electronics

MDPI

Article

Approximate LSTM Computing for Energy-Efficient Speech Recognition

Junseo Jo [1], Jaeha Kung [2] and Youngjoo Lee [1,*]

[1] Department of Electrical Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, Korea; joejs@postech.ac.kr

[2] Department of Information and Communication Engineering, Daegu Gyeongbuk Institute of Sceince and Technology (DGIST), Daegu 42988, Korea; jhkung@dgist.ac.kr

* Correspondence: youngjoo.lee@postech.ac.kr

Received: 30 October 2020; Accepted: 18 November 2020; Published: 25 November 2020

Abstract: This paper presents an approximate computing method of long short-term memory (LSTM) operations for energy-efficient end-to-end speech recognition. We newly introduce the concept of similarity score, which can measure how much the inputs of two adjacent LSTM cells are similar to each other. Then, we disable the highly-similar LSTM operations and directly transfer the prior results for reducing the computational costs of speech recognition. The pseudo-LSTM operation is additionally defined for providing the approximate computation with reduced processing resolution, which can further relax the processing overheads without degrading the accuracy. In order to verify the proposed idea, in addition, we design an approximate LSTM accelerator in 65 nm CMOS process. The proposed accelerator newly utilizes a number of approximate processing elements (PEs) to support the proposed skipped-LSTM and pseudo-LSTM operations without degrading the energy efficiency. Moreover, sparsity-aware scheduling is introduced by introducing the small-sized on-chip SRAM buffer. As a result, the proposed work provides an energy-efficient but still accurate speech recognition system, which consumes 2.19 times less energy than the baseline architecture.

Keywords: approximate computing; LSTM architecture; speech recognition; VLSI design

1. Introduction

In the last few years, speech recognition using deep neural networks (DNNs) has become an important challenge in the internet-of-things (IoT) communities as well as the artificial intelligence (AI) industries [1]. In general, to handle sequential data such as voice signals, it is well known that the recurrent neural network (RNN) is suitable to improve the recognition accuracy by activating the prior histories during the inference processing [2]. In order to achieve the attractive algorithm-level performance that can be even applied to the commercialized products; however, the contemporary RNN-based algorithm normally necessitates a huge amount of computing resources [3]. For example, the recent long short-term memory (LSTM) model utilizes additional gate operations to provide state-of-the-art end-to-end speech recognition; however, it significantly increased the memory overheads storing the excessive number of trained parameters [4]. Considering the amount of accessing energy for external DRAMs [5], therefore, the modern DNN-based speech recognition solutions are mainly available at the server-scale or cloud-level computing platforms. Targeting the on-device speech recognition, on the other hand, it is impractical to directly apply the recent LSTM architectures at the resource-limited IoT or mobile platforms.

Several studies have presented optimization techniques for reducing energy consumption of neural network processing [6,7]. For example, the work from [8] provides the energy-accuracy trade-off by utilizing the control gate, which is suitable for handling the image classification problem. The pruning-based approach can reduce the number of multiply-accumulate (MAC) operations as well as the number of memory accesses by eliminating less important parameters [9]. Considering the continuous speech signal, the conventional LSTM-based network is modified in [10] by adding more gate networks that determine the activation of state updates. As the computing steps in LSTM operations can be expressed as matrix-to-matrix multiplications, the approximate computing from the singular-value decomposition (SVD) is applied to relax the computational complexity without degrading the algorithm-level accuracy [11]. It is also possible to further reduce the computing costs of each RNN/LSTM cell by adopting the concept of delta-network, which adjusts the level of approximate parts by changing the sparsity level dynamically [12]. More aggressive method adopting binary neural network (BNN) can be applied to minimize the energy consumption of DNN processing [13]. However, it is generally reported that such a BNN-based method severely degrades the recognition performance; even its flexible hardware can be easily implemented by utilizing the latch-based scratch-pad memories (SCMs). In this work, the proposed work reduces computational complexity through an approximate computing approach that focuses on complicated networks utilizing multiple bits to represent weight values, which normally provide more accurate accuracy than BNN models.

In addition to the previous works, there are still potential rooms to enhance the energy-efficiency of DNN-based speech recognition if we carefully tack account of the properties of input speech signals. Compared to the practical time interval to construct each input state of LSTM (or RNN) structure, more precisely, the input states from the speech data change quite slowly compared with other types of inputs [12]. Therefore, it is reasonable to assume that the adjacent input states tend to be highly similar to each other. Starting from our prior work in [14], this paper presents an energy-efficient LSTM accelerator design that performs the approximate LSTM operations based on the similarity-based cell activation. As described in [14], we first introduce the similarity score measuring how two adjacent LSTM inputs are similar to each other. The skipping method is then applied by limiting the number of consecutive disabled cells, remarkably reducing the number of activated cells during the end-to-end speech recognition with acceptable accuracy. The computational cost is further relaxed by adopting the pseudo-skipping approach, which is basically the variation of delta-LSTM processing in [12].

In the proposed accelerator, we first realize the efficient controller architecture to support the on-demand calculation of the similarity score. The processing elements (PEs) are then carefully designed to support two types of MAC operations dynamically, i.e., the normal full-precision MAC computations and the approximate MAC processing with the reduced precision. The new data scheduling technique is proposed to allow the small-sized SRAM buffer by latching only the survived parameters to handle the sparse matrix-to-matrix multiplications. As a result, the prototype LSTM accelerator in 65nm CMOS technology supports the low-power on-device speech recognition by achieving the energy-efficiency of 211 GOPS/W, which is only 49% of the required energy at the baseline implementation.

