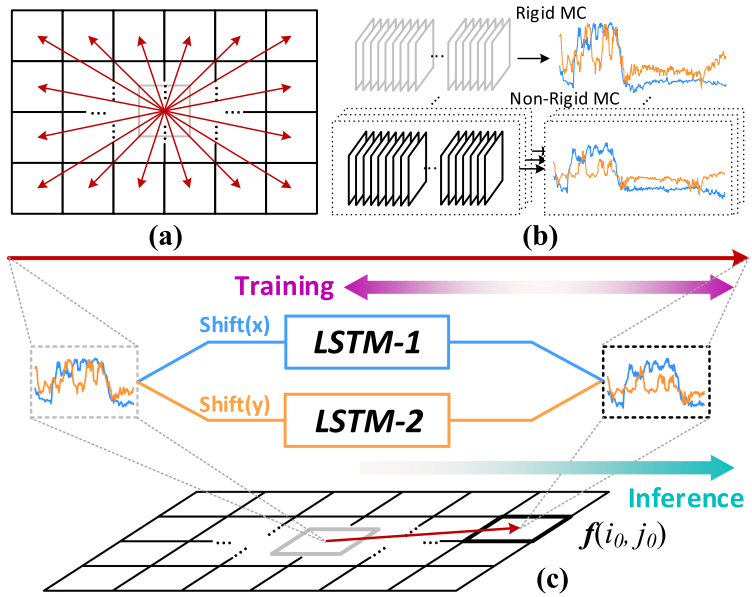

Fig. 6. (a) Proposed LSTM-assisted non-rigid motion correction. (b) Motion vector extraction for the central region and all image patches. (c) LSTM inference on one image patch.

Table 2. Performance of the LSTM Method Compared with Conventional Motion Correction Algorithms

	ROF(pixel)	$diff(x)$	$diff(y)$	$diff(a)$
NoRMCorre [30]	0.19 ± 0.11	0	0	0
Rigid method	0.60 ± 0.45	0.879	0.658	1.248
LSTM [6]	0.31 ± 0.22	0.621	0.362	0.804

the training target. For one specific mouse or rat, a pair of compact LSTM networks is trained to adapt the motion vector components of the central region to those of the patch $f(i_0, j_0)$ throughout a training video session along the horizontal and vertical axes, respectively. After the LSTM pairs are well trained, they are deployed for online inference to approximate the performance of the computationally intensive offline algorithm with much shortened latency and reduced cost.

Table 2 summarizes the evaluated residual of optical flow (ROF) [30] for non-rigid, rigid, and LSTM-based methods on a 1,000-frame calcium imaging test session. Evaluation results show that the LSTM-based method on average reduces 48% of ROF compared with the rigid method, and the accuracy performance is comparable with the NoRMCorre. We also compared the horizontal component mean absolute difference (MAD) $diff(x)$, vertical component MAD $diff(y)$, and amplitude MAD $diff(a)$ between the rigid, proposed methods and the NoRMCorre, respectively. Comparison results in Table 2 again show that the LSTM-based method achieves a higher accuracy than the rigid method in approximating the non-rigid motion correction.

3.3 LSTM-Based Trace Extraction

We also propose using LSTM inference for efficient online trace extraction from calcium images, as shown in Figure 7. The processing flow consists of the motion correction, image enhancement, fluorescence tracing, and LSTM inference steps. We apply template-based rigid motion correction

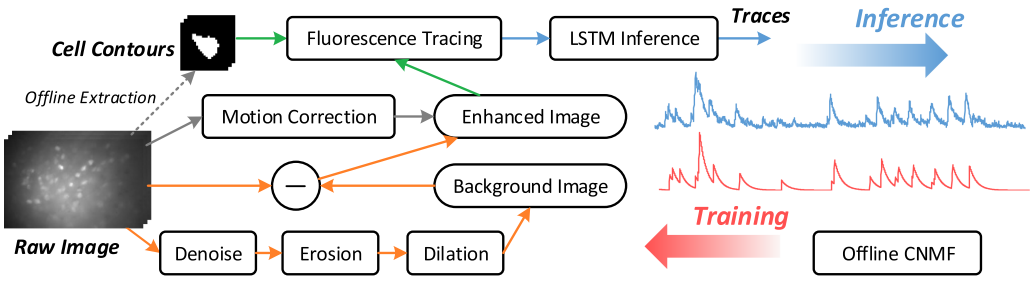

Fig. 7. Proposed LSTM-based calcium-image processing flow [5].

and perform image enhancement by subtracting the background image estimated by the morphological opening operation [28].

Fluorescence tracing relies on the cell contours extracted offline by the CNMF algorithm. The cell contours are applied on the motion-corrected and enhanced images to extract the fluorescence traces. Although this naïve method is efficient in extracting traces, the drawback is a drop in its accuracy when the SNR is low.

We use the LSTM inference to improve trace extraction accuracy for the contour-based method. During offline training, online traces are used as input and traces extracted from the CNMF algorithm are used as the training target. We carried out a separate LSTM training for each cell. After the LSTMs are well trained, they are deployed for online inference and run sequentially to extract traces from cells. Here, LSTM inference is used to approximate the outcome of the accurate but time-consuming offline algorithm, and the causality of the LSTM-based method helps reduce processing latency.

We adopted cross correlation as a measurement to evaluate the similarity between the online and offline traces quantitatively. Evaluation results show that 90.3% of cells have shown an increase in cross correlation after employing LSTM inference, which indicates improvement in trace extraction accuracy.

