Parallelization and Hardware Mapping of Deep Neural Network on Reconfigurable Platform for AI-Enabled Biomedical System

Okada Yuuki,[a)] Jiangkun Wang,[b)] Tomohide Fukuchi,[c)] and Abderazek Ben Abdallah[d)]

The University of Aizu, School of Computer Science and Engineering, Adaptive Systems Laboratory, Japan.

[a)]*Corresponding author: m5251136@u-aizu.ac.jp*
[b)]*Electronic mail: ku@ieee.org.*
[c)]*Electronic mail: d8222111@u-aizu.ac.jp.*
[d)]*Electronic mail: benab@u-aizu.ac.jp.*

Abstract. COVID-19 is still disrupting many parts of the world. A rapid and accurate diagnosis solution is needed to combat the pandemic. As a part of the AIRBiS(AI-Enabled Real-time Pneumonia Detection Bio-medical System), this work conduct hardware acceleration to speed up the diagnosis. We found that more than 90% of the current diagnosis time is spent on the convolution function and have conducted three methods to speed up the convolution operations. Firstly, by applying the Winograd algorithm on convolution layers, the multiplication operations of the matrices can be decreased, which speeds up the calculation. The next step is to improve the data exchange speed between the FPGA and CPU by replacing the normal buffer with LineBuffer. We also tried to improve the calculation speed by quantization, reducing the number of bits used for the filter and the input image. The FPGA board we used for this research is ZCU102. The application used for high-level synthesis is Xilinx SDSoC 2019.1. Using the mentioned approaches, we improved the inference speed from 106ms to 22.2ms per image.

INTRODUCTION

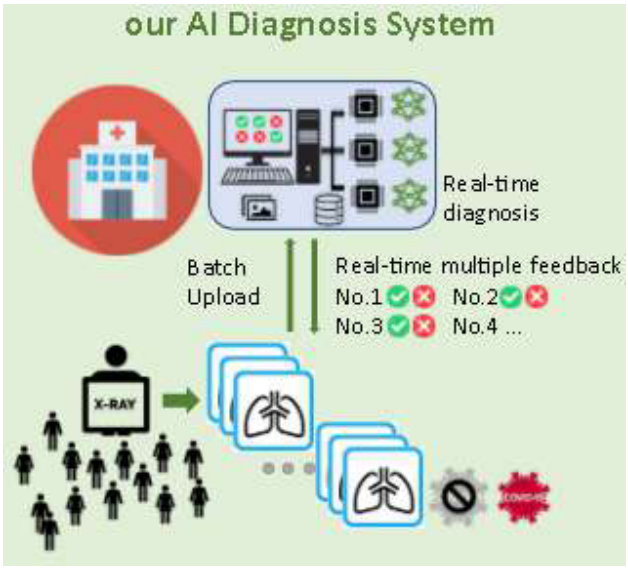

FIGURE 1. AIRBiS system

From the start of 2020, the coronavirus disease (COVID-19), induced by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), [1] became a global health crisis. In response to it, countries worldwide put in effect a range of public health measures. However, the disease continues to spread, and as of October 12, 2021, the number of confirmed global cases has risen to 237,383,711, and deaths to 4,842,716 as confirmed by the World Health Organization (WHO) [2]. As a result of the brisk spread and degree of mortality at the break of the crisis, there was an

immense demand for medical resources which could not be met [3]. The shortage and reliability issues associated with early testing kits needed for precise diagnosis and the lack of coordinated systems required for quick response have prompted the need for an alternative method of diagnosing and responding to the crisis.

Machine learning at the end of the AI spectrum has been increasingly seen in applications in the medical field [4, 5, 6, 7] and is now playing a vital role in combating COVID-19. Several studies have been published which show a good result on the use of AI in COVID-19 diagnosis. The survey in [8] proposed a conceptual framework to screen for COVID-19 by scanning chest CT images for pneumonia types between COVID-19 and intestinal lung disease. In [9], the authors proposed a weakly supervised deep learning framework to detect the probability of COVID-19 using 3D CT volumes. The authors in [10] proposed a 3D densely connected Convolution Neural Network (CNN) to classify COVID-19 patients as either high or low risk by combining CT and clinical information. They aimed to predict whether or not a patient will survive within 14 days based on the patient's initial CT scan and clinical data. While these works have shown impressive results, there remains a need for a comprehensive system that leverages AI for fast and accurate diagnosis and analysis, real-time reporting, and timely response.

