Small Area Configurable Deep Neural Network Accelerator for IoT System

Liangkai Zhao, Ning Wu, Fen Ge, Fang Zhou, Jiahui Zhang, Tong Lu
College of Electronic and Information Engineering
Nanjing University of Aeronautics and Astronautics
Nanjing, China
e-mail: zhao_lkk@nuaa.edu.cn

***Abstract*—With the development of Internet of things (IoT) technology, deep learning algorithm has been widely used in IoT devices. Convolutional neural network (CNN) and recurrent neural network (RNN) play a significant role in image field and sequence data respectively. In order to enable the IoT terminal SoC with limited computing power and resources to support CNN and RNN algorithms, this paper proposes a small area configurable deep neural network accelerator with fixed-point accuracy. The main computing components of CNN and RNN are implemented by hardware. Each computing module completes the calculation of complex neural network through parameter configuration and combination. In order to verify the performance of the accelerator, a SoC verification system based on Cortex-M3 was constructed. The lenet-5 network and the Long Short-Term Memory (LSTM) network with 2 layers and 128 hidden units are implemented on this accelerator, and the execution time of the two networks on Intel i5 7500, Cortex-A53 and Cortex-A7 processors is compared. The comparison results show that the CNN and RNN computing power of the accelerator exceeds that of Cortex-A53 and Cortex-A7 at the main frequency of 50MHz.**

Keywords-configurable accelerator; CNN; RNN; IoT

I. INTRODUCTION

With the increasing development of Internet of things technology and artificial intelligence technology, more and more deep learning applications begin to develop and deploy on Internet of things devices [1]. Processing data on IoT devices can improve energy efficiency, reduce bandwidth usage and latency, suitable for applications with tight time and bandwidth constraints [2].

Most of the accelerated design pursues high performance, but the cost is high and the power consumption is high. It is usually used for cloud computing. The terminal equipment needs to send the collected data to the cloud for processing and then return from the cloud. Generally, there are high requirements for network quality, and can not work offline [3, 4]. In [5, 6], fixed-point number is used to design accelerator, which has the advantages of low power consumption and low resource consumption. However, its network is fixed and can only be used in specific scenarios and cannot be configured according to different scenarios. In [7, 8, 9], the method of FPGA+ARM is used to accelerate the algorithm of artificial intelligence by processor controlled accelerator, which has the advantage of flexible configuration. However, they use floating-point number or dynamic fixed-point number, which makes the accelerator a heavier burden in terms of computing performance and power consumption.

Based on the consideration of circuit area, acceleration performance and circuit flexibility, this paper proposes and designs a depth neural network accelerator with configurable fixed-point accuracy which is suitable for IoT system. Firstly, according to the characteristics of deep neural network, the basic operation unit is extracted, and four CNN acceleration chains and one LSTM acceleration chain are designed. Then, MCU is used to configure the acceleration circuit as a specific network to accelerate the specific algorithm. Therefore, the designed accelerator has the characteristics of small circuit area and high configurability.

The rest of the article is organized as follows. Section II introduces the overall design of the accelerator and the specific design of the basic acceleration unit. The Section III discusses the accelerator verification system, including the implementation of SoC based on Cortex-M3, lenet-5 network, and the LSTM with 2 layers 128 hidden units. Section IV demonstrates the acceleration performance and resource consumption of the accelerator. Finally, the conclusion and future work is furnished in Section V.

II. ACCELERATOR DESIGN

A. Accelerator Structure

CNN is a machine learning algorithm. The main operation of CNN include two-dimensional matrix convolution, nonlinear activation and pooling operation. RNN has the ability to learn from data sequences, which is the basis of real-time applications. RNN calculations provide limited data reuse, resulting in high data traffic. According to the characteristics of CNN and RNN, combined with the requirements of compact structure and low power consumption of the Internet of things system, a fixed-point precision deep neural network accelerator is designed. The overall structure of the designed unit is shown in the Fig. 1.