4 LSTM ACCELERATOR DESIGN

In order to realize an efficient implementation of the LSTM inference method, we designed a customized accelerator, shown in Figure 8. The architecture consists of an array of LSTM inference processing elements (PEs) and a controller. The PE consists of matrix-vector multiplication (MVM) and recurrent state update (REC) modules, weight buffers for the MVM and the REC modules, and the controller. The non-linear *sigmoid* and *tanh* operations are realized by a piecewise linear approximation [14]. The MVM weight buffer stores input and hidden layer weights, whereas the REC weight buffer stores output weights. The MVM and the REC modules operate in pipelines under the controller.

4.1 MVM Design

Figure 9(a) shows the circuit design of the MVM. According to Equation (2), the matrix consists of weights of the LSTM input and hidden layers, and the vector corresponds to the input and hidden states. The matrix has a dimension of $(4 \times N_H) \times (N_H + 1)$, whereas the vector has $(N_H + 1)$ elements. The MVM is composed of a 1-D multiplier bank and an adder tree. In each clock cycle, the MVM receives one 16-bit input sample, fetches N_H 16-bit weights from the MVM weight buffer and performs $N_H + 1$ multiplication operations, where N_H is the number of hidden nodes. The multiplication results are right shifted by 12 bits and the adder tree calculates the sum of the

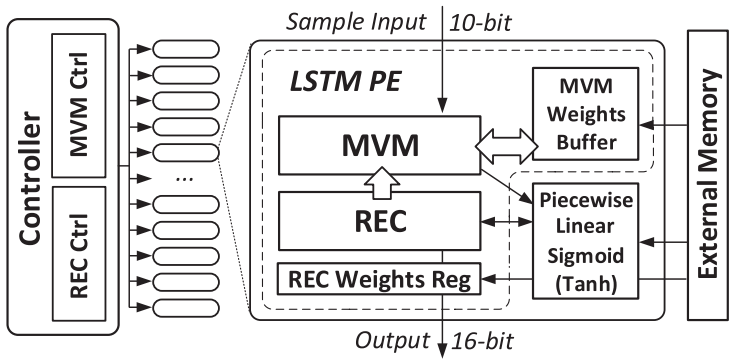

Fig. 8. General architecture of the LSTM inference accelerator.

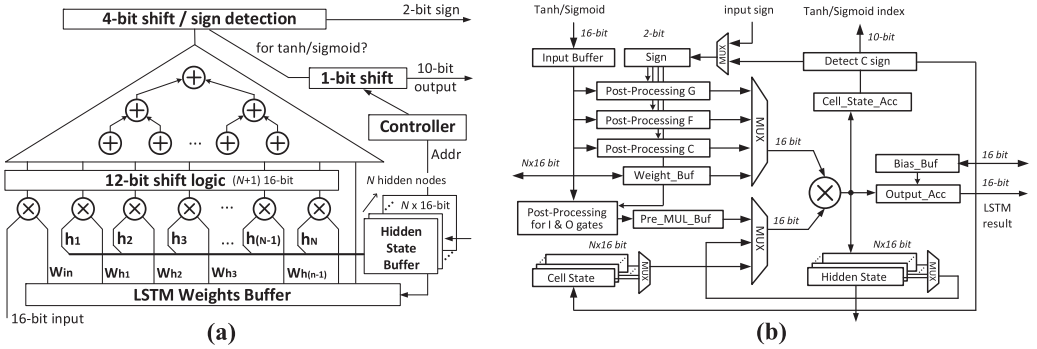

Fig. 9. Schematic diagrams of the (a) MVM and (b) REC modules.

shifted values. The sum is shifted again by 4 bits and a 2-bit sign is calculated for piecewise linear approximation. The MVM operates in N_H iterations for each inference. In each iteration, the input gate, cell gate, forget gate, and output gate are updated in full pipeline since there is no data dependency between two consecutive updates. For the update of the cell gate, the MVM performs an additional 1-bit shift on the result, because the *tanh* and the *sigmoid* operations can be unified by Equation (6). Finally, the hidden node values are updated at the end of each inference by the REC module.

$$\tanh(x) = 2sigmoid(2x) - 1 \tag{6}$$

4.2 REC Design

Figure 9(b) shows the REC circuit. The REC consists of a temporary buffer *Pre_MUL_Buf*, cell and hidden node buffers, a cell value accumulator *Cell_State_Acc*, an output accumulator *Output_Acc*, and weight buffers composed of *Weight_Buf* and *Bias_Buf*. Two multiplexers are used to reconfigure data paths for the multiplication operations. A dedicated circuit is designed to detect the 2-bit sign from the accumulated cell value C and perform post-processing on the fetched data from the look-up table to get the non-linear operation results. The idea of the REC circuit design is to reconfigure the data path to reuse the multiplier and fully pipeline the update of the cell and hidden nodes and the computation of the LSTM inference output defined in Equation (3).

Figure 10 shows the timing diagram of the MVM and REC. Each iteration of the REC update is pipelined into five consecutive cycles, in which the input data is updated by the piecewise linear

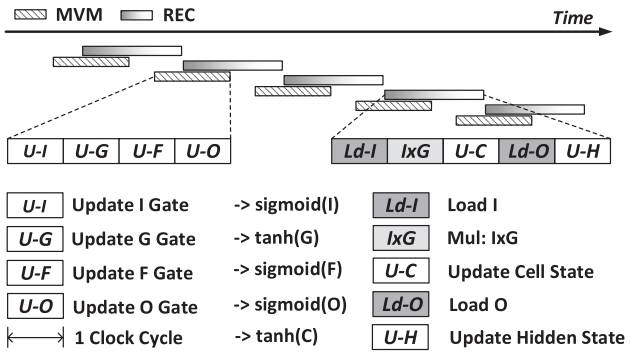

Fig. 10. Timing diagram of the fine-grained pipelines within and between the MVM and REC modules.

