2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS) | 979-8-3503-2649-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICECS58634.2023.10382787

# Lookupx: Next-Generation Quantization and Lookup Techniques for Empowering Performance and Energy Efficiency

Cagla Irmak RUMELILI KOKSAL[1,2] Nihat Mert CICEK[1,2] Ayse YILMAZER METIN[2] Berna ORS[2]
[1]*Aselsan Inc., Ankara, Turkiye*
irumelili,nmcicek@aselsan.com.tr
[2]*Istanbul Technical University, Istanbul, Türkiye*
yilmazerayse,orssi@itu.edu.tr

***Abstract*—Long Short Term Memory (LSTM) networks as one of the most used Recurrent Neural Networks (RNN) structures offer high accuracy for sequence learning tasks. However, it is challenging to offer low latency and high throughput while satisfying the low power constraints at the same time for computationally expensive LSTM operations. This work offers a two-pronged approach to accelerate inference in RNN networks. First, linear quantization technique is applied to reduce the complexity of operations, power consumption and required memory resources. Then, a new activation implementation method is proposed, called lookupx, to accelerate sigmoid function computation during inference. It is shown that lowering precision to 4-bit integer numbers for inputs causes only 2% accuracy loss and the lookupx activation methodology has 1.9x better performance and 50x lower power consumption while decreasing the required chip area 1.2x compared to integer domain activation functions with the same accuracy result.**

***Index Terms*—quantization, nonlinear activation functions, RNN, LSTM, accelerator, low power**

## I. Introduction

In recent years, Recurrent Neural Networks (RNN) have been remarkably successful in applications such as automatic speech recognition [16], sentiment analysis [21], machine translation [18] and so on. The computations of the network are done both on cloud servers [6] and low power mobile devices [7]. To reduce the total execution time and power consumption of the computations, neural networks are usually processed in specialized hardware architectures, namely accelerators.

Long Short Term Memory (LSTM) architecture [12] is a widely preferred architecture for RNN implementation. A high performance and low power RNN implementation is hard due to time consuming matrix operations and complex nonlinear operations. Quantization stands as an extremely effective technique for enhancing matrix operations. It offers superior performance and reduced power consumption by enabling less hardware resources, minimizing operation complexity, and facilitating faster memory access. Since quantization alters the input set of the nonlinear function, piecewise linear approximation and lookup table methods are commonly employed. In this paper, we focus on quantization and nonlinear function implementation. Our contributions are as follows:

- We investigate the impact of aggressive low-precision representations of weights and inputs. Although the input and weight values are quantized to 4-bit and 5-bit integer values respectively, the accuracy changed only 2%.
- We propose a new algorithm-hardware co-design to leverage sigmoid and tanh computation. Our evaluation demonstrates that this method offers either better accuracy or better time, area, and energy efficient implementation compared to state of the art integer domain non linear activation function approximation techniques [4].

The remainder of the paper is organized as follows. Section II provides basic concepts of LSTM as well as a review of prior work on LSTM accelerators. The proposed architecture is presented in Section III. Section IV provides the details of the experimental setup and results and finally, the paper is concluded in Section V with final remarks.

## II. Overview & Related Work

Standard RNN cells, also known as vanilla RNNs, have no choice but eventually to forget due to the vanishing gradient problem [3]. LSTM is the RNN architecture offered to overcome this problem [13]. In this section, first, the internal structure of the LSTM network is described. Then, prior work dealing with the LSTM implementations that focuses on quantization algorithms and activation functions is briefly reviewed.

### A. Long Short Term Memory

An LSTM cell is composed of four units commonly called gates. These four gates are named as forget gate, input gate, cell gate and output gate [12]. Each of these gates is responsible for the flow of information in the LSTM. In other words, gates decide how much information will be forgotten from the previous input and how much information will be contributed from the new data.

The mathematical expressions for each gate and the output of the LSTM cell ($h_t$) can be seen in (1), (2), (4), (3), (5) and (6). $f_t$, $i_t$, $c_t$, $o_t$ represent the output vectors of forget gate, input gate, cell gate, and output gate at time $t$ respectively. $x_t$ is the input vector for the current step, whereas $h_{t-1}$ is

the hidden layer output vector from the previous step. In other words, $x_t$ is used for forward connections, while $h_{t-1}$ includes feedback information from the previous layers. In each gate, $W_{*x}$, $W_{*h}$ and $b_*$ terms represent the weight matrix for the input, weight matrix for hidden layer output, and bias of the gate, respectively (e.g. $W_{fx}$ is the input weight matrix of the forget gate). Hyperbolic tangent (tanh) and sigmoid ($\sigma$) functions are the activation functions that are used in LSTM cells for activation. $\odot$ represents pair-wise multiplication [8]. The huge matrix multiplications and activation functions are the main cause of the power consumption in LSTM cells.

