Hardware Acceleration of Convolution Neural Network for AI-Enabled Real-time Biomedical System

Okada Yuuki[*], *Jiangkun* Wang, *Ogbodo* Mark Ikechukwu, and *Abderazek* Ben Abdallah

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

Abstract. COVID-19 is currently on the rage all over the world and has become a pandemic. To efficiently handle it, accurate diagnosis and prompt reporting are essential. The AI-Enabled Real-time Biomedical System (AIRBiS) research project aims to develop a system that handles diagnosis using chest X-ray images. The project is divided into UI, network, software and hardware. This work focuses on the hardware, which uses CNN technology to create a model that determines the presence of pneumonia. This CNN model is designed on an FPGA to speed up diagnostic results. The FPGA increases the flexibility of circuit design, allowing us to optimize the computational processing during data transfer and CNN implementation, reducing the diagnostic measurement time for a single image.

1 Introduction

At the dawn of 2020, the coronavirus disease (COVID-19), which is induced by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) [1] became a global health crisis. In response to it, countries around the world put in effect a range of public health measures. However, the disease continues to spread and as of December 8, 2020, the number of confirmed global cases have risen to 66,422,058, and deaths to 1,532,418 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 needed for the quick response have prompted the need for an alternative method of diagnosing and responding to the crisis. the AI-Enabled Real-time Biomedical System (AIRBiS) [4] project.

Machine learning which is at the end of the AI spectrum has been increasingly seen in application in the medical field [5–7], 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 study 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 convolutional neural network (CNN) to classify COVID-19 patients as either high risk 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 information. While these works have shown impressive results, there remains a need for a comprehensive system that not only leverage AI for fast and accurate diagnosis, but also analysis, real-time reporting, and timely reference.

2 AIRBiS Overview

In this section, we describe the proposed system AIRBiS which is based on Deep learning. AIRBiS leverages Deep learning medical imaging methods to provide not just a system for accurate diagnosis of COVID-19, but also, 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 on AIRBiS as described in Fig. 1 are made by extracting COVID-19 features from chest X-ray images [11], and with these features through a CNN on an AI-Chip, classify 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) an efficient distributed learning is conducted. The system provides three user interfaces: one for the FML aggregation, one for the testing/diagnosing locations, and one for a mobile user interface. The mobile user interface enables users to upload their chest X-ray images in order 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.