The rest of this paper is organized as follows. Section 2 describes the background knowledge of the end-to-end speech recognition using the DeepSpeech Network [15], which is the baseline model of this work. Section 3 describes the algorithm-level innovations of the proposed method, including skipping and approximate cell operations. The proposed accelerator architecture is then detailed with the hardware-level optimizations in Section 4. Simulation and implementation results are shown and compared to the prior works in Section 5, and the concluding remarks are finally made in Section 6.

2. Background

2.1. DNN-Based End-to-End Speech Recognition

Figure 1 depicts how the DNN-based end-to-end speech recognition handles the input speech signal and generates the final recognition results. Note that the input speech signal in the time domain is divided into equal-length windows, i.e., each window contains the same number of speech samples. Then, we capture the essential features of the input window, which is normally based on the frequency-domain analysis [16]. The captured features are issued at the DNN-based recognition system, resulting in a character with the most probability for each window input. As we perform the window-level DNN processing, as illustrated in Figure 1, the dedicated decoder should be added at the end of the recognition system to construct the final output sentence.

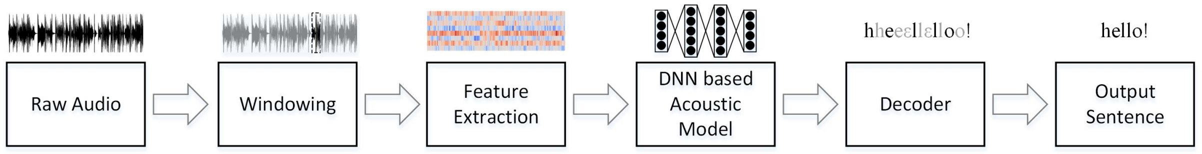

Figure 1. Processing overview of end-to-end speech recognition.

In the baseline system used for this work, considering the contemporary speech processing works [17], we adopt the time interval of 100 ms for making the input speech windows. After performing FFT processing, numerous Mel-frequency cepstral coefficient (MFCC) values in the frequency domain are used as features to be the input state of the LSTM-based network where the internal processing layers will be described later. Similar to the other state-of-the-art solutions [18], the connectionist temporal classification (CTC) step is finally applied for realizing the decoder architecture, removing the copies of detected characters, as shown in Figure 1.

2.2. *DeepSpeech Network and LSTM Operations*

For the DNN-based recognition system, in this work, we adopt the recent DeepSpeech network from [15] as a baseline network to develop our energy-efficient optimization schemes. Accepting the input MFCC features, as illustrated in Figure 2, the DeepSpeech network utilizes several processing layers; three fully-connected layers (FC1~FC3) for tailoring the input MFCC features, a bi-directional LSTM layer over n feature sets, and two fully-connected layers (FC4~FC5) for making the reliability values of 29 character candidates. Here, n denotes the number of windows for the input speech data. Note that the character candidates include 26 alphabets, an apostrophe symbol, a blank symbol, and a punctuation mark. With the proper pre-/post-processing steps, including feature extractions and CTC operations, as described in [14], the DeepSpeech network achieved an attractive word-error-rate (WER) of 9.33% when it applied to recognize Librispeech dataset [19].

The LSTM cell operation in the DeepSpeech network is a key computation for providing such a high recognition quality. More precisely, as described in Figure 3, each LSTM operation receives the current input data, which is a $d \times 1$ vector $\mathbf{x}_t$, and the hidden state and cell state of the previous LSTM unit, which are two $k \times 1$ vectors denoted as $\mathbf{h}_{t-1}$ and $\mathbf{c}_{t-1}$, respectively. Note that the subscript t here is the index of the window sample, representing the timing stamp during the recognition process. The LSTM cell then computes the output cell and hidden states, i.e., $\mathbf{c}_t$ and $\mathbf{h}_t$. More precisely, three gate operations called input, output, and forget gates, first generate internal vectors denoted as $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$, respectively, which are formulated as follows.

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{1}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{3}$$

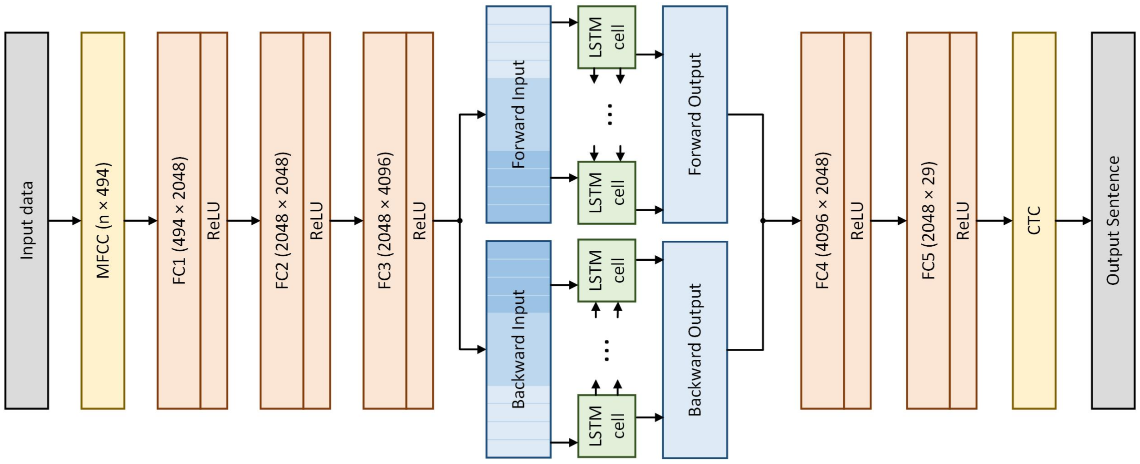

Figure 2. DeepSpeech network architecture.