AIRBIS OVERVIEW

In this section, we describe the proposed system AIRBiS [11] [12] which is based on deep learning. AIRBiS leverages deep learning medical imaging methods to provide a system for accurate diagnosis of COVID-19 and a combination of a real-time edge-based system for detection/diagnosis and a cloud-based platform for federated learning. AIRBiS merges the merits of software and hardware (AI-Chip) to provide a smart management platform with rapid computation and low power consumption. Detection and diagnosis of AIRBiS as described in Figure 1 are made by extracting COVID-19 features from chest X-ray images [13], and with these features through a CNN on an AI-Chip, classifying patients either as normal or infected. These AI-Chips, when distributed in testing/diagnosing locations which could be hospitals, will form a cluster, and with the help of federated machine learning (FML), efficient distributed learning is conducted. The system provides three user interfaces: one for the FML aggregation, one for testing/diagnosing locations, and one for a mobile user interface. The mobile user interface lets users upload their chest X-ray images to obtain a real-time diagnosis. In this work, however, we focus specifically on the hardware platform to accelerate the CNN of the AIRBiS system. And the CNN model used in this study is shown in Table I.

TABLE I. AIRBiS CNN 64 $\times$ 64 model

Layer	**Input Size**	**Output size**
Conv1	(64,64,1)	(62,62,32)
Max-pooling1	(62,62,32)	(31,31,32)
Conv2	(31,31,32)	(29,29,32)
Max-pooling2	(29,29,32)	(14,14,32)
Conv3	(14,14,32)	(12,12,32)
Max-pooling3	(12,12,32)	(6,6,32)
FC1	1152	128
FC2	128	1

HARDWARE ACCELERATION OF AIRBIS CNN

CNN is a well-established deep learning algorithm employed in computer vision tasks and is designed to learn spatial features at various pattern levels robustly. Its components mainly consist of convolution, pooling, and fully connected layers. In our experiments, we have found out that more than 90% of the computation cost in CNN happens in convolutional layers. In AIRBiS, the CNN model architecture in the table I is used for the prediction of COVID-19. The goal of this CNN acceleration is to speed up diagnosis by optimizing data transfer, and computational processing [14]. We considered FPGA (Field Programmable Gate Array) to achieve this speed-up in hardware. FPGAs save power compared to software implementations and are suitable for prototyping because the circuits can be reconfigured, allowing us to examine various speed-up methods. In addition, the parallelism inherent in hardware provides for high-speed processing. Specifically, we use data transfer methods, pipeline and data segregation for parallel operations,

quantization, and the famous Winograd algorithm for the convolution function [15], and LineBuffer to achieve high speed. The whole system is written in C++, and its structure is shown in Figure 2.

Description of the functions used in CNN Acceleration Designed in C++

- Winograd Convolution 0-2: This function is mainly an adaptation of each convolution function to the Winograd algorithm. The arguments are feature map, weight, and weights.
- Load Data 0-2: In this function, the feature map needed for the first calculation is added to the LineBuffer.
- Write Row Data 0-2: This function extracts each row of data from the feature map.
- Convolution Calculation 0-2: This function is the main function for calculation using the linebuffer to perform convolution.
- Convolution Write 0-2: This function writes the feature map after convolution to the shared memory.
- ReLU: In this function, it plays the role of ReLU in the activation function.
- Pooling: A function that sends data to max pooling.
- Forward: In this function, the function performs in the fully-connected layer.

Data Transfer Method

When accelerating on FPGAs, more access to shared memory is required. For example, AIRBiS CNN uses many loops to perform computations, so it needs to access the shared memory for each calculation. However, using this approach to speed up FPGAs is inefficient because FPGAs run at a lower clock frequency than CPUs, and this approach slows down the computation [16]. A better system is to use DMA (Direct Memory Access). Usually, the host processor handles the data transfer, but when DMA is added, the DMA transfers the data instead of the host processor. As a result, the host processor only performs the arithmetic processing, improving arithmetic speed. In addition, since information is transferred directly by hardware, high speed and large capacity data transfer is possible. In addition, using a linebuffer in FPGA reduces the frequency of memory access, and the data processing time can be greatly reduced, especially when handling large amounts of data.