approximation corresponding to $sigmoid(I)$, $tanh(G)$, $sigmoid(F)$, $sigmoid(O)$, and $tanh(C)$, respectively. In the first cycle, the approximation value is post-processed to compute the $sigmoid(I)$ according to Equation (2) and the computation result is transferred to the buffer *Pre_MUL_Buf*. Then, the selected weight in the *Weight_Buf* is multiplied by the corresponding hidden node value to update the *Output_Acc*. In the second step, the $sigmoid(I)$ stored in the *Pre_MUL_Buf* is multiplied by the $tanh(G)$ stored in the input buffer. The multiplication result is transferred to the *Cell_State_Acc*. In the third step, the $sigmoid(F)$ from the input buffer is multiplied by the selected cell value and the result is accumulated to the *Cell_State_Acc* to update the cell value. In the next step, the $sigmoid(O)$ is stored in the *Pre_MUL_Buf*. In the last step, the $tanh(C)$ from the input buffer is multiplied by the $sigmoid(O)$ stored in the *Pre_MUL_Buf* to update the corresponding hidden node value. Once all of the N_H iterations are finished, the hidden node values are used to update the hidden node buffer in the MVM.

5 LSTM MODEL COMPRESSION

Quantization and pruning are common neural network compression techniques. The advantage of either one or a combination of both is that it reduces the model size and, thus, improves the inference performance and efficiency without sacrificing much accuracy. In this section, we introduce a different quantization method that can be used for the LSTM inference. We also discuss the pruning that can be combined with this quantization for further improvement on the energy efficiency of the LSTM inference.

5.1 Bit-Sparse Quantization

The quantization method we proposed is based on a number representation that limits the count of 1s in a fixed-length sequence of binary digits, called the *bit-sparse representation*. It is a hardware-friendly representation because it can turn expensive multiplications into simple steps of bit-shifts with/without additions.

The bit-sparse representation is arranged in an n-sbit/M format, in which the n is set far less than the M to satisfy the "sparse" claim. In such a format, no more than n digits are set to 1 while the remaining digits are 0s in an M-digit sequence. For example, the "00010000_00010000" and the "00100000_00000000" can be considered as 2-sbit/16 numbers, whereas the latter but not the former can be viewed as a 1-sbit/16 number.

There are several differences between the bit-sparse and fixed-point representations: (1) The count of 1s in an n-sbit/M bit-sparse number never exceeds n, which is far less than the bit-width M. In contrast, an M-bit fixed-point number can be densely packed with 1s. (2) The bit-sparse

Algorithm 1 Bit-Sparse Quantization

```
1:  procedure
2:      weight ← original weight
3:      n ← number of sparse bits
4:      M ← number of quantization bits
5:  top:
6:      sign ← sign bit of weight
7:      wint ← quantize absolute of weight to M-bit
8:      i ← M − 1
9:      count ← 0
10: loop:
11:     if (wint ≫ i) % 2 = 1 then
12:         count ← count + 1.
13:         if count = n then
14:             wint = ((wint + (1 ≪ (i − 1))) ≫ i) ≪ i.
15:             goto end;
16:     i ← i − 1.
17:     if i = 0 then
18:         goto end;
19:     goto loop.
20: end:
21:     wint ← add sign bit to wint
22:     return wint
```

Fig. 11. Pseudocode for the bit-sparse quantization.

representation tends to require fewer binary bits to encode the number. For example, it requires only 5 bits to encode the position of the 1 in a 1-sbit/16 number. (3) The bit-sparse representation can expand the dynamic range of data representation. Comparing a 1-sbit/16 with a 5-bit fixed-point number, the dynamic range difference is $2^{15}/2^4$ = 2,048x, which surpasses three orders of magnitude. The enlarged dynamic range expands the exploration space of an accuracy–efficiency trade-off for the LSTM inference.

We apply bit-sparse quantization on the LSTM weights inside the hidden layer through quantization-aware training. In the training process, we quantize the weights into the bit-sparse format for feedforward inference and keep them in floating point for the error backpropagation and weights update. As the training progresses, the weights converge into the bit-sparse format and become ready to be deployed for inference. Figure 11 shows the pseudocode of the detailed bit-sparse quantization process. In this process, we first extract the sign from the weights, and then quantize the weight amplitude into an M-bit fixed-point *wint*. We count the number of 1s from the most significant bit (MSB) to the least significant bit (LSB) of the *wint*. When the number reaches n, we round up the *wint* by setting the rest of bits 0, and attach the sign back. The rounding process causes no increase in the number of 1s, and the final converted number complies with the n-sbit/M format.

Figure 12 shows the circuits design for the bit-sparse LSTM inference kernel. The circuits feature a 1D array of bit-shift operators with the size (N_H+1), which corresponds to 1 input and N_H hidden nodes. In this design, we use the bit-shift operation to replace the conventional fixed-point multiplication, as we have quantized the LSTM weights on the feedforward path into the 1-sbit/16 representation. As the bit-shift operator takes much less circuit area than the multiplier, this design largely reduces the MVM area. In addition, the weights quantized in 1-sbit/16 width can be encoded in 5 bits, which helps reduce the local weight memory compared with 8-bit and 16-bit representations. Since the bit-shift operation neither increases the cycle count nor degrades the clock frequency, the inference latency remains unchanged. Under the same inference performance, the reduced circuit area contributes to reduced power consumption and improved energy efficiency.