$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right) \quad (1)$$
$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right) \quad (2)$$
$$\tilde{c}_t = tanh\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right) \quad (3)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (4)$$
$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (5)$$
$$h_t = o_t \odot tanh(c_t) \quad (6)$$

### B. Quantization

Reducing the precision during the inference is one of the common methods simplifying LSTM operations to minimize power consumption and cost of the hardware, while improving the throughput and latency values [22], [17], [6], [15], [10], [5]. 16-bit floating point, fixed point and integer quantization are the common techniques for accelerating LSTM networks. For example, when 8-bit fixed point representation is used instead of 32-bit floating point, 3.3x less power is consumed for addition operations while 15.5x less power is consumed for multiplication operations [19]. Similarly, 8-bit integer is sufficient for inference of several networks in terms of accuracy and it offers 6x reduction of power consumption compared to 16-bit floating-point multiplication [14].

Another important outcome of lowering the precision is reducing the memory footprint, hence, total on-chip/off-chip memory size and total number of accesses. This reduction is significant since accessing on-chip and off-chip memory is considered as the main actor for dynamic power consumption [11], throughput and latency [17].

### C. Activation Function

Sigmoid (7) and hyperbolic tangent (8) functions are used as activation functions in LSTM cells. Both of these functions are computationally expensive due to exponentiation and division calculations. Further performance improvement could be achieved via accelerating these functions in RNN accelerators. There are two main approaches to reduce the function complexity in artificial neural networks: Piecewise linear approximation [5], [9], [20] and lookup table [10].

$$sigmoid(x) = 1/(1 + e^{-x}) \quad (7)$$
$$tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \quad (8)$$

## III. PROPOSED ARCHITECTURE

We conducted an experiment and found that an LSTM cell on an Nvidia K80 GPU takes approximately one-third of the time to calculate the sigmoid and tanh functions during inference. This indicates that accelerating the activation function has a significant impact on LSTM inference time.

In this work, we offer a fast and compact implementation called lookupx as it is formulated in (9). The proposed technique approximates the activation function, works in quantized integer domain, and eliminates the extra hardware resources for the dequantization/quantization process on the fly. In this method, the summation of shifted input and an offset from lookup table is used for activation function calculation. The offset is calculated as the average of the difference between the activation function and the shifted input for the input dataset, and it is referred to as $offset_x$. The lookup table is addressed via input bits. By using the input as a primary source of the function the required lookup entry size is drastically reduced. Fig. 1 shows the lookup table with 8 entries, lookupx table with 8 entries and the scaled sigmoid functions for (-512, 512) input domain. Note that, (-512, 512) input domain is observed for the activation function after linear quantization as it is explained in IV-B. The accuracy of lookupx method is closer to scaled sigmoid compared to the same size lookup table as it is seen in Fig. 1.

$$lookupx(x) = x/4 + \text{offset}_x \quad (9)$$

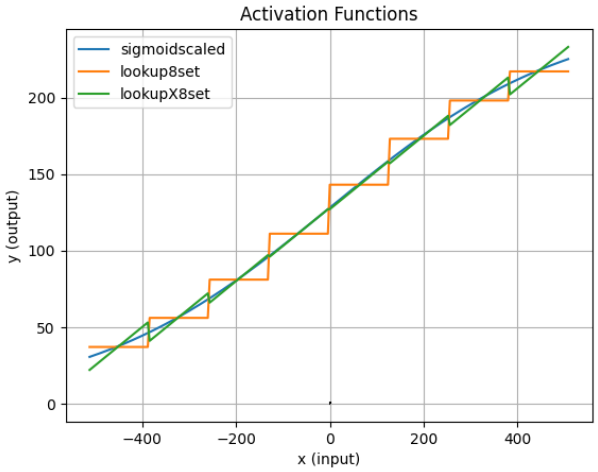


Fig. 1. Lookup table, sigmoid function and proposed lookupx

## IV. EVALUATION

### A. Reference Model

Internet Movie Database (IMDb) sentiment dataset [2] is used to evaluate the proposed quantization and activation techniques. This dataset includes 50000 movie reviews taken from IMDb website [1] and a sentiment label is attached to each review.

This dataset is trained with 8 cell LSTM network and the training results are collected in terms of precision, recall, F1-score and accuracy. These results are shown in Table I. For the rest of the paper, these results are used as a reference point and further discussions are made accordingly.