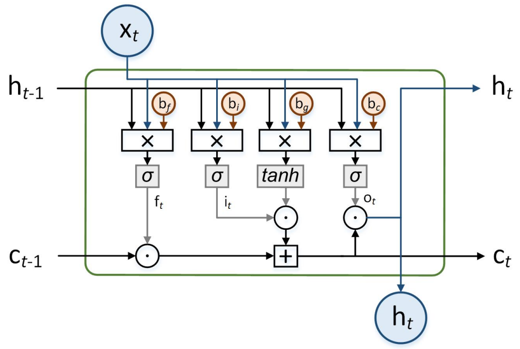

Figure 3. LSTM cell architecture including three gate operations.

To reflect the contributions of input data and the hidden state, as shown above, we introduce two types of weight matrices $\mathbf{W}_x$ and $\mathbf{W}_h$ at each gate operation. As elements in each matrix are trained individually for the gate operations, it is required to keep a total of eight matrices for realizing the LSTM cell. In addition, a vector **b** having the same size as the input data is used as bias values. As depicted in Figure 3, note that non-linear operators like a sigmoid function and a hyperbolic tangent function are applied at the end of gate operations. The final outputs of the LSTM cell are then calculated as follows, where $\odot$ stands for the element-wise product between two vectors.

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_{xg}\mathbf{x}_t + \mathbf{W}_{hg}\mathbf{h}_{t-1} + \mathbf{b}_g) \quad (4)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (5)$$

3. Similarity-Based LSTM Operation

Although the LSTM-based DeepSpeech network achieves an attractive accuracy, it requires a huge amount of computing cost and network size, which is difficult to apply in resource-limited systems such as mobile and IoT environments. After performing feature extraction, more specifically, the baseline DeepSpeech network converts the input speech data into n series of features, each of which includes 494 MFCC elements. Compared to the five FC layers utilizing 22M MAC operations, as reported in [14], the bi-directional LSTM structure actually dominates the overall computational costs by requiring more than 100M MAC computations. In order to improve the efficiency of DNN-based speech recognition, therefore, it is necessary to relax the computing overheads of LSTM processing.

3.1. Cell-Skipping Method Using Similarity Score

For the input voice signal, in general, the MFCC-based speech recognition manages the excessive number of windows for taking account of various speakers having different speeds. Therefore, even a single character input is divided into several windows having similar MFCC coefficients, generating similar features handled by consecutive LSTM cells. As a result, in the baseline DeepSpeech system in Figure 2, the output sequence at the FC5 layer contains several identical characters for a single input character, and the final CTC step is inevitable to remove these redundant results. As described in our conference paper [14], we newly propose the cell-level deactivation and approximation schemes for the energy-efficient speech recognition by detecting these redundant positions at the run time, which is conceptually illustrated in Figure 4a. At the time stamp of t, the LSTM cell compares the current input $\mathbf{x}_t$ with the prior one $\mathbf{x}_{t-1}$ to obtain the similarity score S_t before activating its gate operations. More precisely, we can compute the similarity score at the t-th window sample as follows.

$$S_t = \frac{C(\mathbf{x}_t/2^w - M(\mathbf{x}_t) \odot \mathbf{x}_{t-1}/2^w) - C(\mathbf{x}_t)}{C(\mathbf{x}_t)} \tag{6}$$

For the input $1 \times k$ vector, the function $C(\cdot)$ returns the number of nonzero elements, whereas the function $M(\cdot)$ generates a $1 \times k$ masking vector at which its elements become one for the nonzero positions and otherwise zero. In order to ignore the small variations in two adjacent inputs, we define the clipping parameter w to adjust the meaningful ranges for computing the similarity score. As reported in [14], input vectors with high similarity scores successfully correspond to the redundant time positions that generate the characters to be deleted at the later CTC step.

The similarity-based cell skipping method is then simply introduced by setting the skipping threshold of θ_s. When the current S_t is larger than the pre-defined threshold, then we simply copy the prior outputs without activating the LSTM cell operations, as exemplified in Figure 4b, which can significantly save the overall computational costs. As the consecutive skips may degrade the recognition accuracy severely, as described in [14], it is possible to limit the number of serial skips properly, leading to the energy-efficient and accurate LSTM processing scenario. In this work, to develop the cost-effective LSTM accelerator, we allow only two consecutive skips of LSTM operations for highly-similar inputs, which can reduce the overall complexity by 26.8% under the 1% accuracy drop compared to the baseline LSTM operations activating all the cells [14].