Parallel Computing Through Pipeline and Data Separation

Typically, one instruction can not be executed until the next one is completed, but in CNNs, the computation is done in many loops, and if we were to do it one by one, it would be really inefficient. Therefore, the pipeline is applied to the computation instructions to make the computation process more efficient in a cycle consisting of multiple steps. In FPGA, there is a command option to convert pipelining and data separation automatically. #pragma HLS pipeline, #pragma HLS array_patition enables data isolation and pipeline of calculations.

Speeding up the Multiplications by Quantization

In CNN, we calculate the result we want by repeating several multiplications, using floating-point numbers. By default, these floating-point numbers are 32 bits. To speed up the multiplication process, we apply quantization, using AP_FIXED<total number of bits,bits representing integer numbers>, which is one of the libraries provided by Xilinx to change the number of a floating-point number from 32-bit to 8-bit.

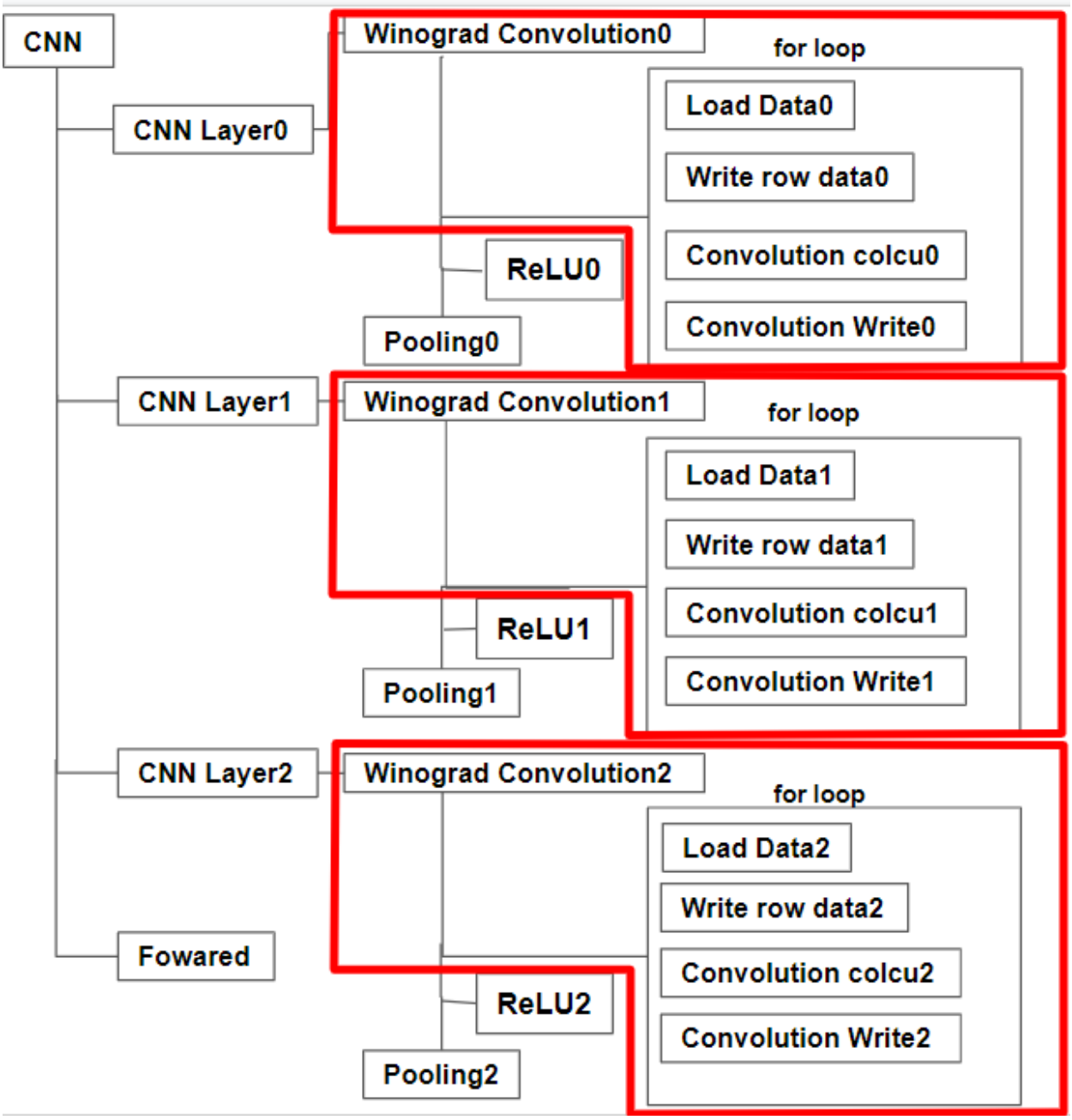

FIGURE 2. CNN functions structure using Winograd algorithm and LineBuffer

Winograd Algorithm

In this section, the process of computation of the Winograd algorithm is described. We define the r-tap FIR filter as $F(m,r)$ and the result of computing m outputs. An FIR filter $F(2,3)$ is computed as follows. In the Winograd algorithm, four multiplications can be formulated using the transformation matrices A, B, and G as follows,

$$\begin{gathered}
\text{In} = \begin{bmatrix} z_0 & z_1 & z_2 & z_3 \end{bmatrix}^{\mathrm{T}} \quad \text{F} = \begin{bmatrix} x_0 & x_1 & x_2 \end{bmatrix}^{\mathrm{T}} \\
\text{Out} = \begin{bmatrix} y_0 & y_1 \end{bmatrix}^{\mathrm{T}} \\
\begin{bmatrix} z_0 & z_1 & z_2 \\ z_1 & z_2 & z_3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} m_1 + m_2 + m_3 \\ m_2 - m_3 + m_4 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}
\end{gathered} \tag{1}$$

m_1, m_2, m_3, m_4 are:

$$\begin{aligned}
m_1 &= (z_0 - z_2)\, x_0 \\
