Journal of KIIT. Vol. 19, No. 12, pp. 63-70, Dec. 31, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.12.63

합성곱 신경망의 추론 정확도 향상을 위한 하드웨어 구조 연구

전성호*, 옥승호**

A Study on Hardware Architecture to Improve Inference Accuracy of Convolutional Neural Networks

Seong-Ho Jeon*, Seung-Ho Ok**

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 지역지능화혁신인재양성 (Grand ICT연구센터) 사업의 연구결과로 수행되었음" (IITP-2021-2020-0-01791) "본 연구는 IDEC에서 EDA Tool를 지원받아 수행하였습니다."

요 약

본 논문에서는 합성곱 신경망의 추론 정확도 향상을 위해 배치 정규화가 내장된 하드웨어 구조를 제안한다. 배치 정규화는 합성곱 신경망 각 계층별 데이터 분포를 정규화함으로써 추론 정확도를 향상시키지만 평균 및 분산 등의 연산이 요구되며, 하드웨어 자원 사용량에 제한이 있는 경우 연산 정확도가 낮아지는 단점이 있다. 이에 본 논문에서는 연산 정확도 향상을 위해 추론 시 8비트 양자화 기법을 적용하고, 내부 컨트롤 신호를 활용하여 배치 정규화 연산이 생략 가능한 하드웨어 구조를 제안한다. 제안하는 하드웨어 구조는 Verilog-HDL을 사용하여 모델링한 후 하드웨어 연산의 중간 출력값 및 최종 출력값을 소프트웨어 기반 검증 모델의 출력 결과와 비교 평가함으로써 정상 동작함을 확인하였으며, Samsung 28-nm 공정을 사용하여 합성하였다.

Abstract

In this paper, we propose a hardware architecture with built-in batch normalization to improve inference accuracy of convolutional neural networks. Batch normalization improves inference accuracy by normalizing the data distribution by the convolutional neural network layer. However, operations such as average and variance are required, and if there is a restriction on hardware resource usage, the operation accuracy decreases. Therefore, this paper proposes a hardware architecture in which batch normalization operations can be omitted by applying an 8-bit quantization technique when inferencing to improve computational accuracy and utilize internal control signals. The proposed hardware architecture was modeled using Verilog-HDL. Then the intermediate and final output values of the hardware operation were compared and evaluated with the output results of the software-based verification model and synthesized using the Samsung 28-nm process.

Keywords

AI accelerator, CNN, batch normalization, verilog-HDL, ASIC

* 동의대학교 인공지능학과 석사과정
- ORCID: https://orcid.org/0000-0002-8869-5294
** 동의대학교 로봇·자동화공학전공 조교수(교신저자)
- ORCID: https://orcid.org/0000-0002-9036-0872

· Received: Nov. 16, 2021, Revised: Dec. 10, 2021, Accepted: Dec. 13, 2021
· Corresponding Author: Seung-Ho Ok
Dept. of Robot·Automation Engineering, Dong-eui University, Korea,
Tel.: +82-51-890-2265, Email: osh@deu.ac.kr

Ⅰ. 서 론

최근 고성능 컴퓨팅 분야에서는 다양한 AI 전용 반도체를 사용하여 컴퓨팅 처리 속도를 가속화하기 위한 연구가 활발히 이루어지고 있다[1][2]. 최적화된 AI 가속기 개발을 위해서는 AI 알고리즘 및 합성곱 신경망의 추론 정확도 향상을 위한 연구가 활발히 이루어지고 있다[3].

합성곱 신경망의 추론 정확도 향상을 위해서는 특징맵 추출 기법인 합성곱뿐만 아니라 배치 정규화를 사용하거나, 그래디언트(Gradient) 소실을 방지하기 위한 네트워크 구조 연구가 있다[4]. 특히 배치 정규화의 경우 사용 유무에 따라 추론 정확도의 차이가 있어 합성곱 신경망의 추론 정확도 향상을 위해 필수적으로 사용한다[5].