TABLE I
TEST RESULTS

| | Accuracy | Presicion | Recall | F1-score | # of reviews |
|---|---|---|---|---|---|
| Negative | | 0.86 | 0.89 | 0.88 | 12500 |
| Positive | | 0.89 | 0.86 | 0.87 | 12500 |
| All | 0.88 | | | | 25000 |

### B. Linear Quantization

In this work, we used uniform quantization [19] to represent 32-bit floating numbers as integer numbers and investigate the impact of aggressive low-precision representations of weights and inputs. Both input and weight are quantized with $scale$ while the bias parameter is quantized with $scale^2$.

We calculate the prediction results for several $scale$ values for IMDb dataset and the results are illustrated in Fig. 2. Note that "p_" and "n_" prefixes are used for positive and negative sentiment reviews, respectively. When the $scale$ equals 16, the input, weight and bias in the IMDb dataset are represented with 4-bit, 5-bit and 10-bit integers, respectively, resulting in an accuracy of 86%. By utilizing lower precision, the memory footprint is reduced by up to 8 times compared to the floating-point representation, with only a 2% loss in accuracy.

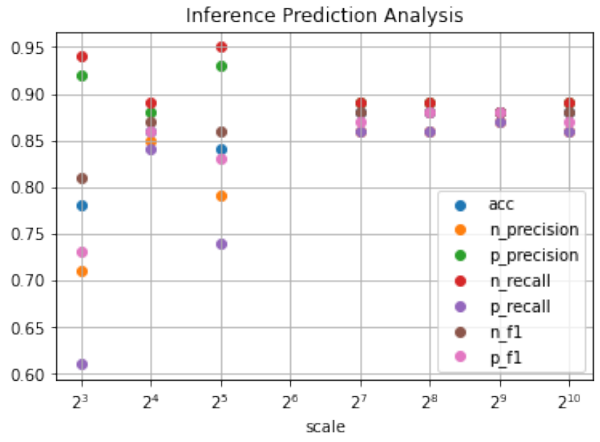


Fig. 2. Inference prediction accuracy vs quantization scale parameter

### C. Hardware Implementation

Lookup table and piecewise linear approximation techniques that accelerate activation functions are implemented and experimented with FPGA [10], [5], [9] and with ASIC [20] under TSMC 90nm technology node. However, the experimental results for the aforementioned accelerators are not given specifically for the activation layer. In this work, we also implemented these techniques in hardware to compare with the lookupx method.

*1) Piecewise Linear Approximation:* The method in [4] is modified for our network and IMDb dataset as follows: The gradient of the lines is selected such that the implementation of the function can be represented via shift operations. Although PLAN [4] offers a piecewise linear approximation method for only the sigmoid function, we approximate the hyperbolic tangent function in a similar way to reduce the hardware complexity. The resulting equations are illustrated in Table II and Table III and these functions are implemented via simple logic operations and comparators.

TABLE II
SIGMOID PIECEWISE LINEAR FUNCTION

| Condition | Function |
|---|---|
| $x < -2$ | 0 |
| $-2 < x < -0.94$ | x/8 + 0.62 |
| $-0.94 < x < 0.94$ | x/4 + 0.5 |
| $0.94 < x < 2$ | x/8 + 0.38 |
| $2 < x$ | 1 |

TABLE III
HYPERBOLIC TANGENT PIECEWISE LINEAR FUNCTIONS

| Condition | Function |
|---|---|
| $x < -2$ | -1 |
| $-2 < x < 2$ | x |
| $x > 2$ | 1 |

*2) Lookup Table:* When dealing with a quantized dataset, the size of the input set determines the sizes of the table entries. In this study, both the input and weights are scaled using a factor of 16, which means the input of the activation functions is scaled by a factor of 256. For the IMDb dataset, the floating-point input range of [-2, 2] is transformed into a quantized range of [-512, 512], resulting in a table size of 1024. Hence, an SRAM with 1024 entries is implemented to achieve full accuracy. Additionally, a lookup table with the same size as the lookupx method (8 entries) is implemented by averaging every 128 items from the larger table of 1024 entries, enabling a comparison of accuracy between the proposed lookupx method and lookup table method.

*3) Lookupx:* The division in (9) is implemented via shift operation. 8 different offset values are stored in registers. These registers are addressed via the most significant bits of the $x$.

### D. Results

*1) Accuracy Results:* Our 8-entry lookupx method is compared with the piecewise linear approximation method, 8-entry averaged lookup table, and 1024-entry lookup table. All these techniques use the IMDb dataset and linear quantization ($scale$=$2^4$) for inference. As it is seen in the results listed in Table IV, the lookupx technique has the same accuracy as the original activation function and offers 3% better accuracy compared to state-of-the-art approximation methods, namely the piecewise linear approximation and the same-sized lookup table. Lookupx method achieves the same level of accuracy as a 1024 entry lookup table.