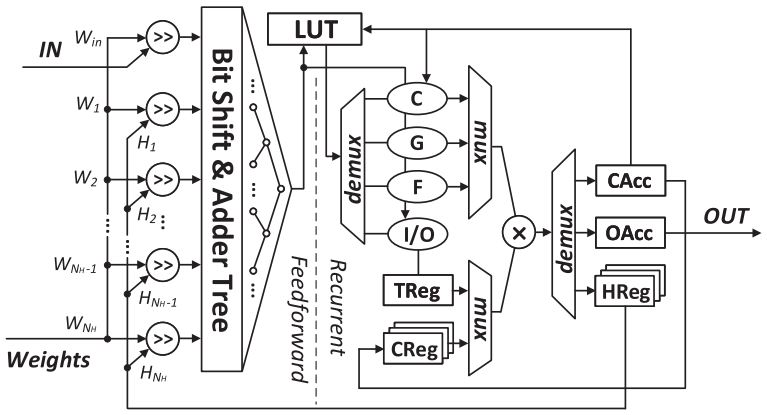

Fig. 12. Circuits design for the bit-sparse LSTM inference kernel.

5.2 Weight Pruning

We apply the pruning in addition to the bit-sparse quantization to improve the energy efficiency of the LSTM inference further. Our pruning method aims at reducing the number of bit-shift operators (under bit-sparse quantization) for the MVM module of the LSTM inference accelerator.

Our original N_H-node LSTM accelerator design requires N_H bit-shift operators for the MVM module, as Figure 13(a) shows. By removing a portion of recurrent connections in the LSTM layer, the number of bit-shift operations required at each clock cycle can be reduced. For example, consider that we set the weights on the recurrent connections from all of the hidden nodes to the hidden node H_1 to 0, as illustrated in Figure 13(b). In this case, the update of I, F, G, O states does not depend on the H_1 state according to Equation (2). Thus, we can save 1 bit-shift operator and simplify the adder tree in the MVM design. Considering that $N_H = 5$, removing connections from a single hidden node leads to a 20% increase on sparsity. In another case, if we eliminate recurrent connections from all of the hidden nodes except for the H_5 as displayed in Figure 13(c), the update of the gates will depend only on the H_5 state and the MVM module can be realized by a single bit-shift operator. As we prune away connections from 4 out of 5 hidden nodes, the sparsity of the hidden layer becomes 80%.

We integrated pruning with bit-sparse quantization into a holistic LSTM training process. During the feedforward inference of the training, we applied pruning followed by bit-sparse quantization. The error backpropagation took place at all connection paths in the floating point. We evaluated the performance of the LSTM inference accelerator under a variety of data representations with or without quantization: floating point (fp), 16-bit, 12-bit, 8-bit, 4-bit, 1-sbit/16 (1sb) and 2-sbit/16 (2sb), in combination with pruning at different levels by getting rid of connections from 0∼5 hidden nodes in a 5-node LSTM. Detailed experimental results and analysis are discussed in the next section.

6 EXPERIMENTS

6.1 Accuracy under Quantization and Pruning

We evaluated the accuracy performance of LSTM inference under the proposed bit-sparse quantization and pruning optimizations with 8 test cases. The test cases were extracted from in-vivo EEG and calcium-imaging applications: EEG analytical processing for Theta and Gamma bands [7], non-rigid motion vector prediction [6], and enhanced cell fluorescence trace extraction [5] from calcium images. We summarize the number of samples in the test cases in Table 3. We

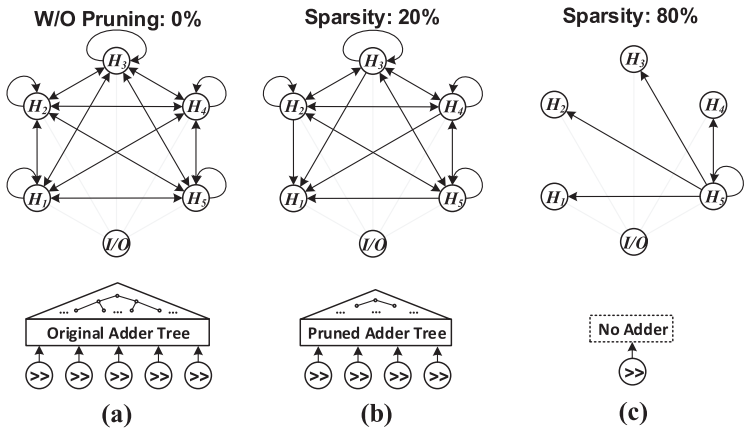

Fig. 13. (a) Pruning of connections from hidden nodes in the LSTM model and (b) its benefits on reducing computation logic in the circuit design.

Table 3. Number of Samples in the Training and Test Sequences in Different Test Cases

	Samples in Training Seq.	Samples in Test Seq.
EEG processing for the Theta band	20,000	57,600
EEG processing for the Gamma band	36,000	460,800
Non-rigid motion vector prediction	25,000	1,000
Enhanced cell fluorescence trace extraction	3,300	6,600

used a common LSTM inference model with a signal layer and 5 hidden nodes ($N_H = 5$) for the targeted EEG and calcium-imaging applications, considering the trade-off between accuracy and efficiency [7]. Our accelerator is scalable and can support LSTM inference with more hidden nodes by increasing the number of multiplication/bit-shift operators in the MVM module described in Sections 4.1 and 5.1.

We evaluated a combination of pruning and quantization optimizations for LSTM inference on the selected test cases. We measured LSTM inference accuracy by calculating the normalized cross correlation ρ between the inferenced and targeted sequences X and Y on the test set:

$$\rho = \frac{E\left[(X - \mu_X) \cdot (Y - \mu_Y)\right]}{\sigma_X \sigma_Y}, \tag{7}$$

where E is the expected value operator and μ_X (μ_Y) and σ_X (σ_Y) represent the mean and standard deviations of the sequence $X(Y)$, respectively.