[*]Corresponding Author: Okada Yuuki e-mail: s1250129@u-aizu.ac.jp

https://doi.org/10.1051/shsconf/202110204019

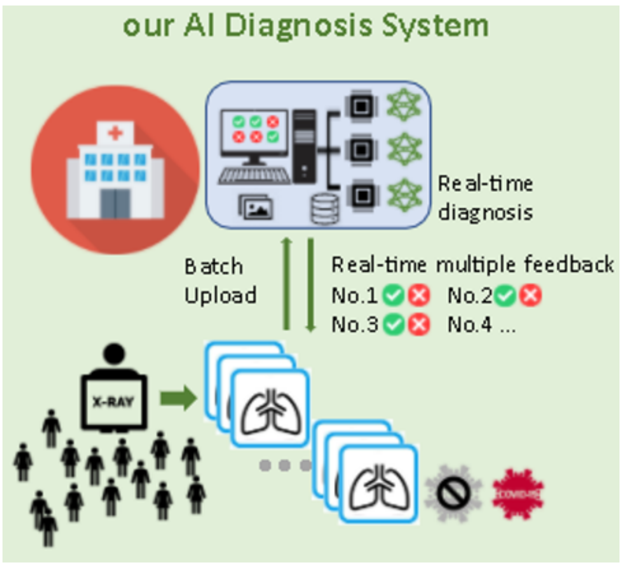

Figure 1. AIRBiS system

Table 1. AIRBiS CNN 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

3 Hardware Acceleration of AIRBiS CNN

CNN is an established Deep learning algorithm employed in computer vision tasks, and it is designed to learn spatial features at various pattern levels robustly. Its building blocks are composed mainly of convolution, pooling and fully connected layers. The AIRBiS system uses the CNN model described in Table 1 for the COVID-19 prediction, and the acceleration of this CNN is aimed at improving the speed of diagnosis by focusing on data transfer and computational processing. In carrying out this acceleration in hardware, a field programmable gate array (FPGA) is considered. This is because FPGA are power-efficient when compared to software implementation, its circuits are reconfigurable and suitable for prototyping, enabling us to explore several acceleration approaches. Also, the parallelism available in the hardware can be leveraged for rapid processing. A development flow of hardware design on FPGA is shown in Fig. 2. The acceleration begins with the creation of the CNN model in python. When this is determined, the creation begins. To improve the diagnostic speed, speed changes can be obtained by methods of transferring data from shared memory, and also, pipeline processing and parallel processing are adopted to shorten the calculation processing time. CNN processes large amounts of data. Therefore, when accelerating on FPGA, there is much access to shared memory. The AIRBiS CNN uses many loops to perform calculations, and this requires accessing the shared memory for each calculation.When this approach is accelerating on FPGA, this approach is inefficient because the FPGA operates at a lower clock frequency than the CPU, and utilizing this approach will make the calculations slower [12]. A better approach will be to utilize direct memory access (DMA). Typically, the host processor processes data transfer. However, when DMA is added, DMA transfers data on behalf of the host processor. As a result, the host processor performs only the calculation process, which improves the calculation speed, as shown in Fig. 3. Besides, since data is transferred directly using hardware, high-speed and large-volume data transfer is possible. The use of Buffer on the FPGA reduces the frequency of memory access which saves a large amount of data processing time, especially when dealing with extensive data.

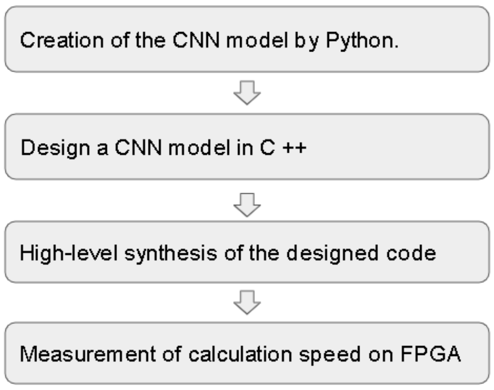
Creation of the CNN model by Python.
Design a CNN model in C ++
High-level synthesis of the designed code
Measurement of calculation speed on FPGA

Figure 2. Hardware development flow using FPGA

3.1 Optimization by Pipelining

By using Pipelining processing [13], which is one of the instruction execution methods, one instruction is divided into multiple stages and executed in different circuits, enabling the execution of several instructions in parallel. Parallel computing is the process of subdividing a particular process into several smaller, independent processes in a computer, and having each process run simultaneously on multiple processing devices. There is a command option to convert parallelism and pipelining processing automatically on FPGA. However, we could do the parallel processing manually because it is faster than the command option. Also, because data is exchanged between functions and this takes time, data processing could be shortened by combining functions that can be combined to reduce the exchange of data between functions.

3.2 Optimization by Function combination

Data is exchanged between functions. Therefore, data processing could be shortened by combining functions that

Table 2. FPGA acceleration result (function combination + buffer + Unroll)

Circutized Functions	Convolution	Convolution + Pooling	Convolution + Pooling + Forward
Implementation Time (ms)	612.587	510.757	520.419

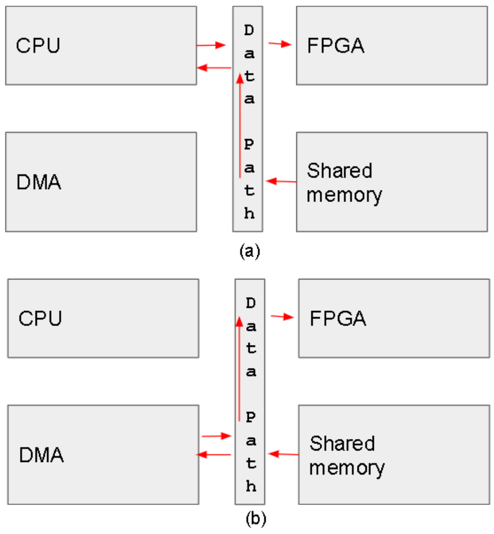

Figure 3. Memory access methods on FPGA: (a) CPU controlled memory access. (b) Direct memory access (DMA). The CPU controlled access method requires the CPU to access the memory for data. However with the DMA approach, this responsibility is absolved from the CPU, so that it can focus on the calculation process.