TABLE IV
LINEAR ACTIVATION FUNCTION ACCURACY EVALUATION

| Method | Accuracy |
|---|---|
| piecewise linear approximation | 0.83 |
| lookup (8 entries) | 0.83 |
| lookup (1024 entries) | 0.86 |
| lookupx (8 entries) | 0.86 |

*2) Synthesis Results:* We synthesized piecewise linear approximation, lookup table and the proposed lookupx logic using Synopsys' Design Compiler under TSMC 28nm technology. For lookup table method, TSMC 28nm memory compiler is used to generate 1024-entry SRAM macros. Since lookupx method is sufficient with a smaller number of entries, registers are used for hardware implementation. Table V lists the time, area, and power results for each method.

As it is seen in Table V and Table IV the lookupx activation methodology has 1.9x better performance and 50x lower power consumption while decreasing the required chip area 1.2x compared to integer domain activation functions with the same accuracy result.

TABLE V
LINEAR ACTIVATION FUNCTION SYNTHESIS RESULTS

| Method | Period (ns) | Area ($um^2$) | Power (mW) |
|---|---|---|---|
| piecewise linear app. | 0.75 | 90 | 0.026 |
| lookup | 0.58 | 220 | 2,9 |
| lookupx | 0.3 | 180 | 0.058 |

## V. CONCLUSION

The LSTM accelerator aims to reduce power consumption by utilizing low-precision representations of numbers and employing simplified hardware. In this paper, to the best of our knowledge, we present the first 4-bit integer representation for the LSTM cell while having an accuracy loss smaller than $2\%$ without any post-training optimization. Additionally, our proposed lookupx method enables the development of power, area, and time-optimized hardware for activation functions, while maintaining high accuracy levels. It is important to note that the algorithm we propose can be extended to other neural network architectures that utilize non-linear activation functions.

## REFERENCES


[1] IMDb. https://www.imdb.com/. Accessed: 2021-06-17.

[2] IMDb sentiment classification. https://github.com/tensorflow/models/tree/master/research/adversarial_text. Accessed: 2021-06-17.

[3] Lecture10: Recurrent neural networks. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf. Accessed: 2021-07-17.

[4] H. Amin, K. Mervyn Curtis, and Barrie R Hayes-Gill. Piecewise linear approximation applied to nonlinear function of a neural network. *In Circuits, Devices and Systems, IEE Proceedings*, December 1997.

[5] E. Azari and S. Vrudhula. An energy-efficient reconfigurable lstm accelerator for natural language processing. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4450–4459, 2019.

[6] Jeremy Fowers et al. A configurable cloud-scale dnn processor for real-time ai. In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–14, 2018.

[7] Chang Gao, Antonio Rios-Navarro, Xi Chen, Shih-Chii Liu, and Tobi Delbruck. Edgedrnn: Recurrent neural network accelerator for edge inference. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(4):419–432, 2020.

[8] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[9] Yijin Guan, Zhihang Yuan, Guangyu Sun, and Jason Cong. Fpga-based accelerator for long short-term memory recurrent neural networks. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 629–634, 2017.

[10] 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. Ese: Efficient speech recognition engine with sparse lstm on fpga, 2017.

[11] John L. Hennessy and David A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[12] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

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

[14] N. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017.

[15] Liu Liu et al. Duet: Boosting deep neural network efficiency on dual-module architecture. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 738–750, 2020.

[16] Yajie Miao, Mohammad Gowayyed, and Florian Metze. EESEN: end-to-end speech recognition using deep RNN models and wfst-based decoding. *CoRR*, abs/1507.08240, 2015.

[17] Franyell Silfa, Gem Dot, Jose-Maria Arnau, and Antonio Gonzàlez. E-pur. *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*, Nov 2018.

[18] Shashi Pal Singh, Ajai Kumar, Hemant Darbari, Lenali Singh, Anshika Rastogi, and Shikha Jain. Machine translation using deep learning: An overview. In *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pages 162–167, 2017.

[19] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

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

[21] Guixian Xu, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, and Xu Wu. Sentiment analysis of comment texts based on bilstm. *IEEE Access*, 7:51522–51532, 2019.

[22] Reza Yazdani, Olatunji Ruwase, Minjia Zhang, Yuxiong He, Jose-Maria Arnau, and Antonio Gonzalez. Lstm-sharp: An adaptable, energy-efficient hardware accelerator for long short-term memory, 2019.