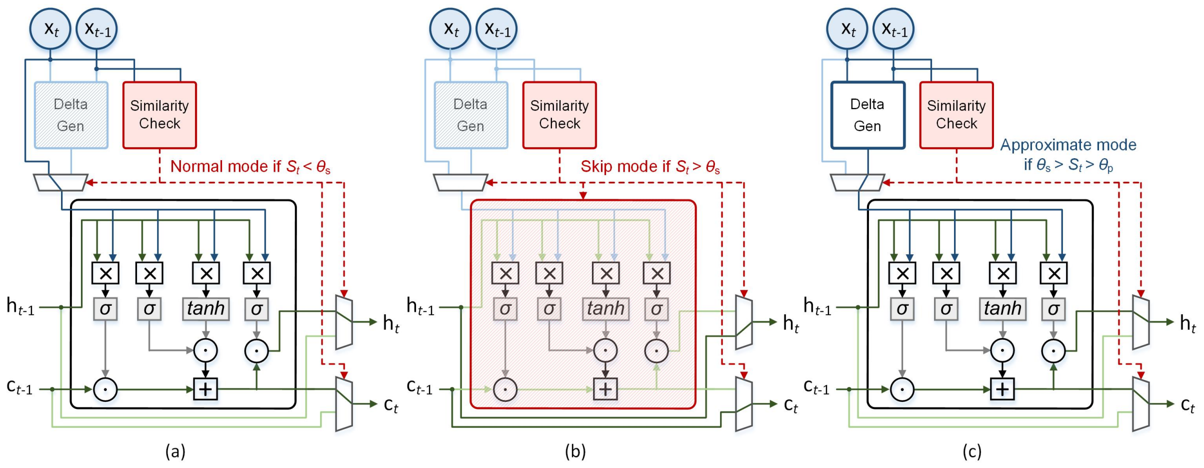

Figure 4. Proposed LSTM cell processing for (**a**) normal mode, (**b**) skip mode, and (**c**) approximate mode.

3.2. Pseudo-Skipping Method for Approximate LSTM Operations

In order to further relax the computational costs in the algorithm level, we also present novel approximate computing of LSTM cell when the LSTM inputs are relatively similar but not enough to disable their cell operations. In this work, we define the additional threshold θ_{p} to activate the approximate computing rather than the original LSTM cell when $\theta_{\mathrm{p}} < S_t < \theta_{\mathrm{s}}$. To realize the approximate LSTM operations, more precisely, we adopt the concept of the delta-RNN scheme from [12], which calculates the difference vector between two consecutive inputs followed by the clipping operation, increasing the sparsity of matrix multiplications. As we only apply the delta computing for marginally-similar cases, it is shown in [14] that the LSTM cells activating this pseudo-skipping mode avoid operations from the inputs having large differences naturally. When the pseudo-skipping cases are detected, therefore, we can allow more aggressive approximation on the LSTM operations by reducing the computing resolution compared to the normal LSTM operations. In other words, as illustrated in Figure 4c, the time positions with large input changes automatically activate the original LSTM cells with high-resolution MAC operators, and the pseudo-skipping scenario is only used for handling the small number of changes between two inputs by exploiting the reduced computing precision. Combined with the cell-skipping method described in the prior subsection, the algorithm-level optimization can reduce the number of MAC operations of DNN-based speech recognition by 49% when compared to the original DeepSpeech processing, relaxing the computational complexity accordingly while maintaining the WER performance. To develop the hardware accelerator, we reduce half of the original computing resolution for approximate LSTM operations, allowing the high utilization of hardware resources.

4. Accelerator Design for Approximate LSTM Processing

Figure 5 shows the conceptual architecture of the proposed energy-efficient LSTM accelerator, including an external memory interface for receiving input vectors and weight matrices, a similarity check module for checking S_t of incoming input vectors, non-linear operators for computing activation functions, an LSTM engine including 32 parallel processing elements (PEs), and on-chip memories for storing input/output feature maps. When a 4096-sized 16-bit quantized input vector is transferred from the external memory, it is first stored in on-chip SRAM buffers. In order to provide a sufficient bandwidth required by the LSTM operations, the on-chip memory is configured by using eight 64-entry dual-port SRAM banks. Note that total n vectors are serially accessed from the external memory and issued to the LSTM engine for handling a sentence, which depends on the length of the recorded speech signal. To support the similarity-based LSTM cell operations, it is required to keep the prior input vector for calculating the similarity score as well as the sparse vector, as illustrated in Figure 4. The addresses of on-chip buffers are managed in a ping-pong manner, efficiently supporting the proposed LSTM processing without consuming the additional energy caused by vector movements.

The similarity check module then easily accesses two most recent vectors to compute S_t and selects the current processing mode among three candidates, i.e., the normal LSTM mode, the approximate LSTM mode, and the skipping mode. Two thresholds θ_{s} and θ_{p} can be initialized at the run time based on numerous simulations as described in [14]. The LSTM engine performs the dedicated operations according to the selected processing mode. Note that the internal MAC processing arrays are totally disabled when we observe the highly similar inputs satisfying $S_t > \theta_{\mathrm{s}}$ whereas the large amount changes of input vectors ($S_t < \theta_{\mathrm{p}}$) allow the fully-activated LSTM engine. As shown in Figure 5, the LSTM engine mainly consists of 32 PEs, each of which includes a MAC operator. Hence, 32 MAC operations are processed in parallel for activating the normal LSTM cell operations. The memory controller efficiently manages data flows between the LSTM engine and numerous dual-port on-chip buffers, maintaining the hardware utilization to be high enough.