The acceleration module includes source data loading module, CNN parameter cache module, two ping-pong buffers, four CNN acceleration units, LSTM weight parameter cache and a LSTM acceleration unit. The CNN acceleration module consists of two-dimensional convolution, matrix addition, ReLU activation function and pooling. With these modules, four matrix convolution, matrix addition, nonlinear function and subsampling operations can be completed simultaneously. LSTM acceleration module is composed of three circuit modules. Tanh Gate and Sigmiod

978-1-7281-8141-7/20/$31.00 ©2020 IEEE

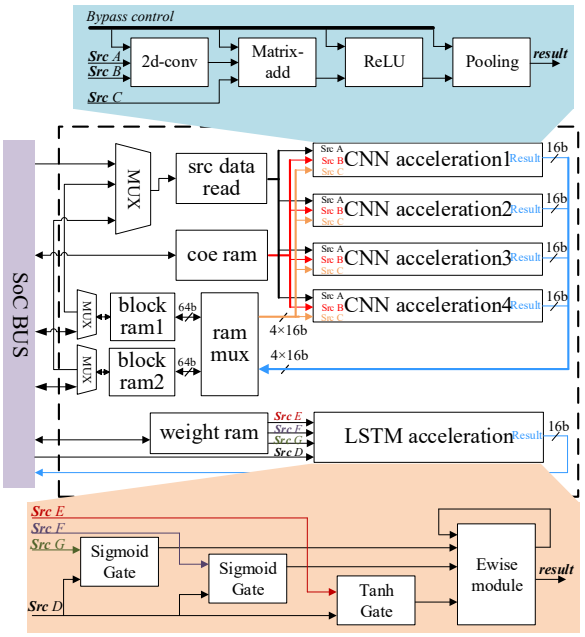

Figure 1. The architecture of the fixed-point accelerator

Gate perform matrix multiplication, vector addition and nonlinear operation of input vector and weight matrix. There are two kinds of nonlinear operation, tanh and sigmoid. The Ewised module performs the element-wise multiplication operation of vector, vector addition operation and tanh function.

B. CNN Circuit Design

In CNN algorithm and image processing algorithm, two-dimensional convolution takes up a lot of computation, accounting for more than 90% of the total network computation [11]. The optimization of convolution operation will significantly improve the resource utilization and computing performance of signal processing algorithm. The convolution operation of a two-dimensional n×n matrix X and a k×k-sized convolution kernel W can be expressed as:

$$Y_{n,m} = \sum_{i=-(k-1)/2}^{(k-1)/2} \sum_{j=-(k-1)/2}^{(k-1)/2} X_{n+i,m+j} * W_{i,j} \quad (1)$$

It can be seen from (1) that the amount of data to be loaded for convolution of n×n matrix with k×k convolution kernel is k×k×(n-k+1)×(n-k+1). Because of the nature of convolution operation, there are a lot of repetitions in the required data. The loading of repeated data not only limits the throughput of its own operation, but also consumes the access bandwidth of the bus, thus affecting the performance of the whole system. In order to decrease the data bandwidth occupation of convolution operation and improve the throughput of the acceleration unit, from the perspective of reducing the data bit width, the fixed-point number of Q8.8 format is used for the data in the acceleration circuit, and the fixed-point number of 16bits reduces the data bit width, bandwidth and storage while not affecting the network accuracy.

We have designed a low-bandwidth convolution circuit to reduce the data repetition rate. The source data loading unit in convolutional circuit is the point to decrease the data bandwidth. The source data loading unit consists of a data reading unit and a circular queue. Taking the size of the source matrix as M×N and the size of the convolution kernel as 3×3 as an example, the loading process of the two-dimensional convolution source data is shown in Fig. 2.

When the convolution starts, the start pointer of the loop queue points to the front of the queue. The source data loading unit loads 9 data from the first round of convolution operations into the circular queue in columns. At the same time, the convolution circuit sequentially reads 9 data needed for the first round of convolution operation from the start pointer. After the first round of operation, the source data loading unit takes out the three data in the next column from the memory, replaces the three data in the queue that the start pointer points to the address, and moves the start pointer backward 3 bits. At the same time, the convolution circuit sequentially reads 9 data needed for the second round of convolution operation from the start pointer. And so on, we can get the data loading mode after. For k×k convolution kernel convolution n×n matrix, the amount of data to be loaded after optimization is:

$$N*k*(N-k+1) \quad (2)$$

The optimization rate η is:

$$\eta = 1 - \frac{N*k*(N-k+1)}{k*k*(N-k+1)*(N-k+1)} = 1 - \frac{N}{k*(N-k+1)} \quad (3)$$

It can be seen from (3) that the larger the convolution core is, the better the optimization rate of the source data loading unit is. Taking the first layer of lenet-5 as an example, if n is 32, K is 5, then η is 77.1%.