m_2 &= (z_1 + z_2)\, \tfrac{x_0 + x_1 + x_2}{2} \\
m_4 &= (z_1 - z_3)\, x_2 \\
m_3 &= (z_2 - z_1)\, \tfrac{x_0 - x_1 + x_2}{2}
\end{aligned} \tag{2}$$

To calculate $m1$ to $m4$, four multiplication is required using matrices A, B and G.

$$\begin{gathered}
\text{Out} = A^{\mathrm{T}} \left[(GF) \odot \left(B^{\mathrm{T}} \text{ In} \right) \right] \\
B^T = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \quad G = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix} \\
A^T = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & -1 & -1 \end{bmatrix}
\end{gathered} \tag{3}$$

Winograd algorithm for two-dimensional output $F(m \times m, r \times r)$ is calculated as follows. Where $\odot$ is element-wise multiplication (EWMM), m is the output size, r is the filter size, and n is the input size.

$$\begin{gathered}
\text{Out} = A^{\mathrm{T}} [U \odot V] A \\
U = GFG^{\mathrm{T}} \quad V = B^{\mathrm{T}} \text{In} B
\end{gathered} \tag{4}$$

2-D Winograd algorithm as a mixed general-purpose and element-wise matrix multiplication is determined by the definition of the transformation matrices A, B, and G, the n and r are determined. $m \times m$ matrices are generated every time of the Winograd algorithm computation. The number of multiplications is defined by Equation 4. To compute the output feature map, the conventional algorithm requires $m^2\ r^2$ multiplications while the Winograd algorithm requires n^2 multiplications. In our case, more addition is required than the conventional algorithm as it needs to add the intermediate results together.

Change from conventional buffer to LineBuffer

In CNN, we deal with a large amount of data. However, the buffer we used takes a long time to traverse the data when referring to the last information because it is stored in one dimension. This leads to delays in data referencing. Therefore, we changed the buffer to a LineBuffer to reduce the data referencing time. Using LineBuffer, one long buffer is divided into multiple short buffers, which improves the data access speed. Figure 4 shows the difference between the normal buffer and LineBuffer.