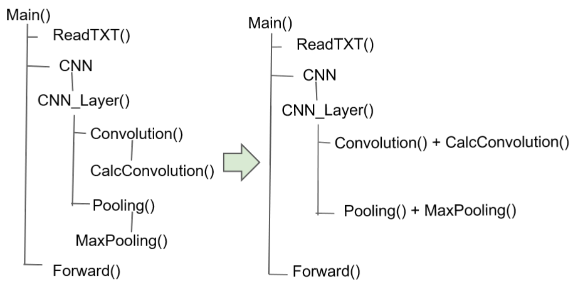

Figure 4. CNN source code structure

can be combined to reduce the exchange of data between functions. The structure of the CNN we designed in C++is shown in Fig. 4. The functions combined in this structure are convolution, calcconvolution, Maxpooling, and pooling.
Function description:

- convolution: A function that sends the necessary data for convolution operations.
- calcconvolution: A function that performs a convolution operation.
- pooling: A function that sends data to max pooling.
- Maxpooling: A function that implements max pooling.

3.3 Optimization of circuit

In Xilinx, there are high level synthesis (HLS) tools which provide optimizations using options called #pragma. The optimization options used in this study are as follows. The optimization options are written in the C/C++ source code.

1. #pragma SDS data mem attribute: used to allocate memory for DMA.
2. #pragma SDS data access pattern: Can select the data transfer method as either bulk transmission or random transmission.
3. #pragma SDS data zero copy: Can direct data transfer from shared memory.
4. #pragma SDS data buffer depth: Set up the buffer.
5. #pragma HLS unroll
6. #pragma HLS Pipeline

5 and 6 are options for parallel processing and pipeline processing.

4 Evaluation

4.1 Evaluation Methodology

The AIRBiS CNN acceleration was done in C++, using the Zynq Ultra Scale+ MP SoC ZCU102 FPGA board, and the Xilinx SDx IED development tool. The experiment setup is shown in Fig. 5. The X-ray images used for this evaluation are a combination of two data sets from [11], and [14], and are categorized into two classes normal, and abnormal. Normal indicates that the patient is healthy, while abnormal indicates the presence of COVID-19 related pneumonia. After classification, The average processing time taken to classify an image is measured to determine theprocessing speed and then compared to the time taken to run on just the FPGA ARM CPU.

4.2 Evaluation Results

For evaluation, we use 5216 training sets and 624 test images, and the CNN model was trained in grayscale. Several image sizes were used to improve the accuracy of the adopted CNN model, and the results are summarized in

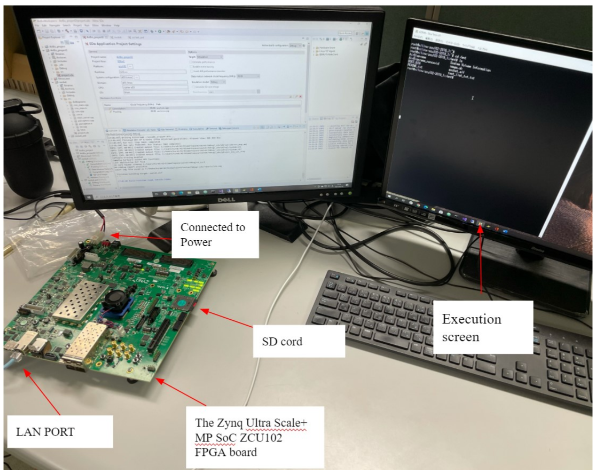

Figure 5. Setup for hardware acceleration experiment.

Table 3. Prediction accuracy over various image sizes

Input size	Test Accuracy
32×32	0.8958
64×64	0.9327
128×128	0.9279
256×256	0.9103

Table 4. Prediction accuracy over various image sizes

Onboard ARM CPU	Without opt.	With opt.
Implementation time	771.82	429.012

Table. 3. From the results, It can be seen that increasing the size of the image used did not yield any increase in inaccuracy. However, because the CNN model was not complex, the accuracy came out best with an image size of 64x64. Another technique that was employed to improve accuracy is dropouts placement, which attempted dropout after the last pooling and then after each pooling. As a result, overshooting problems during learning were reduced. However, there was no difference in test accuracy.