In order to increase repeatability, we trained each LSTM independently 5 times and recorded the maximum ρ among the trials. Figure 14 shows heat maps of normalized cross correlation values under different combinations of quantization and pruning strategies for all selected datasets. Each row (column) of the heat map corresponds to a specific quantization (pruning) condition for the inference. A larger normalized cross correlation value labeled in a darker color indicates a higher LSTM inference accuracy.

We can make several observations from the evaluation. (1) The accuracy performance is very data dependent, but a general trend is that the accuracy tends to get worse as the quantization and pruning become more aggressive, which means shorter bit-width and more pruned weights.

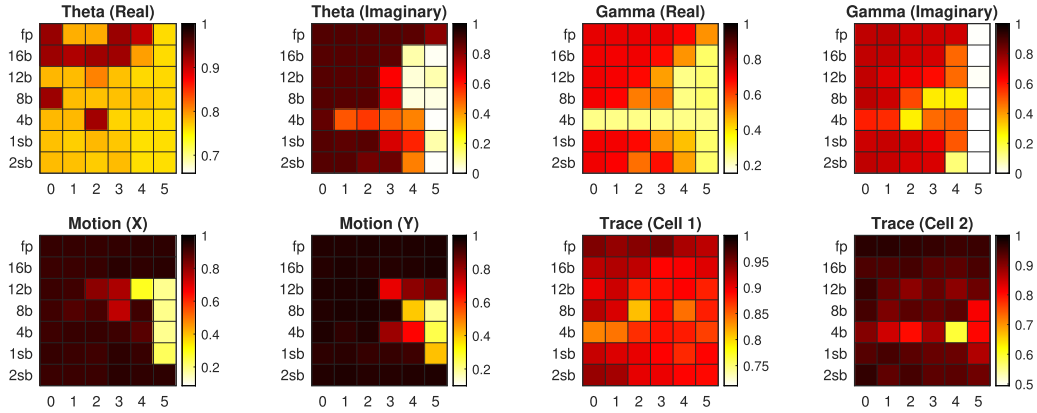

Fig. 14. Accuracy measured in normalized cross correlation after pruning and quantization.

Table 4. Circuit Area and Power Consumption Saving in Using Customized Bit Quantization

Quantization	16 bit	1-sbit/16	2-sbit/16	2-sbit/16	2-sbit/16	2-sbit/16	2-sbit/16
Pruning strategy	-	-	-	Prune-20%	Prune-40%	Prune-60%	Prune-80%
MVM area (μm^2)	4,339	1,781	3,893	3,287	2,656	2,034	1,359
Reduction	-	59.0%	10.3%	24.2%	38.8%	53.1%	68.7%
PE area (μm^2)	17,119	7,704	12,312	10,837	9,345	7,865	6,328
Reduction	-	55.0%	28.1%	36.7%	45.4%	54.1%	63.0%
PE power (mW)	10.01	4.58	6.62	5.87	5.12	4.37	3.57
Reduction	-	54.2%	33.9%	41.4%	48.9%	56.3%	64.3%

(2) The 1-sbit/16 quantization constantly outperforms the 4-bit fixed-point on inference accuracy. This can be explained by a wide dynamic range provided by the bit-sparse data representation compared with the fixed-point. (3) The 2-sbit/16 quantization always works better than the 1-sbit/16, and its performance can match 12-bit or even 16-bit when applied with a 3-node pruning. These experimental results prove that it is promising to combine bit-sparse quantization with pruning for more efficient LSTM inference with no accuracy loss.

6.2 Area and Energy Efficiency

We designed the proposed LSTM inference accelerator in a 28-nm process using the Synopsys Design Compiler. We estimated the circuit area and power consumption of the accelerator under a typical process corner and 600-MHz clock frequency, which is adjustable according to specific application requirements. Table 4 shows the evaluation results on the circuit area and the power reduction under various combinations of pruning and quantization strategies. As the comparison result shows, the proposed 1-sbit/16 quantization achieves a significant amount of saving on the circuit area and power consumption, 55.0% and 54.2%, respectively. Under the 2-sbit/16 quantization, as we increase the level of sparsity by applying pruning, we gain more saving on the circuit area and power consumption. The area and power saving benefit from the reduction of both the MVM module and the LSTM model footprint.

We evaluated the achievable accuracy and energy efficiency under various pruning and quantization combinations across datasets, as Figure 15 shows. We can make several observations on the evaluation results: (1) In most cases except for the band pass filtering for the real part of the Theta signal, the LSTMs optimized by the proposed bit-sparse quantization can achieve similar

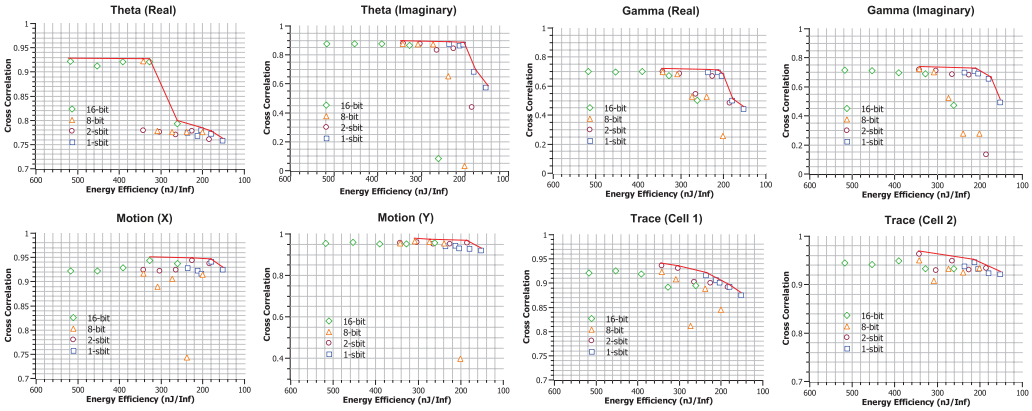

Fig. 15. Trade-off and Pareto frontiers between the energy efficiency and accuracy under different combinations of pruning and quantization strategies across datasets.