배치 정규화는 입력 특징맵에 대한 평균, 분산 등의 추가 연산이 필요할 뿐만 아니라 합성곱 신경망 추론 시 제한된 임베디드 환경에서 적은 수의 비트를 사용하여 연산을 수행할 경우 중간 연산 값의 데이터 손실로 인해 추론 정확도가 저하된다. 이를 위해 적은 비트의 고정 소수점 신경망, 이진 신경망 등에 배치 정규화를 사용하여 신경망을 학습하고 추론 시 배치 정규화를 생략하는 연구가 있다. 하지만 이는 부동 소수점으로 학습시킨 모델 대비 추론 성능이 저하되는 단점이 있다[6]-[8].

본 논문에서는 배치 정규화를 하드웨어로 구현 시 저하되는 추론 정확도를 유지하기 위해 부동 소수점으로 학습을 진행하며, 추론 시 8비트 대칭 양자화 기법을 사용한다. 또한, 추론 시 배치 정규화 연산을 생략하기 위해 합성곱과 배치 정규화를 결합하는 방법을 제시하고[9], 이를 기반으로 한 하드웨어 구조를 제안한다. 본 논문에서 제안하는 배치 정규화가 내장된 하드웨어 구조는 하드웨어 기술 언어(Verilog-HDL)를 사용하여 구현하였으며, 제안하는 하드웨어 구조를 검증하기 위해 선행 연구인 C 기반 검증 모델[10]을 사용하여 출력 결과를 비교 평가하였다.

본 논문의 2장에서는 배치 정규화의 필요성 및 합성곱과 배치 정규화를 결합한 알고리즘에 관해 설명하며, 3장에서는 본 논문에서 제안하는 배치 정규화가 내장된 하드웨어 구조에 대해 설명한다. 이후 4장에서는 제안하는 하드웨어 구조의 출력 연산 검증 결과를 제시하며, Samsung 28nm 공정 Synopsys 툴을 사용하여 합성한 결과를 제시한다. 5장에서는 본 논문의 결론과 향후 연구과제에 대해 논한다.

Ⅱ. 관련 연구

2.1 배치 정규화의 필요성

신경망 학습 과정에서는 첫 번째 층의 매개변수가 변화함에 따라 현재 계층 입력 분포가 달라지며, 특히 심층 신경망의 경우 달라진 분포가 누적되어 네트워크 계층이 깊어질수록 그래디언트 소실로 인해 신경망이 수렴되는 문제가 발생한다. 배치 정규화는 이러한 공변량 시프트 현상을 최소화하기 위해 네트워크 각 계층 단위로 정규화를 적용하는 기법이다. 이를 통해 학습 시 높은 학습률(Learning rate)로 적용할 수 있어 기존 대비 정확도 향상이 가능하다[11]. 본 논문에서는 배치 정규화의 장점을 활용하고 합성곱 신경망의 추론 정확도 향상을 위해 합성곱과 배치 정규화를 결합한 알고리즘을 하드웨어로 설계한다.

2.2 합성곱과 배치 정규화를 결합한 알고리즘

합성곱과 배치 정규화는 식 (1)과 (2)로 표현할 수 있다. 식 (1)은 합성곱 연산을 위해 입력 데이터(x)와 가중치(w)간의 곱연산 및 바이어스와의 덧셈 연산 과정을 의미한다. 식 (2)는 입력 데이터의 평균과 분산을 활용한 배치 정규화 연산 과정을 의미한다. 식 (1) 및 (2)에 적용되는 가중치, 스케일, 시프트의 경우 학습이 가능한 변수이며, 학습 과정을 거쳐 안정화 된 고정값이므로 추론 시 이를 결합하여 수식화가 가능하다. 식 (3)은 식 (2)의 입력 데이터(x_i)가 식 (1)의 합성곱 연산의 출력 데이터(x_i)임을 활용하여 두 식을 결합한 후, 이를 합성곱 연산과 바이어스가 결합된 식 (1)의 형태로 표현하였다. 식 (4)는 이를 $\overline{w}, \overline{b}$로 치환한 것을 나타낸다.

$$x_i = x \times w + b \tag{1}$$

$$y = \gamma \times \frac{x_i - \mu_b}{\sigma_b^2 + \epsilon} + \beta \quad (2)$$

$$y = x \times \frac{\gamma \times w}{\sqrt{\sigma_b^2 + \epsilon}} + \left(\beta - \frac{\gamma \times \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} + b\right) \quad (3)$$

$$\overline{w} = \frac{\gamma}{\sqrt{\sigma_b^2 + \epsilon}} \times w \,, \overline{b} = \beta - \frac{\gamma \times \mu_b}{\sqrt{\sigma_b^2 + \epsilon}} + b \quad (4)$$