The processing time achieved from our applied accelerated approach which merges function combination, use of buffer, and Unroll is described in Table. 2. in this table, we can see that convolution alone takes up 98% of the time while pooling and forward takes up 2%. When compared to the un-optimized implementation on FPGA ARM host processor as described in Table. 4, the accelerated CNN shows 32.57% improvement. However, the performance obtained from the optimized FPGA ARM host processor is yet to be attained. For the hardware complexity of the fully connected network on the ZCU-102 FPGA 9.7% of the LUT, 1.6% of the DSP48, 2.2% FF, and approximately 4.8% 18k BRAM was utilized. In storing the weights, biases, and input features in the format of a single-precision floating-point, BRAM of 731Bytes in total (651-Byte weights + 5-Byte biases + 75-Byte inputs) was utilized. We conclude that the hardware complexity of the system is small as it occupies only a fraction of the available FPGA resources.

5 Discussion

The hardware acceleration of the AIRBiS CNN has been described in previous sections. However, some issues need to be addressed to obtain better diagnostic speed. Creating a model in RGB may be a way to obtain better performance since features can be extracted better in RGB than in grayscale. However, because the images we are using are X-ray images, we created a CNN model in grayscale with the assumption that RGB would not make much difference. Based on FPGA usage, there is still room for improvement in FPGA. So we would like to continue to develop hardware that exceeds the diagnostic speed of the FPGA ARM host processor, further improving the diagnostic speed by focusing on the data transfer method in terms of the yet untested CNN functions to improve the diagnostic speed.

6 Conclusion

In this study, we investigated the use of FPGAs to improve the speed of diagnosis of the AIRBiS CNN. In the data transfer method, from random access to bulk access, placement of buffers in schematized functions, and a combination of several features to reduce data access to increase the diagnostic speed. For computational processing, the use of pipeline processing and parallel processing compilation options to increase diagnostic speed. The goal of increasing diagnostic speed could be achieved by optimizing the above. However, the achieved diagnostic speed could not exceed the diagnostic speed on the the host processor, although the difference in diagnostic speed per piece could be reduced.

References

[1] Rui Han, Lu Huang, Hong Jiang, Jin Dong, Hongfen Peng, and Dongyou Zhang. Early clinical and CT manifestations of coronavirus disease 2019 (COVID-19) pneumonia. *American Journal of Roentgenology*, 215(2):338–343, August 2020.

[2] World Health Organization. Who coronavirus disease (covid-19) dashboard, 2020.

[3] Yunpeng Ji, Zhongren Ma, Maikel P Peppelenbosch, and Qiuwei Pan. Potential association between COVID-19 mortality and health-care resource availability. *The Lancet Global Health*, 8(4):e480, April 2020.

[4] Abderazek Ben Abdallah, Huankun Huang, Nam Khanh Dang, and Jiangning Song. Ai processor, Nov. 2020. Japanese Patent Application Laid-Open No 2020-194733.

[5] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes van Diest, Bram van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen 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(22):2199–2210, 2017.

[6] Paras Lakhani and Baskaran Sundaram. Deep learning at chest radiography: Automated classification of pulmonary tuberculosis by using convolutional neural networks. *Radiology*, 284(2):574–582, August 2017.

[7] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, January 2017.

[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(8):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(8):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(12):3576–3584, 2020.

[11] Paul Mooney. Chest X-Ray Images (Pneumonia). https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia, 2020.

[12] Hidemi Isihara. In *An Introduction to FPGAs for Software Engineers*, pages 96–165, 2017.

[13] Tong Geng, Tianqi Wang, Ahmed Sanaullah, Chen Yang, Rui Xu, Rushi Patel, and Martin Herbordt. Acceleration and load balancing of cnn training on fpga clusters. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2018.

[14] Joseph Paul Cohen, Paul Morrison, Lan Dao, Karsten Roth, Tim Q Duong, and Marzyeh Ghassemi. Covid-19 image data collection: Prospective predictions are the future, 2020.