C. RNN Circuit Design

RNN node loop connection, its current hidden state is the input of the next moment.So RNN can remember the past information, but it can't learn the long-term dependence. It is difficult for vanilla RNN to train growth sequences due to the disappearance or explosion of gradients [13]. LSTM is a variant of RNN, which expressly adds storage controllers to determine when to remember, forget, and output. This allows the LSTM model to learn long-term dependencies [14]. The structure of LSTM is shown in Fig. 3.

There are three gates in the LSTM structure, which are forgetting gate, input gate and output gate. In the figure, $\oplus$ represents the addition of the vectors, and $\odot$ represents the element-wise multiplication of the vectors. The characteristic equation are as follows:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (4)$$

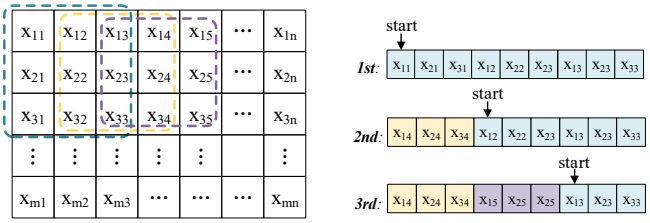

Figure 2. 2D convolution data reading demonstration diagram

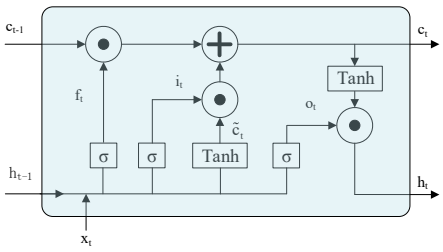

Figure 3. The LSTM architecture

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (5)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (6)$$

$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (7)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (9)$$

According to the characteristics of LSTM structure, the acceleration circuit is designed. There are three main circuit modules. As shown in Fig. 4, Gate module multiplies input x_t and h_{t-1} with weights W_x and W_h respectively, then adds the results and passes through tanh or sigmoid functions. The Ewise module performs the element multiplication operation on the input i_t and $\tilde{c}_t$, f_t and c_{t-1}, then sums them to get c_t, and then after c_t passes through tanh function and take element-wise multiplication with o_t to get h_t. Where c_{t-1} and h_{t-1} are the outputs of the previous time.

III. VERIFICATION PLATFORM CONSTRUCTION

A. Design of Verification Platform Based on Cortex-M3

In order to realize the function verification and performance evaluation of the accelerator, a verification platform is built based on the open-source Cortex-M3 core. Cortex-M3 is the processor core of the IoT system released by ARM company. It has both low power consumption and good performance. The designed SoC structure is shown in Fig. 5.

The designed SoC is composed of functional units such as Cortex-M3, on-chip memory, CNN acceleration unit, LSTM acceleration unit and UART. The Cortex-M3 acts as a processing core in the SoC for executing applications and

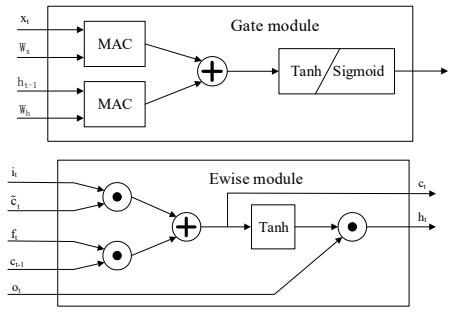

Figure 4. The architecture of the Tanh, Sigmoid and Ewise module

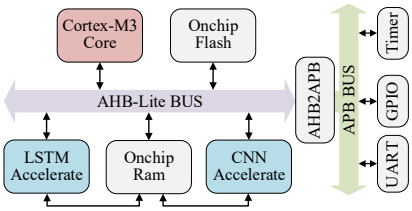

Figure 5. The architecture of SoC

completing the configuration and invocation of multi-algorithm accelerator units. The on-chip memory of SoC is divided into instruction memory and data memory, which are respectively used to save the instruction code and the data generated at runtime. Taking CNN and LSTM acceleration unit as test objects, acceleration is realized under the control of processor. Peripheral devices such as UART and GPIO are used as the interaction unit of SoC, so that users can detect the state of SoC.