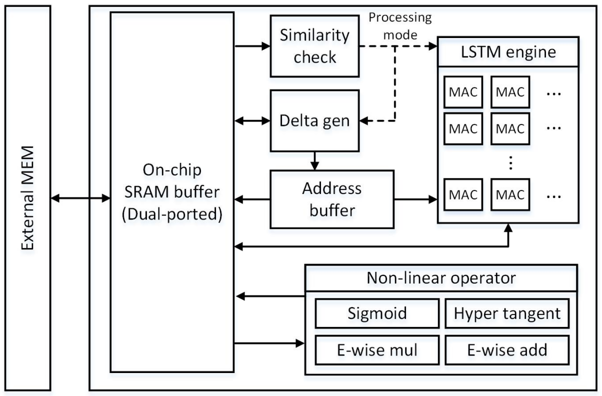

Figure 5. Proposed LSTM accelerator architecture.

For the case of approximate mode, i.e., $\theta_p < S_t < \theta_s$, the LSTM engine supports the approximate operations with the reduced computing resolution. More specifically, the original MAC operations performed at each PE are based on 16 $\times$ 16 multiplications, whereas the approximate mode requires 8 $\times$ 8 multiplications. In order not to degrade the energy efficiency, in this work, the MAC operator is designed to support two multiplications in parallel when the approximate mode is selected.

Figure 6 is an overall structure of the LSTM engine of the proposed accelerator, which can support various computing resolutions required at the proposed approximate LSTM computing method. As shown in the figure, the proposed LSTM engine internally includes a total of 32 processing elements (PEs). By utilizing dual-port on-chip memories, we can perform the read and write accesses simultaneously, improving the energy efficiency of the overall LSTM accelerator by reducing the number of inactive PEs. In fact, each processing element is nothing but a MAC operator, which flexibly supports two different computing resolutions according to the LSTM processing mode delivered from the similarity check module. By supporting the optimal computing precision depending on the current input vector, we can additionally enhance energy efficiency.

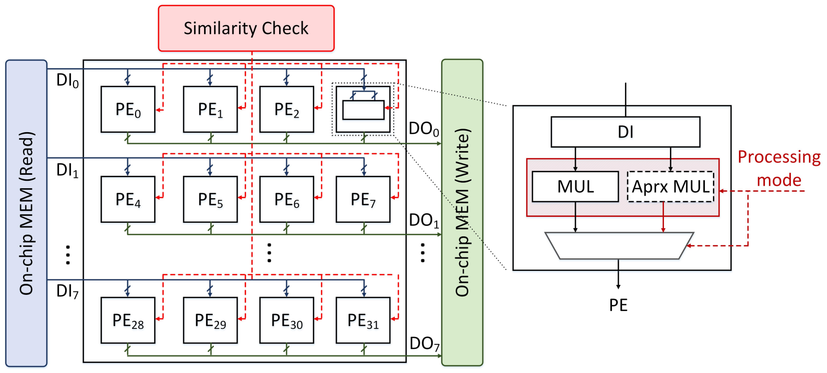

Figure 6. Overall architecture of LSTM engine providing multiple operation modes.

Figure 7 details the internal architecture of each PE, where the connections of multiplication steps are conditionally modified on demand. More precisely, a PE is first initialized by reading 16 bits from the on-chip SRAM and divides the data into two W registers, each of which contains eight bits. When we perform the normal LSTM cell mode, two W registers together denote a weight value, whereas the approximate mode handles two 8-bit weight values separately. Whenever a PE receives the data input (DI) in order to perform the normal LSTM cell operation, which is an element of the current input vector, note that we can turn on two internal 8 $\times$ 8 multiplications followed by a summation unit to support a 16 $\times$ 16 multiplication.

Otherwise, for the case of approximate LSTM mode, we disable the red-colored modules in Figure 7, and then only the leftmost/rightmost 8×8 multipliers are involved in the cell operation, supporting two 8×8 multiplications used for the delta-based LSTM operation with reduced computing precision. Note that the delta generator in Figure 5 stores the approximate input values into on-chip memories, reducing the unnecessary memory accesses associated with the high-resolution values. As a result, the proposed LSTM engine structure offers an attractive energy efficiency by eliminating all the operations for the skipped LSTM cells and also by reducing the number of processing cycles for the approximate computing cases.

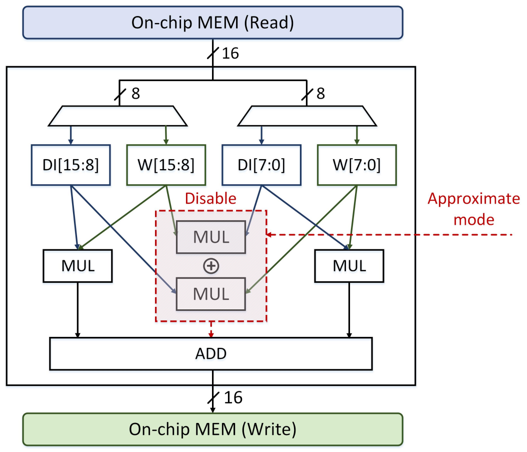

Figure 7. Detailed PE architecture.

As we adopt the delta-based processing from [20] to develop the processing scenario of approximate mode, some elements of input vector are changed to zero values before starting the LSTM cell operation, increasing the sparsity of matrix-to-matrix multiplications for gate operations shown in (1)~(3). If we directly use the same sequential memory accessing for pseudo-skipping LSTM operations as what we apply to the normal mode, as illustrated in Figure 8, the increased sparsity results the additional redundant on-chip memory accesses for taking meaningless weight parameters. To solve this problem, in this work, we newly introduce a small-sized on-chip SRAM to keep the accessing sequence of **W** matrices. This address buffer is initialized at the run time by marking the nonzero element positions whenever the delta generator module in Figure 5 produces the survived input elements with reduced resolution. As we only apply the delta processing for the partially similar inputs, the number of survived elements are naturally less than that of the original delta-RNN processing [20], allowing to reduce the size of this additional SRAM. For the prototype design, we only utilize the address buffer of 64 entries for the prototype design, minimizing the additional hardware complexity. By accessing only meaningful columns of **W**, as a result, the proposed accelerator maximizes the energy efficiency even for supporting the approximate LSTM operations.

