論 文 内 容 の 要 旨

Introduction

Nowadays, using the particle swarm optimization (PSO) algorithms to train neural networks (NN) has become an attractive research. A famous method used to train the NNs is the back-propagation (BP) algorithm. Recently, previous studies have shown that the NN trained by PSO algorithms (NN-PSO) obtained higher recognition rates and lower learning errors when compared to the NN trained by the conventional BP algorithm. However, the standard PSO (SPSO) algorithm may stick to a local minimum in the training process. In this situation, the SPSO algorithm is stopped, leading to a high learning error rate or a low recognition rate of the trained NN. So far few studies have tried to overcome this problem by improving the standard PSO algorithms. Normally in previous studies, the improved versions of the PSO required to add more compute-intensive tasks that often increase computational burden the SPSO algorithm.
Typically, previous researchers have implemented the NN trained by PSO algorithms only in hardware or only in software. Normally, the hardware implementation of the NN-PSO in these studies has been tested in a simulation using ModelSim program with the SPSO algorithm but it has not been tested in the real classification tasks. The FPGA-based NN was commonly investigated with a function-approximation, and the researchers just focused on reducing learning errors in the training phase.

The main objective

To deal with the issues described above, this research has three main objectives. The first is to introduce the improved PSO algorithms which can overcome the premature convergence of the SPSO algorithm without adding many compute-intensive functions to it.
The second objective is to propose co-design architectures between hardware and software for the NN-PSO. Compared to the hardware-only approach, the proposed co-design approach not only maintains the testing speed but also reduces the required FPGA resources concerning the logic elements and the memory bits previously reserved for the hardware implementation of the PSO algorithms. Compared to the software-only approach, the proposed co-design approach preserves the flexibility during in the training phase while obtaining the higher operating speed in the testing phase. The flexibility of the co-design is an easiness to modify the PSO parameters or change the PSO algorithms without redesigning or rebuilding the FPGA part.
The third objective is to investigate the performances concerning the learning errors and recognition rates in classification tasks of the NN trained by PSO algorithms implemented in a real FPGA device with the proposed co-design architectures comparing to the NN trained by the standard and the dissipative PSO (SPSO, DPSO) presented in previous studies. The DPSO algorithm is chosen because it keeps the particles out of the local minimum without adding many tasks to the PSO algorithm.

Summary of the thesis

Chapter 1

This chapter briefly presents the overview and objectives of this research.

Chapter 2

Chapter 2 presents related work concerning the NN, the SPSO, and the DPSO algorithms. It also details how the NN can be trained using the PSO algorithms.

Chapter 3

Chapter 3 focuses on the premature convergence issue of the SPSO algorithm by proposing three improved versions of the SPSO algorithm called wPSOd_CV, PSOseed, and PSOseed2. By modifying the velocity update function of the SPSO algorithm, these improved PSO algorithms do not add many compute-intensive tasks to the SPSO algorithm.

The first PSO algorithm introduced in this chapter is the wPSOd_CV algorithm that has two main components called the velocity control and the weight control. The velocity control has a jump phase to increase the velocity of the particle so that particle can move to another searching area to avoid the premature convergence. The weight control is used to balance between the exploitation task and the exploration task of the wPSOd_CV algorithm.

The wPSOd_CV algorithm has some drawbacks. Even modifying only the velocity update function, this algorithm adds several arithmetic operators, increasing the compute-intensive tasks of the SPSO algorithm. In addition, the jumping phase in the velocity control mechanism always has very big jumps. If the searching area has many solutions, the wPSOd_-CV has the possibility to meet other solutions and jump to other searching space before meeting the best solution. This issue leads to a possibility to ignore the best solution or require more iterations to reach the best solution. To overcome these drawbacks, the PSOseed algorithm is proposed. Avoiding jumping phases, for each particle the PSOseed algorithm uses a new variable called the seed position that is randomly generated in the initial phase of the algorithm. In each iteration, each particle is attracted and pulled to the position of its seed. The seed mechanism could reduce the possibility that the particle falls in a local minimum. Compared to the wPSOd_CV algorithm, the PSOseed algorithm does not use any division operator and use fewer multiplication operators. A limitation is that the PSOseed algorithm highly depends on the generated seeds. The performance of the PSOseed significantly is reduced if the seeds are poorly generated.

This chapter also introduces the PSOseed2 algorithm to solve the issue concerning the seed positions of the PSOseed algorithm by proposing a seed control mechanism. The operation of the PSOseed2 algorithm is similar to the PSOseed algorithm. However, in each iteration, all seed positions will be reseeded if the particles in the PSOseed2 algorithm cannot find a better position when the fitness value of the algorithm does not change.

Chapter 4

In the NN-PSO system, the PSO algorithms are only used in the training phase while the NN is used in both the training phase and the testing phase. Therefore, this chapter investigates co-design architectures by proposing a partitioning methodology between hardware and software for the NN-PSO. In this design, the NN is kept in hardware coded by SystemVerilog programming language while the PSO training is moved to the software side. The PSO algorithms are programmed using C language and executed by a processor.