그림 1은 합성곱과 배치 정규화를 결합한 알고리즘을 적용한 합성곱 신경망 예시이며, 식 (4)의 치환된 값을 적용함으로써 배치 정규화 연산이 생략 가능하다. 그림 2는 식 (4)의 결과인 치환된 가중치 및 바이어스를 통해 합성곱과 배치 정규화를 결합한 알고리즘을 연산하는 과정이다.

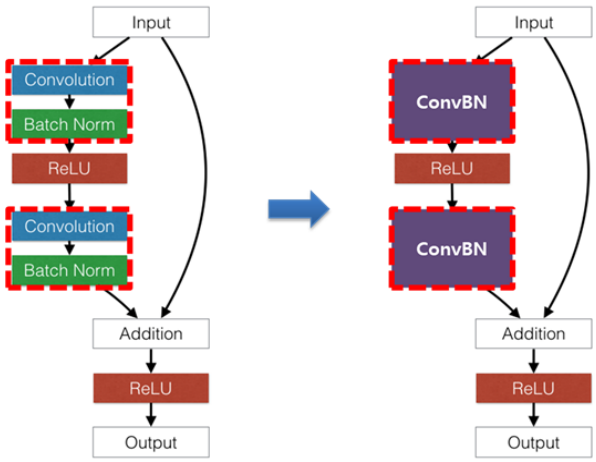

그림 1. 합성곱과 배치 정규화를 결합한 CNN 예시
Fig. 1. CNN example of the combining convolution and catch normalization

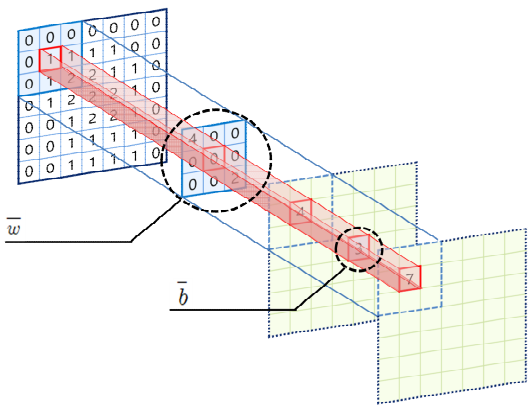

그림 2. 치환값이 적용된 CNN 연산 과정에 관한 삽화
Fig. 2. An illustration of the CNN computation process with permutations applied

III. 제안하는 하드웨어 구조

그림 3은 배치 정규화가 내장된 하드웨어 구조이다. 제안하는 하드웨어 구조는 입출력 특징맵을 저장하기 위한 메모리 영역(Memory area), 데이터 연산을 위한 배치 정규화가 내장된 연산 모듈, 바이어스 전체 컨트롤러 시스템(Controller system)으로 구성되어 있다.

3.1 메모리 영역

메모리 영역은 그림 3의 하단에 표시된 Weight, Bias, Feature Map을 의미한다. 각각의 영역은 제안하는 하드웨어 구조에서 합성곱과 배치 정규화를 결합한 알고리즘을 연산하기 위해 수식 4에서 도출한 치환된 가중치($\overline{w}$), 바이어스($\overline{b}$), 입력 데이터(x)를 저장하기 위한 영역으로 구분된다. 또한, 메모리 영역의 크기를 감소 시켜 연산 속도를 향상하는 것과 동시에 출력 데이터의 정확도를 유지하기 위해 32비트 부동소수점 데이터를 8비트 대칭 양자화 기법(Symmetric quantization)을 적용하여 각 메모리 영역에 크기를 감소시켰다[12]. 8비트 대칭 양자화 연산의 경우 바이어스가 포함된 합성곱의 형태이므로 완전 연결 계층(Fully-connected layer)에 사용되는 양자화 연산을 적용했다[13].