B. Implementation of CNN

In order to verify the acceleration effect of accelerator on CNN algorithm, the classical lenet-5 network is used in this paper. Lenet-5 is a CNN for handwritten character recognition, which is considered to be one of the earliest and most classical convolutional neural networks. With the in-depth study of CNN, many more efficient and accurate CNN structures have been proposed, but lenet-5, as a classic network structure, is still widely used in the performance evaluation of CNN acceleration unit. The structure of Lenet-5 is shown in Fig. 6.

Lenet-5 consists of five hidden layers: convolution layer C1 with six convolution cores, pooling layer S2, partial connection layer C3 with sixteen convolution cores, pooling layer S4 and full connection layer S6 with ten convolution cores [12]. Convolution layer C1 contains six 5×5 size convolution kernels, and the feature map is activated by ReLU function. Each convolution kernel convolves the original image with an input size of 32×32 to generate six 28×28 feature maps. Partial Connection Layer C3 is the most complex layer in Lenet-5 network. It contains sixteen 5×5 convolution cores. Each convolution core has a partial connection with six feature maps output by S2. After calculation and activation of this layer, sixteen 10×10 feature maps will be obtained. Partial connection relation and calculation process are shown in Fig. 7.

Take the 0th feature map of C3 as an example, firstly, the 0th convolution kernel is convoluted with the 0, 1 and 2 feature maps of S2 layer respectively. Then, the results of the three convolutions are added and the biasis added. Finally,

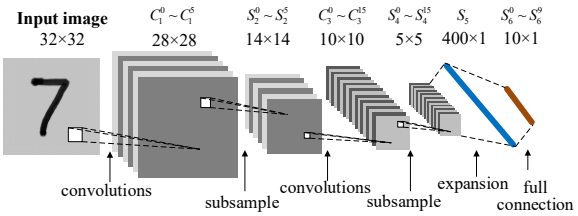

Figure 6. The structure of Lenet-5

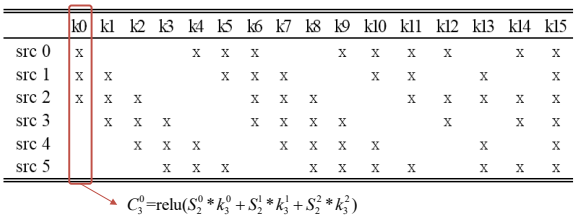

Figure 7. Partial connection relation and calculation process of Layer C3

the 0th feature map of C3 layer is obtained by activation. Pooling layer S4 pools the output of layer C3, and the pooling result is 16 feature maps of 5×5 size. Finally, in the expansion layer S5, sixteen 5×5 feature maps are extended to 1×400 one-dimensional matrix. In the S6 layer, the extended matrix is connected with 10 convolution operators to get 10 classification results, that is, the recognition results of the input image. Therefore, the calculation of lenet-5 network is summarized as follows.

$$\begin{aligned}
&[C_1^0 \sim C_1^5] = relu[S * [K_1^0 \sim K_1^5]] \\
&[S_2^0 \sim S_2^5] = \text{pool}[C_1^0 \sim C_1^5] \\
&[C_3^0 \sim C_3^{15}] = relu[\sum_{i=0}^{5} S_2^i * [K_3^0 \sim K_3^{15}]] \\
&[S_4^0 \sim S_4^{15}] = \text{pool}[C_3^0 \sim C_3^{15}] \\
&S_5 = [S_4^0 \sim S_4^{15}] \\
&[S_6^0 \sim S_6^9] = S_5 * [K_6^0 \sim K_6^9]
\end{aligned} \tag{10}$$

C. Implementation of RNN

The main operations of LSTM hardware are matrix vector multiplication and nonlinear functions (tanh and sigmoid). Matrix vector multiplication is calculated by a multiply accumulate (MAC) cell. Before the calculation, write the offset to Mac, and then carry out the multiplication and accumulation operation. After the calculation, there is no need to add additional offset, reduce the operation steps and reduce the additional hardware logic.