accuracy as those implemented in 8-bit or 16-bit. (2) As the required energy efficiency increases, the bit-sparse quantization tends to outperform the conventional fixed-point quantization in terms of accuracy. (3) As the Pareto frontiers shown in Figure 15, the bit-sparse quantization method dominates the optimized solutions with both higher accuracy and better energy efficiency.

We scaled out our LSTM inference accelerator design with 32 PEs. Under the 2-sbit/16 quantization and 3-node pruning, which corresponds to a 60% sparsity on a 5-node LSTM, the accelerator achieves 1.19 GOPS/mW energy efficiency and 439 GOPS/mm^2 area efficiency. In addition, it can reduce the circuit area and power consumption by 54.1% and 56.3%, respectively, compared with the baseline 16-bit implementation. Table 5 summarizes the characteristics of these two designs in comparison with recent LSTM accelerators. Our design achieves comparable energy efficiency and higher area efficiency against prior state-of-the-art. The reason for the high area efficiency is threefold. First, we adopted a compact LSTM model and optimized it with a combination of hardware-friendly bit-sparse quantization and pruning, and we stored the compressed LSTM model inside the local memory of the PE. Second, our accelerator fully utilizes the hardware resource by completely pipelining the computational operations on the feedforward and recurrent paths. Third, we carried out the circuit area estimation in a more advanced technology process.

Our LSTM accelerator features high scalability, because there is little communication overhead among PEs during inference and the computation is carried out locally at each PE. This can benefit some emerging applications, such as brain–machine inference, considering that the count for simultaneous recording sites and/or channels can reach hundreds or even thousands.

7 RELATED WORKS

High-performance RNN inference accelerators have been designed for a variety of temporal prediction tasks, including speech recognition [10, 19, 35], optical character recognition [33], keyword spotting systems [14], video content recognition [26, 34, 37] and text generation [3].

State-of-the-art RNN processors have realized high-performance energy-efficient RNN inference for a variety of time series prediction tasks based on different design techniques. The authors of [26, 34] presented high-performance and highly energy-efficient CNN/RNN accelerators based on weight clustering and low-bit-width quantization. The authors of [14] presented a low-power LSTM processor for keyword spotting under 5 μW and achieved 60 nJ/inference energy efficiency. The authors of [4] proposed a gated recurrent unit processor with online incremental learning

Table 5. Characteristic Comparison with Other LSTM Accelerator Designs for Edge Applications

	This Work	This Work	[36]	[8]	[27]	[23]
Design	Tape-In	Tape-In	Tape-In	Tape-Out	Tape-Out	Tape-Out
Technology	28 nm	28 nm	90 nm	65 nm	65 nm	65 nm
Clock (MHz)	600	600	600	20	200	8/80
Voltage (V)	0.9	0.9	1	1.24/0.75	1.1	0.68/1.1
Num. Hidden Nodes (N_H)	5	5	512	421	-	512×2
Quantization	16 bit	2-sbit/16	8 bit	8-16 bit	4-16 bit	6 bit
Compression	-	Prune-60%	Circulant	-	-	Structure-16×
Area (mm^2)	0.78[1]	0.38[1]	30.77	0.93	1.84	7.74
Power (mW)	320	140	1,010	29.03/1.24	21	1.85/67.3
Energy efficiency (GOPS/mW)	0.52	1.19	2.44	1.11/3.08	1.1	8.93/2.45
Area efficiency (GOPS/mm^2)	214	439	79.9	34.4	12.6	21.3

capability. The authors of [20] presented a highly efficient CNN/RNN accelerator for neural networks compressed by pruning.

LSTM inference accelerator designs have also been extensively studied on FPGA platforms. The authors of [19] propose a load-balance pruning to reduce a large LSTM model size by 10x to high throughput on an XCKU060 FPGA. The authors of [35] propose structured compression to further reduce the storage requirement and computation complexity for even higher performance and energy efficiency. The authors of [10] and [9] leverage temporal redundancy and spatio-temporal sparsity to achieve high throughput and high energy efficiency LSTM inference on the Zynq platforms for real-time speech recognition. The authors of [17, 18] propose a framework to automate the LSTM accelerator design on an FPGA by optimizing computation and communication. The authors of [31] propose a multi-thread RNN/LSTM inference accelerator on a Stratix 10 PFGA for cloud-based applications. The authors of [32, 37] employ 12-bit or even shorter bit width for an LSTM inference accelerator design.

Compared with these previous works, we target specific application domains in which compact LSTM models can play an important role. Our proposed LSTM inference accelerator achieves short latency and high energy efficiency in EEG signal and calcium-image processing and is suitable to be integrated in a closed-loop neurofeedback device under stringent timing and energy constraints.