After performing the MAC-oriented operations at the LSTM engine, the proposed accelerator enables the non-linear operators such as sigmoid and hyperbolic tangent functions. We realize these non-linear operators by adopting a number of look-up tables, which are basically obtained by piece-wise linear approximations. It is also possible to re-utilize internal MAC operations at the LSTM engine for performing five FC layers; the first three for the pre-processing and the rest two for the post-step as depicted in Figure 2. As a result, the proposed accelerator design shown in Figure 5 successfully supports all the processing sequences defined by the state-of-the-art DeepSpeech network while even allowing multiple LSTM operating modes to save energy consumption by eliminating redundant processing parts as many as possible.

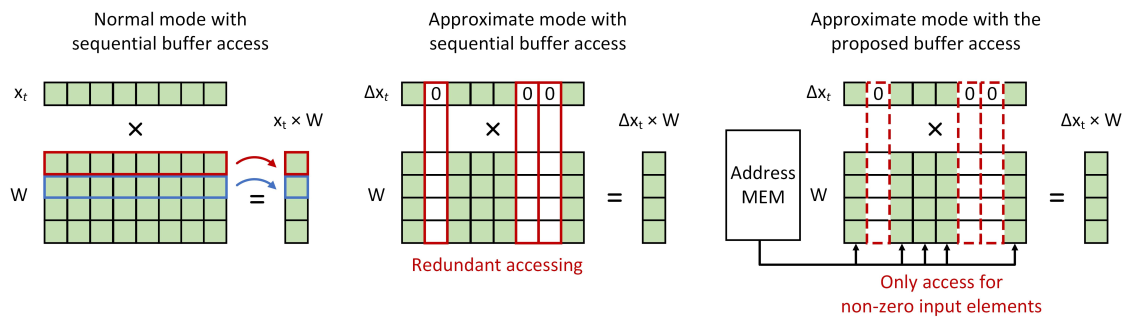
Normal mode with sequential buffer access
Approximate mode with sequential buffer access
Approximate mode with the proposed buffer access
x_t
W
$x_t \times W$
Δx_t
$\Delta x_t \times W$
Redundant accessing
Address MEM
Only access for non-zero input elements

Figure 8. Sparsity-aware buffer accessing method for approximate LSTM processing.

5. Simulation and Implementation Results

5.1. Algorithm-Level Performance

In order to show the algorithm-level innovations from the proposed similarity-based LSTM operations, we simulated various processing scenarios using the DeepSpeech network on the LibriSpeech dataset [14], which are evaluated based on the tensorflow framework [21] to realize the original LSTM processing [15], the delta-based computation [12], and the proposed multi-mode approach. To evaluate the computational costs quantitatively, for each LSTM processing method, we measured the number of active MAC operations, and the number of external memory accesses, which are known to be the two most energy-consuming workload in LSTM operations [22]. For the cost-reduction techniques, we allowed a 1% accuracy drop from the baseline WER.

Figure 9 illustrates the simulation results of different LSTM processing approaches, where the number of MAC operations and the number of memory accesses are normalized to those of the baseline bi-directional LSTM processing [15], which fully utilizes LSTM cells for achieving the most accurate WER result. When compared to the baseline network, more precisely, we can save the number of external memory accesses and the number of MAC operations by 55% and 49%, respectively. Note that our algorithm is also superior to the prior delta-based LSTM processing, as depicted in Figure 9. Therefore, the proposed similarity-based LSTM processing provides the most cost-efficient way to handle speech recognition by dynamically adjusting the processing mode.

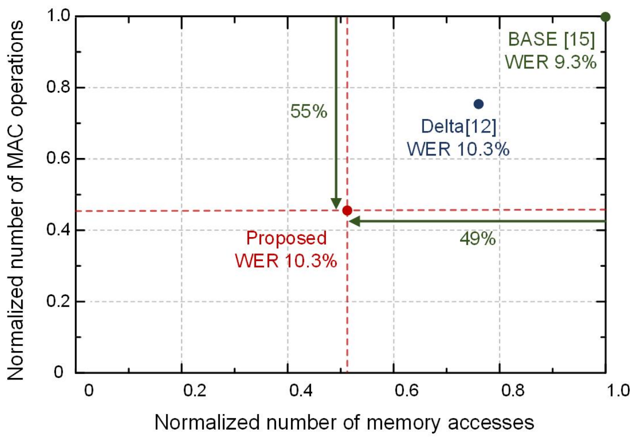

Figure 9. Computing cost reduction by the proposed algorithm-level optimization.