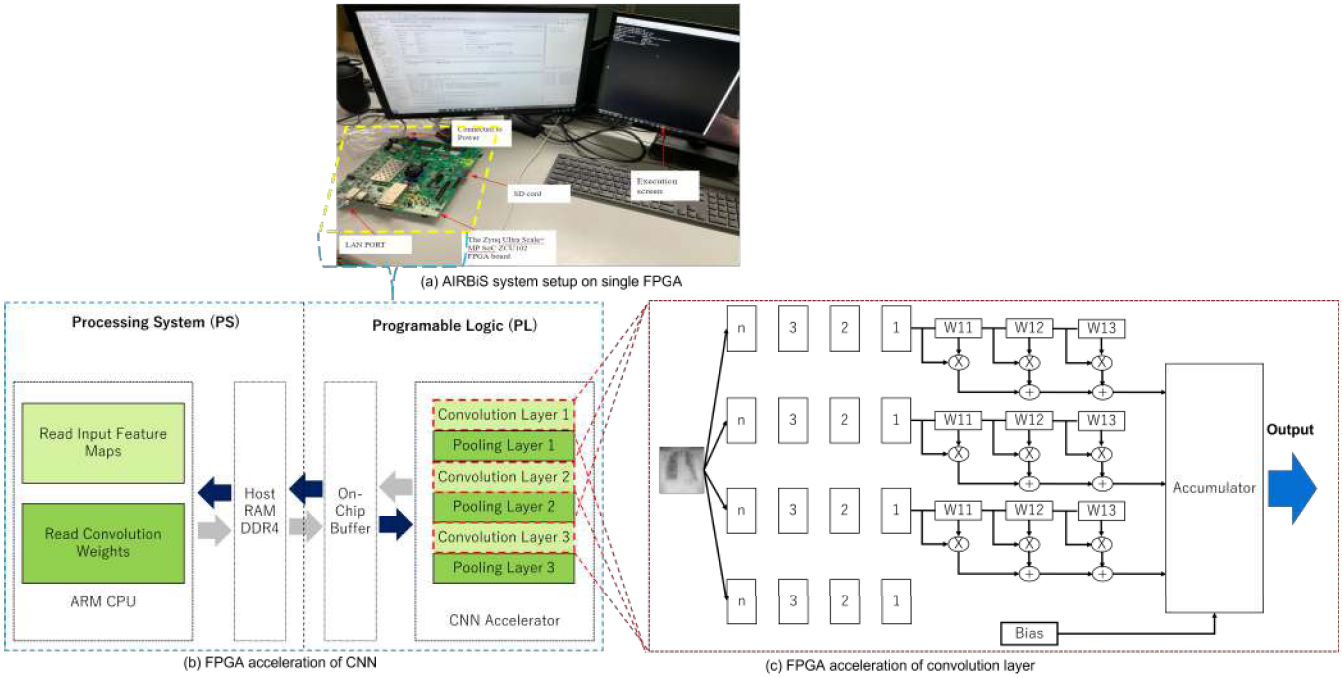

FIGURE 3. Setup for hardware acceleration experiment.