Compared with the nonlinear function ReLU in CNN circuit, it is more complex to calculate tanh and sigmoid directly, which requires a lot of computation. The results can be obtained more accurately by looking up the table, but for the fixed-point number of Q8.8, the size of the required look-up table is 2×2^{16}bit. Therefore, using piecewise linear function to simulate tanh and sigmoid can not only reduce the computation, but also ensure the accuracy. The nonlinear function is divided into 13 univariate functions of y=ax+b. different a and b are used for different x, thus tanh and sigmoid functions are realized.

The calculation of LSTM is divided into three steps. The first step is to load W_{xi}, W_{hi} and W_{xc}, W_{hc} from weight ram into sigmoid gate and tanh gate respectively, and then complete x_t and h_{t-1} with the (4) and (7) respectively, and output the result to the ewise module; the second step is to load W_{xf}, W_{hf} and W_{xo}, W_{ho} from weight ram into tanh gate and another tanh gate respectively, and then complete f_t and o_t with the (5) and (6) respectively, and output the result to the ewise module; the third step is to calculate c_t with (8) of the ewise module, and then calculate h_t with (9), that is the output result of the LSTM. Finally, c_t will be saved in the register as the input of the next time.

IV. Test Results and Analysis

A. Performance Analysis of Acceleration Unit

In order to verify the acceleration effect of the accelerator on CNN and LSTM, the execution time of the lenet-5 and LSTM network was tested on hardware and software platforms. Using C language to implement lenet-5's forward propagation program and a character-level language model, which predict the next character based on the previous character. The LSTM model consists of 2 layers and 128 hidden units. The character input and output is a 65 sized vector one-hot encoded. Let Lenet-5 and LSTM networks run on personal computer, Raspberry Pi 3B+ and Orange Pi PC respectively, and compared with the SoC designed in this paper. The processor of personal computer is Intel 7500 with main frequency of 3.5GHz. The processor of Raspberry Pi 3B+ is Cortex-A53 with main frequency of 1.4GHz. The processor of Orange Pi PC is Cortex-A7 with main frequency of 1.6GHz. The execute time of each platform is shown in Table I.

The results show that the accelerator designed in this paper makes the SoC based on Cortex-M3 core have more computing power for CNN and LSTM than the high-performance cores Cortex-A53 and Cortex-A7. Compared with Intel i5 7500, although the computing time is longer, the main frequency of Intel i5 7500 is 70 times that of Cortex-M3, while the computing time is only 1/7 ~1/6 of that of Cortex-M3.

B. Analysis of Resource Consumption

In order to evaluate the resources consumed by the designed accelerator, the accelerator is implemented on Xilinx xc7z020clg484-1 FPGA. The integrated implementation tool is Vivado 18.3, and the main clock of the system is controlled at 50MHz. The comparison of resource consumption between the accelerator designed in this paper and reference [3, 10] is shown in Table II.

Compared with the other two designs, the accelerating unit of our proposed structure does not pursue high acceleration performance but rather it obtains most suitable acceleration ratio with small resource consumption and conforms to design concept of IoT devices.

TABLE I. Execution Time of Lenet-5 and LSTM on Each Platform

SoC	Core Frequency	Latency /Image	Time /Character
Intel i5 7500	3.5GHz	1ms	1.2ms
Cortex-A53	1.4GHz	11.8ms	12.8ms
Cortex-A7	1.6GHz	14.7ms	17.9ms
Ours	50MHz	6ms	8.3ms

TABLE II. COMPARISON OF RESOURCE CONSUMPTION

	LUT	FF	BRAM	DSP
Ours	9474	9379	72	0
[3]	189871	181634	112	1176
[10]	16086	6006	6	12

V. CONCLUSION

In this paper, we propose and implement a small area configurable depth neural network accelerator with fixed-point accuracy, in order to improve the neural network computing ability of IoT terminal equipment. The experimental results show that the proposed acceleration module can make the SoC with Cortex-M3 as the core obtain higher computing power than Cortex-A53 with less resource consumption. With image sensor and microphone on the designed accelerator, image processing and voice processing can be completed offline, such as providing voice to text function for hearing-impaired people. In this design, the fixed-point number of Q8.8 is used. Although compared with the 32-bit floating-point number, the storage and calculation amount are reduced, but because of the fault tolerance of neural network, a lower bit width can be used in some locations [16]. In order to further reduce the amount of calculation, it is necessary to implement a neural network accelerator with a fixed-point accuracy with a lower bit width in the future.