5.2. Prototype Implementation

To validate the proposed idea, we realized the prototype LSTM accelerator, whose internal architecture is depicted in Figure 5. For hardware analysis, we perform the post-layout simulation of the prototype LSTM accelerator, which is designed in a 65nm CMOS technology. For synthesizing the prototype, more precisely, we adopted Synopsys Design Compiler [23], whereas the place and route steps were based on Synopsys IC compiler [24]. Figure 10a depicts the layout result of the proposed approximate LSTM accelerator, occupying 3.37 mm^2, where the area breakdown is summarized in Table 1. From the post-layout simulation, the prototype decoder can operate at the speed of 100 MHz, consuming only 33 mW by adopting the various hardware-level optimization methods. Figure 10b shows how the proposed schemes gradually improve the energy efficiency of the LSTM-based end-to-end speech recognition, where the energy efficiency is computed by dividing the maximum throughput of the proposed LSTM processing scheme by the corresponding power consumption, which is widely used to present the allowed number of operations for the given power budget [25].

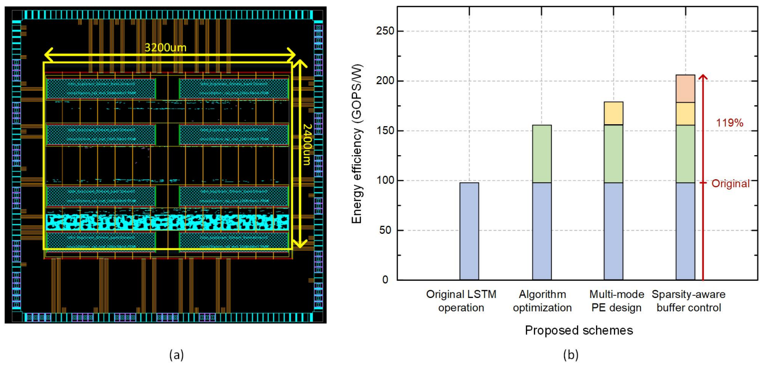

Figure 10. Implementation of the prototype LSTM accelerator. (**a**) Layout result, (**b**) Improvement of energy efficiency).

Table 1. Area breakdown of the prototype LSTM accelerator

Total Area (μm)	Similarity Check (μm)	Delta Gen (μm)	Non-Linear Operations (μm)	LSTM Engine (μm)	SRAM (μm)	Others (μm)
3,367,650	13,830	37,703	522,732	140,629	2,639,526	13,230

For the same processing architecture, note that the original bi-directional LSTM processing provides less than 100 GOPS/W. By introducing the algorithm-level innovations, combining the similarity-based cell skipping and approximation, we can enhance the energy-efficiency up to 160 GOPS/W. As the approximate LSTM operation reduces the required computing precision, as depicted in Figure 6, our accelerator architecture presents the flexible MAC architecture supporting different resolutions, further improving the energy-efficiency by 15.4% compared to the naive implementation of the proposed algorithm-level optimization. Finally, the sparsity-aware memory accessing achieves the most energy-efficient processing by eliminating the unnecessary on-chip buffer requests, increasing the energy-efficiency by 119% and 31% when compared to the original LSTM operations and the naive version, respectively. As a result, the proposed work provides the energy-efficient LSTM processing scenario by exploiting the co-optimization approaches between the algorithm and hardware levels, leading to the on-device end-to-end speech recognition.

However, depending on the applications, there might be some limitations for directly applying the proposed approximate LSTM computing, which is dedicated to reducing the computational cost of bi-directional LSTM operations used for the recent DeepSpeech-based speech recognition systems.

When we consider the streaming applications for the real-time system [26], for example, we have to exploit a one-directional LSTM structure associated with the attention-based computations, and only a few LSTM cells in the attention region are involved in identifying the output characters. In this case, the proposed similarity-based LSTM cell approximation may severely degrade the recognition accuracy. Extending the proposed approximate LSTM operation to the attention-based processing could be solved by introducing more advanced similarity-checking criteria, which would be one of the future research directions for further improving the energy efficiency of the state-of-the-art DNN-based speech recognition systems.

6. Conclusions

In this paper, we have presented an approximate computing method to reduce the processing cost for LSTM operations used for the speech recognition systems. Based on our previous algorithm-level works, the proposed method measures the importance of each LSTM cell at the run time and then deactivates the redundant LSTM cells by the pre-defined similarity score. In addition, we newly define slightly-similar cells to activate the pseudo-skip operations, which are delta-based approximate LSTM calculations with reduced resolution. Therefore, we can reduce the energy consumption without compromising accuracy by mitigating the computational complexity only for less important cells. The dedicated hardware architecture to support the proposed algorithm-level innovations is carefully designed to further minimize the processing costs by newly introducing the multi-mode MAC operators as well as the sparsity-aware buffer control scheme. As a result, the prototype design in 65 nm CMOS technology presents a significant reduction in terms of processing complexity, improving the energy-efficiency by more than two times than the baseline architecture.