8 CONCLUSION

In this article, we present three different case studies to show that the compact LSTM model has the potential to replace quite a few conventional acausal algorithms by making causal and accurate predictions for many real-time applications, especially for those ones that undergo strict latency and energy cost requirements. In addition, we design an LSTM inference accelerator and optimize it with hardware friendly pruning and quantization to improve energy efficiency. We think that the combination of our proposed method and design can expand the solution field to include a wide range of emerging edge applications, including and beyond closed-loop feedback for brain–machine interfaces.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Daniel Aharoni and Dr. Changliang Guo for their support on the Miniscope, thank Dr. Andrew Howe and Garrett J. Blair for help on the data collection from rats and Prof. Peyman Golshani for leading the collaborative project across departments at UCLA.

[1]Estimated by the reported cell area with a 50% place and route overhead.

REFERENCES

[1] Daniel Aharoni, Baljit S. Khakh, Alcino J. Silva, and Peyman Golshani. 2019. All the light that we can see: A new era in miniaturized microscopy. *Nat. Methods* 16, 1 (2019), 11–13.

[2] György Buzsaki. 2006. *Rhythms of the Brain.* Oxford University Press.

[3] Andre Xian Ming Chang and Eugenio Culurciello. 2017. Hardware accelerators for recurrent neural networks on FPGA. In *IEEE Int. Symp. Circuits Syst. (ISCAS).* 1–4.

[4] Chixiao Chen, Hongwei Dong, Huwan Peng, Haozhe Zhu, Rui Ma, Peiyong Zhang, Xiaolang Yan, Yu Wang, Mingyu Wang, Hao Min, and Richard C.-J. Shi. 2017. OCEAN: An on-chip incremental-learning enhanced processor with gated recurrent neural network accelerators. In *IEEE Eur. Solid State Circuits Conf. (ESSCC).* 259–262.

[5] Zhe Chen, Garrett J. Blair, Hugh T. Blair, and Jason Cong. 2020. BLINK: Bit-sparse LSTM inference kernel enabling efficient calcium trace extraction for neurofeedback devices. In *Proc. Int. Symp. Low Power Electron. Des. (ISLPED).* 217–222.

[6] Zhe Chen, Hugh T. Blair, and Jason Cong. 2019. LANMC: LSTM-assisted non-rigid motion correction on FPGA for calcium image stabilization. In *Proc. Int. Symp. Field-Programmable Gate Arrays (FPGA).* ACM, 104–109.

[7] Zhe Chen, Andrew Howe, Hugh T. Blair, and Jason Cong. 2018. CLINK: Compact LSTM inference kernel for energy efficient neurofeedback devices. In *Proc. Int. Symp. Low Power Electron. Des. (ISLPED).* 2:1–2:6.

[8] Francesco Conti, Lukas Cavigelli, Gianna Paulin, Igor Susmelj, and Luca Benini. 2018. CHIPMUNK: A systolically scalable 0.9 mm 2, 3.08Gop/s/mW @ 1.2 mW accelerator for near-sensor recurrent neural network inference. In *IEEE Custom Integrated Circuits Conf. (CICC'18).* 1–4.

[9] Chang Gao, Tobi Delbruck, and Shih-Chii Liu. 2021. Spartus: A 9.4 TOp/s FPGA-based LSTM accelerator exploiting spatio-temporal sparsity. (2021). arXiv:2108.02297.

[10] Chang Gao, Daniel Neil, Enea Ceolini, Shih-Chii Liu, and Tobi Delbruck. 2018. DeltaRNN: A power-efficient recurrent neural network accelerator. In *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA).* ACM, 21–30.

[11] Kunal K. Ghosh, Laurie D. Burns, Eric D. Cocker, Axel Nimmerjahn, Yaniv Ziv, Abbas El. Gamal, and Mark J. Schnitzer. 2011. Miniaturized integration of a fluorescence microscope. *Nat. Methods* 8 (2011), 871.

[12] Andrea Giovannucci, Johannes Friedrich, Pat Gunn, Jérémie Kalfon, Brandon L. Brown, Sue Ann Koay, Jiannis Taxidis, Farzaneh Najafi, Jeffrey L. Gauthier, Pengcheng Zhou, Baljit S. Khakh, David W. Tank, Dmitri B. Chklovskii, and Eftychios A. Pnevmatikakis. 2019. CaImAn an open source tool for scalable calcium imaging data analysis. *eLife* 8 (2019), e38173.

[13] Andrea Giovannucci, Johannes Friedrich, Matthew Kaufman, Anne K. Churchland, Dmitri Chklovskii, and Liam Paninski. 2017. OnACID: Online analysis of calcium imaging data in real time. In *Adv. Neural Inf. Process. Syst. (NIPS).* 2378–2388.

[14] J. S. P. Giraldo and Marian Verhelst. 2018. Laika: A 5uW programmable LSTM accelerator for always-on keyword spotting in 65 nm CMOS. In *IEEE Eur. Solid State Circuits Conf. (ESSCIRC).* 166–169.

[15] Alex Graves, Marcus Liwicki, Santiago Fernandez, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 5 (2009), 855–868.

[16] Alex Graves, Abdel Rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP).* 6645–6649.

[17] Yijin Guan, Hao Liang, Ningyi Xu, Wenqiang Wang, Shaoshuai Shi, Xi Chen, Guangyu Sun, Wei Zhang, and Jason Cong. 2017. FP-DNN: An automated framework for mapping deep neural networks onto FPGAs with RTL-HLS hybrid templates. In *IEEE 25th Annu. Int. Symp. Field-Programmable Cust. Comput. Mach. (FCCM).* 152–159.

[18] Yijin Guan, Zhihang Yuan, Guangyu Sun, and Jason Cong. 2017. FPGA-based accelerator for long short-term memory recurrent neural networks. In *22nd Asia South Pacific Des. Autom. Conf. (ASP-DAC).* 629–634.