ACKNOWLEDGMENT

This work was supported by the Fundamental Research Funds for Central Universities (No. NP2019102), Aeronautical Science Foundation of China (No.201943052001), Project of Science and Technology on Electronic Information Control Laboratory.

REFERENCES

[1] H. Yang, S. Kumara, S. T. S. Bukkapatnam, and F. Tsung, "The internet of things for smart manufacturing: A review," IISE Transactions, vol. 51, (11), pp. 1190-1216, 2019.

[2] Z. Zou, Y. Jin, P. Nevalainen, Y. Huan, J. Heikkonen and T. Westerlund, "Edge and fog computing enabled AI for IoT-An overview," 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hsinchu, Taiwan, 2019, pp. 51-56, doi: 10.1109/AICAS.2019.8771621.

[3] Y. Guan, Z. Yuan, G. Sun and J. Cong, "FPGA-based accelerator for long short-term memory recurrent neural networks," 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC), Chiba, 2017, pp. 629-634, doi: 10.1109/ASPDAC.2017.7858394.

[4] F. Liang, Y. Yang, G. Zhang, X. Zhang and B. Wu, "Design of 16-bit fixed-point CNN coprocessor based on FPGA," 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), Shanghai, China, 2018, pp. 1-5, doi: 10.1109/ICDSP.2018.8631564.

[5] S. Hong and Y. Park, "A FPGA-based neural accelerator for small IoT devices," 2017 International SoC Design Conference (ISOCC), Seoul, 2017, pp. 294-295, doi: 10.1109/ISOCC.2017.8368903.

[6] Z. Li et al., "Laius: An 8-bit fixed-point CNN hardware inference engine," 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, 2017, pp. 143-150, doi: 10.1109/ISPA/IUCC.2017.00030.

[7] J. A. Wijaya and T. Adiono, "Configurable CNN SoC co-processor architecture," 2019 International SoC Design Conference (ISOCC), Jeju, Korea (South), 2019, pp. 281-282, doi: 10.1109/ISOCC47750.2019.9078513.

[8] R. Ding, G. Su, G. Bai, W. Xu, N. Su and X. Wu, "A FPGA-based Accelerator of Convolutional Neural Network for Face Feature Extraction," 2019 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), Xi'an, China, 2019, pp. 1-3, doi: 10.1109/EDSSC.2019.8754067.

[9] P. Meloni, A. Garufi, G. Deriu, M. Carreras and D. Loi, "CNN hardware acceleration on a low-power and low-cost APSoC," 2019 Conference on Design and Architectures for Signal and Image Processing (DASIP), Montreal, QC, Canada, 2019, pp. 7-12, doi: 10.1109/DASIP48288.2019.9049213.

[10] T. Tsai, Y. Ho and M. Sheu, "Implementation of FPGA-based accelerator for deep neural networks," 2019 IEEE 22nd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS), Cluj-Napoca, Romania, 2019, pp. 1-4, doi: 10.1109/DDECS.2019.8724665.

[11] J. Chang, and S. Jin, "An efficient implementation of 2D convolution in CNN," IEICE Electronics Express, vol. 14, (1), pp. 1-8, 2017.

[12] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[13] Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, March 1994, doi: 10.1109/72.279181.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[15] J. Ott, Z. Lin, Y. Zhang, S. Liu, Y. Bengio, "Recurrent neural networks with limited numerical precision," 2016.

[16] K. Guo, S. Zeng, J. Yu, Y. Wang, H. Yang, "A survey of fpga-based neural network accelerator," 2017.

[17] A. X. M. Chang and E. Culurciello, "Hardware accelerators for recurrent neural networks on FPGA," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050816.