The co-design architectures has several advantages. First, the FPGA-based NN maintains the high speed of the testing phase. The FPGA-based NN has a higher operating speed than the software-based NN. Second, the required FPGA resources regarding the logic elements and the memory bits are reduced because the FPGA resources previously reserved for the hardware implementation of the PSO algorithms are not used in this approach. Third, in the training phase, it is easy to change the parameters of the PSO algorithms such as the number of iterations, the number of particles or even the new PSO algorithm can be used to improve performance of the NN-PSO in different practical applications. Finally, the co-design can employ the NIOS II processor or the ARM processor which is already implemented on an FPGA board. In this case, all components of the co-design approach can be handily deployed in a single System on Chip (SoC) FPGA board.

Chapter 5

Based on the partitioning methodology presented in chapter 4, chapter 5 focuses on the first co-design architecture which uses the NIOS II processor on the software side. The NIOS II is chosen because all components of this co-design architecture could be implemented in a single FPGA chip which is portable and has low power consumption. The NIOS II processor is also optimized by Altera for the FPGA design. Several publicly recognized databases for the classification jobs are used to confirm performance of the NN trained by the proposed PSO algorithms in this co-design architecture. Experiments were conducted on a real FPGA device called the DE1-SoC board. Experimental results confirmed that the NN-PSO was successfully implemented. The NN trained by the proposed PSO algorithms also had higher recognition rates and lower learning errors than the NN trained by SPSO algorithm and DPSO algorithm under the same conditions of the parameters and the FPGA device. Among three proposed PSO algorithms (wPSOd_CV, PSOseed, and PSOseed2), the NN trained by the PSOseed2 algorithm obtained the highest accuracy. In addition, the co-design architecture using the NIOS II processor has both advantages regarding the fast speed of the testing phase when compared with the software-only approach, and lower resource utilization when compared with the hardware-only approach.

This chapter also mentions some drawbacks of the co-design architecture uses the NIOS II processor. The first drawback is that the frequency of the NIOS II cannot be set to a high value due to the constraints with the FPGA fabric. The second drawback is the FPGA resources regarding the logic elements and the memory bits. The NIOS II which is created using available resources from the FPGA devices could be the resource-intensive processor, especially, if a high computing power of the NIOS II is required.

Chapter 6

This chapter deals with the performances of the NN trained by proposed PSO algorithms in the second co-design architecture that uses the ARM processor. The ARM processor is used to overcome the drawbacks of the first architecture using the NIOS II processor. The NIOS II processor is replaced by the ARM processor, and PSO algorithms are executed by the ARM processor on the software side.

The using of the ARM processor can be explained by two reasons. First, the ARM processor can operate at a higher clock frequency than the NIOS II processor. Second, the ARM processor is physically implemented on the FPGA board while the NIOS II processor is implemented using the available FPGA resources of the FPGA device. Hence, the co-design approach using the ARM processor can reduce the required resources previously reserved for the NIOS II processor. Beside the ARM processor, the second architecture also can use the direct memory access (DMA) and random access memory (RAM). The DMA may increase the operating speed of the connections between the software side (the ARM processor) and the hardware-based NN. In the first architecure using NIOS II processor presented in chapter 5, on-chip memory which consumes FPGA resources is used. The second architecture using the ARM processor employs DDR3 RAM to reduce the required memory bits previously reserved for the on-chip memory.

Chapter 6 also shows experiments confirming that the NN-PSO was successfully implemented in the co-design architecture using the ARM processor. The co-design architecture using the ARM processor also had a higher testing speed and required lower FPGA resources than the NIOS II processor. The FPGA-based NN trained by the PSOseed2 obtained higher recognition rates and lower learning errors when compared with the FPGA-based NN trained by other PSO algorithms (SPSO, DPSO, wPSOd_CV, and PSOseed). However, the hardware implementation of the NN still requires many resources. A big NN that has a large number of input nodes may not be fitted in the FPGA device.

Chapter 7

Chapter 7 presents a method–to improve performances of the NN trained by proposed PSO algorithms in the proposed co-design architectures. To reduce the required resources while maintaining the high recognition rate, this chapter proposes the third co-design architecture which is a combination between the co-design architecture using ARM processor presented in chapter 6 and the FPGA-based principal component analysis (PCA). The performances of the NN trained by the proposed PSO algorithm in the third co-design architecture are also discussed in this chapter. The Generalized Hebbian Algorithm is used as the PCA because it was considered as one of the best PCA algorithms for the hardware implementation concerning the required resources.

The operation of the third co-design architecture (the NN-PCA architecture) consists of two phases. The first phase is the unsupervised learning for the PCA training. The second phase is the supervised learning using the PSO algorithms to train the weights and biases of the NN.

Regarding the accuracy in classification tasks, several publicly recognized databases for the classification jobs are used. Experimental results confirmed that the NN-PCA architecture was successfully implemented on a real FPGA board. With the same condition of the PSO

parameters and the same number of the hidden nodes, hardware-based PCA can reduce the required resources of the program while keeping high recognition rates. Furthermore, The NN-PCA architecture also obtained higher testing speed than the software-only approach. Compared with four other PSO algorithms (SPSO, DPSO, wPSOd_CV, and PSOseed) the NN trained by the PSOseed2 algorithm in NN-PCA architecture also achieved the highest recognition rates and the smallest learning errors.

Chapter 8

Chapter 8 concludes all chapters and discusses the future directions for continuing the research.