[19] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, Huazhong Yang, and William J. Dally. 2017. ESE: Efficient speech recognition engine with sparse LSTM on FPGA. In *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA).* 75–84.

[20] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient inference engine on compressed deep neural network. In *Proc. Int. Symp. Comput. Archit. (ISCA).* 243–254.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9, 8 (1997), 1735–1780.

[22] Forester W. Isen. 2009. *DSP for MATLAB and LabVIEW III: Digital Filter Design.* Morgan and Claypool Publishers.

[23] Deepak Kadetotad, Shihui Yin, Visar Berisha, Chaitali Chakrabarti, and Jae sun Seo. 2020. An 8.93 TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity for on-device speech recognition. *IEEE J. Solid-State Circuits* 55, 7 (2020), 1877–1887.

[24] Abbas Kazemipour, Ondrej Novak, Daniel Flickinger, Jonathan S. Marvin, Ahmed S. Abdelfattah, Jonathan King, Philip M. Borden, Jeong Jun Kim, Sarah H. Al-Abdullatif, Parker E. Deal, Evan W. Miller, Eric R. Schreiter, Shaul Druckmann,

Karel Svoboda, Loren L. Looger, and Kaspar Podgorski. 2019. Kilohertz frame-rate two-photon tomography. *Nat. Methods* 16, 8 (2019), 778–786.

[25] G. Lazzi. 2005. Thermal effects of bioimplants. *IEEE Eng. Med. Biol. Mag.* 24, 5 (2005), 75–81.

[26] Jinmook Lee, Changhyeon Kim, Sanghoon Kang, Dongjoo Shin, Sangyeob Kim, and Hoi-Jun Yoo. 2018. UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In *IEEE Int. Solid-State Circuits Conf. (ISSCC)*. 218–220.

[27] Jinmook Lee, Dongjoo Shin, and Hoi-Jun Yoo. 2017. A 21mW low-power recurrent neural network accelerator with quantization tables for embedded deep learning applications. In *IEEE Asian Solid-State Circuits Conf. (ASSCC)*. 237–240.

[28] Jinghao Lu, Chunyuan Li, Jonnathan Singh-Alvarado, Zhe Charles Zhou, Flavio Fröhlich, Richard Mooney, and Fan Wang. 2018. MIN1PIPE: A miniscope 1-photon-based calcium imaging signal extraction pipeline. *Cell Rep.* 23, 12 (2018), 3673–3684.

[29] David P. Nguyen, Stuart P. Layton, Gregory Hale, Stephen N. Gomperts, Thomas J. Davidson, Fabian Kloosterman, and Matthew A. Wilson. 2009. Micro-drive array for chronic in vivo recording: Tetrode assembly. *J. Vis. Exp.* 26 (2009), e1098.

[30] Eftychios A. Pnevmatikakis and Andrea Giovannucci. 2017. NoRMCorre: An online algorithm for piecewise rigid motion correction of calcium imaging data. *J. Neurosci. Methods* 291 (2017), 83–94.

[31] Zhiqiang Que, Hiroki Nakahara, Hongxiang Fan, Jiuxi Meng, Kuen Huang Tsoi, Xinyu Niu, Eriko Nurvitadhi, and Wayne Luk. 2020. A reconfigurable multithreaded accelerator for recurrent neural networks. In *Int. Conf. Field-Programmable Technol. (FPT)*. 20–28.

[32] Vladimir Rybalkin, Alessandro Pappalardo, Muhammad Mohsin Ghaffar, Giulio Gambardella, Norbert Wehn, and Michaela Blott. 2018. FINN-L: Library extensions and design trade-off analysis for variable precision LSTM networks on FPGAs. In *Int. Conf. Field-Programmable Log. Appl. (FPL)*. 890–897.

[33] Vladimir Rybalkin, Norbert Wehn, Mohammad Reza Yousefi, and Didier Stricker. 2017. Hardware architecture of bidirectional long short-term memory neural network for optical character recognition. In *Des. Autom. Test Eur. Conf. Exhib. (DATE)*. 1390–1395.

[34] Dongjoo Shin, Jinmook Lee, Jinsu Lee, and Hoi-Jun Yoo. 2017. DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks. In *IEEE Int. Solid-State Circuits Conf. (ISSCC)*. 240–241.

[35] Shuo Wang, Zhe Li, Caiwen Ding, Bo Yuan, Qinru Qiu, Yanzhi Wang, and Yun Liang. 2018. C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs. In *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays (FPGA)*. 11–20.

[36] Zhisheng Wang, Jun Lin, and Zhongfeng Wang. 2017. Accelerating recurrent neural networks: A memory-efficient approach. *IEEE Trans. Very Large Scale Integration (VLSI) Syst.* 25, 10 (2017), 2763–2775.

[37] Xiaofan Zhang, Xinheng Liu, Anand Ramachandran, Chuanhao Zhuge, Shibin Tang, Peng Ouyang, Zuofu Cheng, Kyle Rupnow, and Deming Chen. 2017. High-performance video content recognition with long-term recurrent convolutional network for FPGA. In *Int. Conf. Field-Programable Log. Appl. (FPL)*. 1–4.

[38] Pengcheng Zhou, Shanna L. Resendez, Jose Rodriguez-Romaguera, Jessica C. Jimenez, Shay Q. Neufeld, Andrea Giovannucci, Johannes Friedrich, Eftychios A. Pnevmatikakis, Garret D. Stuber, Rene Hen, Mazen A. Kheirbek, Bernardo L. Sabatini, Robert E. Kass, and Liam Paninski. 2018. Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. *eLife* 7 (2018), e28728.

Received July 2021; revised October 2021; accepted October 2021