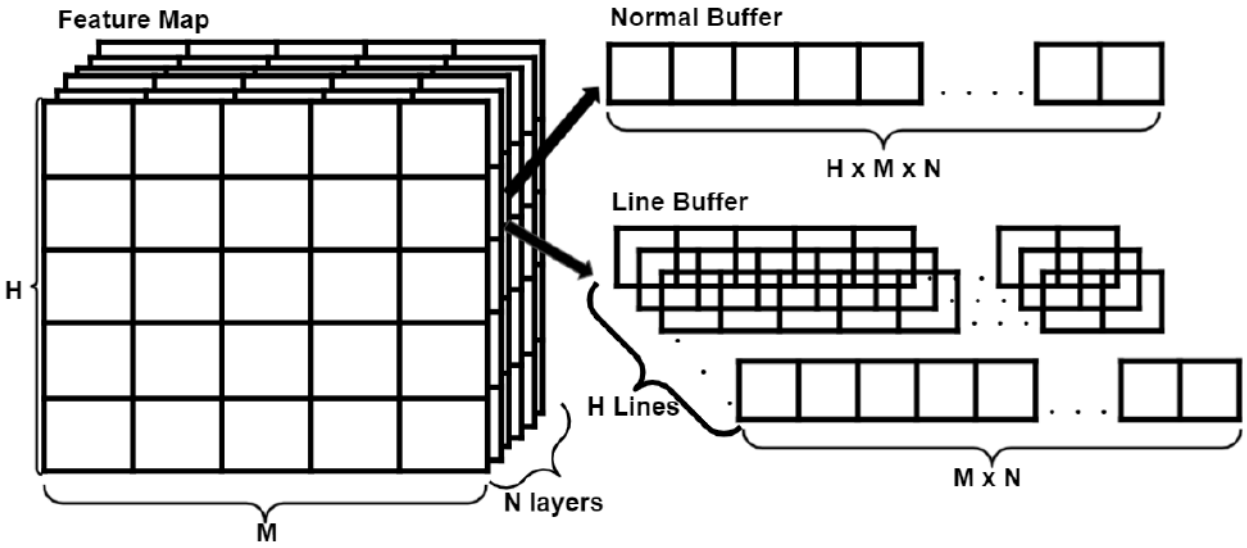

FIGURE 4. Normal Buffer and LineBuffer

EVALUATION

Evaluation Methodology

The AIRBiS CNN acceleration was done in C++, using the Zynq Ultra Scale+ MPSoC ZCU102 FPGA board and the Xilinx SDx IDE development tool. The experiment setup is shown in Fig. 3. The X-ray images used for this evaluation are a combination of two data sets from [13], and [17], and are categorized into two classes, normal and abnormal. Normal indicates that the patient is healthy, while eccentric suggests the presence of pneumonia, including COVID-19. For the evaluation, we measured the average processing time taken to classify lung X-ray images is to got the inference/detection speed, and then the results of different methods are compared.

TABLE II. Time-cost and Power Consumption

MethodResult	Time-cost (ms)	Power(W)	Accuracy
Convolutional(99.99MHz)	130	3.96	4*98.9%
Convolutional(149.99MHz)	106	4.15	
Winograd(99.99MHz)	58.8	6.21	
Winograd(149.99MHz)	57.5	6.49	
Winograd 8bit(99.99MHz)	29.3	5.421	94.3%
Winograd 8bit(149.99MHz)	22.2	7.31	94.4%

TABLE III. Resource utilization

Resource	**Utilization**	**Available**	**Utilization %**
LUT	154586	274080	56.4%
LUTRAM	23901	144000	16.6%
FF	201649	548160	36.8%
BRAM	679	912	74.5%
DSP	87	2520	3.5%
BUFG	27	404	6.7%
MMCM	1	4	25%

Evaluation Results

Table II shows the results of time cost and power consumption for the CNN model with and without the proposed optimization methods. The Winograd with the 8-bit process got less accurate compared to the 32-bit. However, this accuracy is still acceptable. The Winograd 8-bit method achieves a 22.2ms latency when the FPGA clock is 149.99MHz and the best power consumption of 5.421W when the FPGA speed is set to 99.99MHz. Using 8-bit quantization, we achieved 1.32 times better latency results with a 99.99MHz FGPA clock. With a 149.99MHz FPGA clock, we also achieved 4.8 times better latency with the proposed method than the conventional method. Table III shows the result of the resource utilization for the conventional method and the proposed method. Since the proposed method utilizes more hardware resources for parallelization, the hardware complexity and power consumption are slightly higher than the traditional method.

DISCUSSION

In addition to the previous work [18, 19], in this research, we have, for the first time, applied the Winograd algorithm to the convolution function to speed up the processing. However, we were only able to try an input image size of 64 $\times$64 for the inference measurement experiment currently because it used a large number of FPGA resources. Several possible reasons are listed as follows. The first is to change the data from a regular buffer to a LineBuffer

and use #pragma HLS array_patition. This pragma uses more resources than a normal array to access each data. This allows for efficient data access because the information is separated when pipelined and referenced. The second is the frequency of pipeline usage. As We mentioned earlier, the use of #pragma HLS array_patition makes data access much more efficient, and we may have increased too much scope of the pipeline to improve the measurement. In addition, we believe that the large number of functions that we have mapped to FPGA this time is also causing a lack of resources. In this research, we use the ZCU102 board only for system prototype design and experiment. The final target is to deploy it in inexpensive edge programmable devices. In the future, we would like to carefully select the scope of circularization to minimize the size of circularization and conduct research to improve the measurement accuracy.

CONCLUSION AND FUTURE WORK

In this study, we achieved a good acceleration result by adopting the Winograd algorithm and LineBuffer optimization. The first is to optimize the computation through algorithm adaptation, which has led to the speed increase. The second is to apply #pragma HLS array_patition to the LineBuffer to improve the efficiency of data referencing during computation, even though this increased resource utilization. The resources of a single FPGA are limited, which prevents us from processing data with a larger input size and inferencing complex networks with more features. Therefore, the future research goal is to improve processing speed and accuracy by using multiple FPGA.