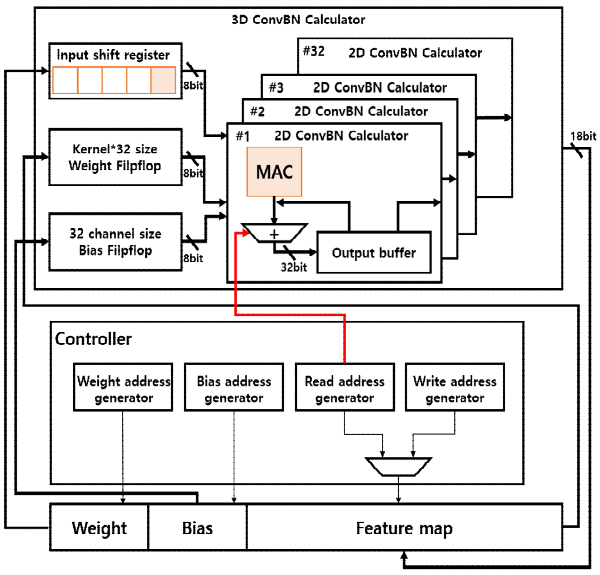

그림 3. 제안하는 하드웨어 구조
Fig. 3. Proposed hardware architecture

3.2 제안하는 연산 모듈

그림 4는 배치 정규화가 내장된 연산 모듈을 나타내며, 식 (3) 및 (4)에서 가중치와 바이어스의 치환되는 과정을 통해 배치 정규화 연산이 생략 가능하다. 연산을 위해 필요한 입력 및 가중치 데이터는 메모리에 순차적으로 시프트 되어 저장된다.

입력 및 가중치 데이터는 8비트 대칭 양자화 기법에 의해 8비트 정수 데이터가 저장되며, 내부 카운터를 사용하여 그림 5와 같이 채널 단위로 가중치를 저장하도록 하였다. 2D 연산기에서는 입력 및 가중치 데이터가 곱연산을 수행하며, 채널 단위의 가중치 데이터이기 때문에 순차적인 곱연산이 가능하다. 이러한 순차적인 곱연산은 단일 연산뿐만 아니라 병렬 연산 수행이 가능하며, 연산 출력 채널 및 연산 모듈의 크기를 고려하여 최대 32채널 병렬 연산을 수행하도록 설계하였다. 곱연산을 수행한 결과는 출력 버퍼에 저장되며, 저장된 결과는 합성곱 연산 방식에 따라 누적 연산을 반복 수행한다.

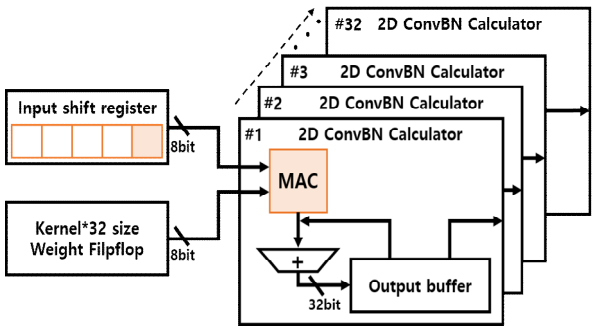

그림 4. 배치 정규화가 내장된 연산 모듈
Fig. 4. Calculation module with built-in batch normalization

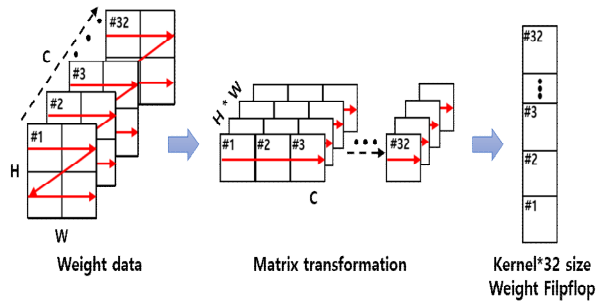

그림 5. 32채널 병렬 연산을 위한 가중치 행렬 변환 과정
Fig. 5. Weight matrix transformation process for 32-channel parallel operation

3.3 컨트롤러 시스템

컨트롤러 시스템은 입력, 가중치, 바이어스, 출력 데이터의 순차적인 읽고 쓰기를 위한 주소 생성기이다. 컨트롤러 시스템은 연산 모듈의 연산을 위해 그림 6과 같이 유한상태기계 기반 FSM(Finite-State Machine)을 활용하였으며, 각 주소 생성기들의 동작을 순차적으로 제어한다.