Author Contributions: J.J. and Y.L. worked together during the whole editorial process of the manuscript. J.J., J.K. and Y.L. reviewed, edited and finalized the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program (IITP-2020-2020-0-01612) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation), was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No.2020M3H6A1085498 and 2019R1F1A1060767), and was supported by the Sports Promotion Fund of Seoul Olympic Sports Promotion Foundation from Ministry of Culture, Sports and Tourism.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gao, F.; Huang, T.; Wang, J.; Sun, J.; Hussain, A.; Zhou, H. A novel multi-input bidirectional LSTM and HMM based approach for target recognition from multi-domain radar range profiles. *Electronics* **2019**, *8*, 535. [CrossRef]
2. Kang, S.I.; Lee, S. Improvement of Speech/Music Classification for 3GPP EVS Based on LSTM. *Symmetry* **2018**, *10*, 605. [CrossRef]
3. Kumar, J.; Goomer, R.; Singh, A.K. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Comput. Sci.* **2018**, *125*, 676–682. [CrossRef]
4. Kadetotad, D.; Berisha, V.; Chakrabarti, C.; Seo, J.S. A 8.93-TOPS/W LSTM recurrent neural network accelerator featuring hierarchical coarse-grain sparsity with all parameters stored on-chip. In Proceedings of the ESSCIRC 2019-IEEE 45th European Solid State Circuits Conference (ESSCIRC), Cracow, Poland, 23–26 September 2019; pp. 119–122.
5. Wang, M.; Wang, Z.; Lu, J.; Lin, J.; Wang, Z. E-lstm: An efficient hardware architecture for long short-term memory. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 280–291. [CrossRef]
6. Kung, J.; Kim, D.; Mukhopadhyay, S. Dynamic approximation with feedback control for energy-efficient recurrent neural network hardware. In Proceedings of the 2016 International Symposium on Low Power Electronics and Design, San Francisco, CA, USA, 8–10 August 2016; pp. 168–173.

7. Byun, Y.; Ha, M.; Kim, J.; Lee, S.; Lee, Y. Low-complexity dynamic channel scaling of noise-resilient CNN for intelligent edge devices. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 114–119.
8. Campos, V.; Jou, B.; Giró-i Nieto, X.; Torres, J.; Chang, S.F. Skip rnn: Learning to skip state updates in recurrent neural networks. *arXiv* **2017**, arXiv:1708.06834 .
9. Moon, S.; Byun, Y.; Park, J.; Lee, S.; Lee, Y. Memory-Reduced Network Stacking for Edge-Level CNN Architecture With Structured Weight Pruning. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2019**, *9*, 735–746. [CrossRef]
10. Neil, D.; Pfeiffer, M.; Liu, S.C. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3882–3890.
11. Dai, R.; Li, L.; Yu, W. Fast training and model compression of gated RNNs via singular value decomposition. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–7.
12. Neil, D.; Lee, J.H.; Delbruck, T.; Liu, S.C. Delta networks for optimized recurrent network computation. In Proceedings of the International Conference on Machine Learning (PMLR), Sydney, Australia, 11–15 August 2017; pp. 2584–2593.
13. Andri, R.; Cavigelli, L.; Rossi, D.; Benini, L. YodaNN: An ultra-low power convolutional neural network accelerator based on binary weights. In Proceedings of the 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Pittsburgh, PA, USA, 11–13 July 2016; pp. 236–241.
14. Jo, J.; Kung, J.; Lee, S.; Lee, Y. Similarity-Based LSTM Architecture for Energy-Efficient Edge-Level Speech Recognition. In Proceedings of the 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), Lausanne, Switzerland, 29–31 July 2019; pp. 1–6.
15. Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv* **2014**, arXiv:1412.5567.
16. Mirghafori, N.; Morgan, N. Combining connectionist multi-band and full-band probability streams for speech recognition of natural numbers. In Proceedings of the Fifth International Conference on Spoken Language Processing, Sydney, Australia, 30–4 December 1998.
17. Nguyen, T.S.; Stüker, S.; Niehues, J.; Waibel, A. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. In Proceedings of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, 4–8 May 2020; pp. 7689–7693.
18. Miao, Y.; Gowayyed, M.; Metze, F. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In Proceedings of the 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Scottsdale, AZ, USA , 13–17 December 2015; pp. 167–174.
19. Panayotov, V.; Chen, G.; Povey, D.; Khudanpur, S. Librispeech: an asr corpus based on public domain audio books. In Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, 19–24 April 2015; pp. 5206–5210.
20. Gao, C.; Neil, D.; Ceolini, E.; Liu, S.C.; Delbruck, T. DeltaRNN: A power-efficient recurrent neural network accelerator. In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, 25–27 February 2018; pp. 21–30.
21. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX symposium on operating systems design and implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
22. Silfa, F.; Dot, G.; Arnau, J.M.; Gonzàlez, A. E-PUR: An energy-efficient processing unit for recurrent neural networks. In Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques, Limassol, Cyprus, 1–4 November 2018; pp. 1–12.
23. Dupenloup, G. Automatic Synthesis Script Generation for Synopsys Design Compiler. U.S. Patent 6,836,877, 28 December 2004.
24. Kommuru, H.B.; Mahmoodi, H. *ASIC Design Flow Tutorial Using Synopsys Tools*; Nano-Electronics & Computing Research Lab, School of Engineering, San Francisco State University: San Francisco, CA, USA, 2009.

25. Moon, S.; Lee, H.; Byun, Y.; Park, J.; Joe, J.; Hwang, S.; Lee, S.; Lee, Y. FPGA-based sparsity-aware CNN accelerator for noise-resilient edge-level image recognition. In Proceedings of the 2019 IEEE Asian Solid-State Circuits Conference (A-SSCC), Macao, China 4–6 November 2019; pp. 205–208.
26. Jorge, J.; Giménez, A.; Iranzo-Sánchez, J.; Civera, J.; Sanchis, A.; Juan, A. Real-Time One-Pass Decoder for Speech Recognition Using LSTM Language Models. In Proceedings of the 20th Annual Conference of the International Speech Communicatoin Association(INTERSPEECH), Graz, Austria, 15–19 September 2019; pp.3820–3824.