REFERENCES

1. R. Han, L. Huang, H. Jiang, J. Dong, H. Peng, and D. Zhang, "Early clinical and ct manifestations of coronavirus disease 2019 (covid-19) pneumonia," American Journal of Roentgenology **215**, 338–343 (2020).
2. W. H. Organization, "Who coronavirus disease (covid-19) dashboard," (2020).
3. Y. Ji, Z. Ma, M. P. Peppelenbosch, and Q. Pan, "Potential association between COVID-19 mortality and health-care resource availability," The Lancet Global Health **8**, e480 (2020).
4. B. Ehteshami Bejnordi, M. Veta, P. Johannes van Diest, B. van Ginneken, N. Karssemeijer, G. Litjens, J. A. W. M. van der Laak, , and the CAMELYON16 Consortium, "Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer," JAMA **318**, 2199–2210 (2017).
5. P. Lakhani and B. Sundaram, "Deep learning at chest radiography: Automated classification of pulmonary tuberculosis by using convolutional neural networks," Radiology **284**, 574–582 (2017).
6. A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," Nature **542**, 115–118 (2017).
7. M. Nakamura, J. Wang, S. Phea, and A. B. Abdallah, "Comprehensive study of coronavirus disease 2019 (covid-19) classification based on deep convolution neural networks," in *SHS Web of Conferences*, Vol. 102 (EDP Sciences, 2021) p. 04007.
8. J. Wang, Y. Bao, Y. Wen, H. Lu, H. Luo, Y. Xiang, X. Li, C. Liu, and D. Qian, "Prior-attention residual learning for more discriminative covid-19 screening in ct images," IEEE Transactions on Medical Imaging **39**, 2572–2583 (2020).
9. X. Wang, X. Deng, Q. Fu, Q. Zhou, J. Feng, H. Ma, W. Liu, and C. Zheng, "A weakly-supervised framework for covid-19 classification and lesion localization from chest ct," IEEE Transactions on Medical Imaging **39**, 2615–2625 (2020).
10. L. Meng, D. Dong, L. Li, M. Niu, Y. Bai, M. Wang, X. Qiu, Y. Zha, and J. Tian, "A deep learning prognosis model help alert for covid-19 patients at high-risk of death: A multi-center study," IEEE Journal of Biomedical and Health Informatics **24**, 3576–3584 (2020).
11. A. B. Abdallah, H. Huang, N. K. Dang, and J. Song, "Ai processor," (Japanese Patent Application Laid-Open No 2020-194733 Nov. 2020).
12. M. Nakamura, "Ai-enabled hardware-software system for pneumonia detection." (MS thesis, The University of Aizu, Japan, March 2022.).
13. P. Mooney, "Chest x-ray images (pneumonia)," (2020).
14. T. H. Vu, R. Murakami, Y. Okuyama, and A. Ben Abdallah, "Efficient optimization and hardware acceleration of cnns towards the design of a scalable neuro inspired architecture in hardware," in *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)* (2018) pp. 326–332.
15. L. Lu, Y. Liang, Q. Xiao, and S. Yan, "Evaluating fast algorithms for convolutional neural networks on fpgas," in *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)* (IEEE, 2017) pp. 101–108.
16. H. Isihara, in *An Introduction to FPGAs for Software Engineers* (2017) pp. 96–165.
17. J. P. Cohen, P. Morrison, L. Dao, K. Roth, T. Q. Duong, and M. Ghassemi, "Covid-19 image data collection: Prospective predictions are the future," (2020), arXiv:2006.11988 [q-bio.QM].
18. J. Wang, M. Nakamura, and A. Ben Abdallah, "Efficient AI-Enabled pneumonia detection in chest x-ray images," in *2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech) (IEEE LifeTech 2022)* (Osaka, Japan, 2022).
19. O. Yuuki, J. Wang, O. M. Ikechukwu, and A. B. Abdallah, "Hardware acceleration of convolution neural network for ai-enabled realtime biomedical system," in *SHS Web of Conferences*, Vol. 102 (EDP Sciences, 2021) p. 04019.