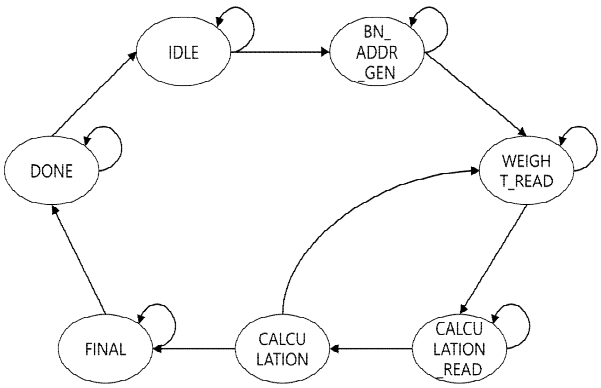

그림 6. 유한상태기계 기반 컨트롤 시스템
Fig. 6. FSM-based control system

연산 수행 전 입력 및 바이어스 데이터는 연산을 위한 모든 데이터의 주소를 생성하여 각 메모리 영역으로 전송하며, 가중치는 출력 채널 크기만큼 커널 단위 데이터의 주소를 생성하여 메모리 영역으로 전송한다. 이후 곱연산을 수행하며, 저장한 데이터 크기만큼의 연산 수행이 완료될 경우 다음 곱연산 및 누적 연산을 위해 연산을 정지하고 가중치 주소 생성기를 동작 시켜 다음 데이터를 메모리 영역에 전송하는 과정을 반복 수행한다. 모든 연산 수행이 완료될 경우 출력 주소 생성기를 통해 출력 데이터를 순차적으로 저장한다.

3.4 바이어스 연산을 위한 멀티플렉서

합성곱과 배치 정규화를 결합한 알고리즘은 출력 특징맵 크기에 비례한 배치 정규화 연산이 생략 가능하나, 출력 채널 크기에 비례한 바이어스 덧셈 연산을 추가적으로 수행해야 한다. 이를 위해 그림 7과 같이 입력 주소 생성기와 멀티플렉서를 사용하여 내부 컨트롤 신호를 적용하였다.

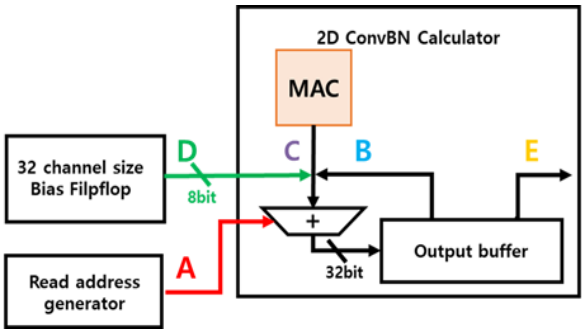

그림 7. 바이어스 연산을 위한 멀티플렉서
Fig. 7. Multiplexer for bias calculation

입력 주소 생성기를 활성화 신호로 사용하여 입력 특징맵의 마지막 채널의 데이터가 메모리 영역으로 전송될 경우 누산기에 바이어스 활성화 신호를 전송함으로써 마지막 채널에 대한 누적 연산 수행 시 바이어스와 동시에 덧셈 연산을 수행하도록 설계하였다.

그림 8은 바이어스 연산을 위한 멀티플렉서에 활성화 신호를 인가하는 과정에 관한 시뮬레이션 결과이다. A는 바이어스 활성화 신호가 활성화가 되는 타이밍이며, 마지막 채널의 입력 데이터가 가중치와 연산을 수행하는 타이밍이다. B는 이전 연산의 출력 데이터, C는 다음 연산을 위한 입력 데이터, D는 바이어스 데이터이며, E는 B, C, D의 총합을 나타낸다.

Ⅳ. 제안하는 하드웨어 구조 검증 및 평가

4.1 성능 평가 환경 및 검증 방법

본 논문에서 제안하는 배치 정규화가 내장된 하드웨어 구조는 하드웨어 합성 툴인 Xilinx vivado 환경에서 Verilog-HDL을 사용하여 연산 모듈을 구현하였다. 제안하는 하드웨어 구조 검증을 위해 얼굴 벤치마킹 데이터 세트인 LFW[14]와 CNN 모델인 ResNet18을 사용하였으며, C 기반 검증모델을 사용하여 추론 정확도를 비교 검증하였다. 검증한 결과를 기반으로 본 논문에서 제안하는 하드웨어 구조를 구현하기 위해 Verilog-HDL을 사용하여 모델링하였으며, 소프트웨어 출력 결과와 비교 검증을 수행하여 제안하는 하드웨어 구조의 정확성을 판별하였다.

4.2 실험 결과 및 성능 평가

표 1은 제안하는 하드웨어 구조 검증을 위해 소프트웨어로 검증한 결과이다. 이를 위해 C 기반 검증 모델을 활용하였으며, 기존 모델과 결합된 알고리즘을 기반으로 수정된 모델의 추론 정확도를 평가하였다. 표 1과 같이 총 10번 테스트한 결과 기존 대비 평균 99.68%의 높은 정확도를 보임을 확인하였다.

표 1. 결합한 알고리즘 검증 결과
Table 1. Combined algorithm verification results

Num	Before(%)	After(%)	Accuracy(%)	Average(%)
1	0.996605	0.996664	99.99408	99.68
2	0.988652	0.992728	99.58772	
3	0.99457	0.993876	99.93022	
4	0.988453	0.986866	99.83945	
5	0.988943	0.989195	99.97452	
6	0.988538	0.986645	99.80851	
7	0.97109	0.964886	99.36113	
8	0.996804	0.997559	99.92426	
9	0.987402	0.987703	99.96952	
10	0.981286	0.983164	99.80862	

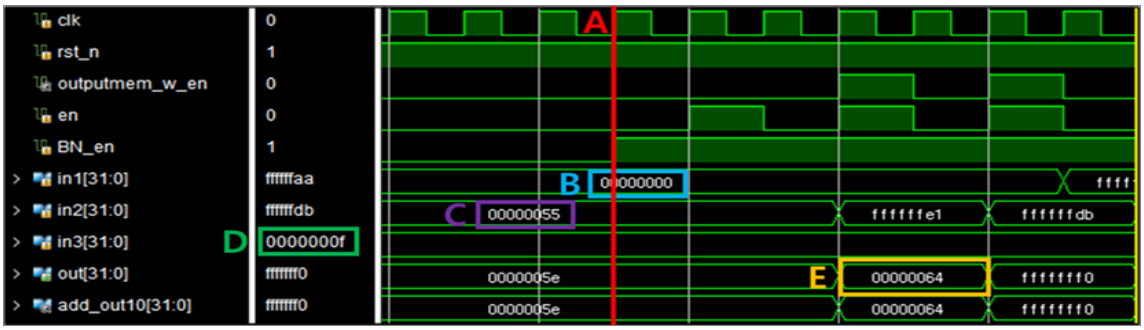

그림 8. 시뮬레이션 결과
Fig. 8. Simulation results

그림 9와 10은 합성곱과 배치 정규화를 결합한 알고리즘을 Verilog-HDL을 사용하여 모델링한 후 연산 출력 결과를 소프트웨어 출력 결과와 비교 검증을 수행한 결과이다. 제안하는 하드웨어 구조의 중간 및 최종 출력 결과값을 비교 검증하였으며, 1채널 단위와 32채널 단위의 출력 결과값을 비교 평가함으로써 정상 동작함을 확인하였다.

표 2는 제안하는 하드웨어 구조의 정확한 자원 이용량 및 전력량 측정을 위해 Samsung 28-nm 공정을 사용하여 합성을 수행한 결과이다.

표 2. 하드웨어 자원 이용량 측정 결과
Table 2. Measurement results of hardware utilization

Scenario - cmos28lpp / base_rvt_ss / 0p900v_125c			
	Number		Area(μm^2)
Port	12520	Combinational	12536
Net	29567	Buf/Inv	1034
Cell	16112	Nonconbinational	5675
Combinational logic	13763	Macro/Black box	0
Sequanial cell	2243	Total cell	18212
Buf/Inv	2842	Core	35019

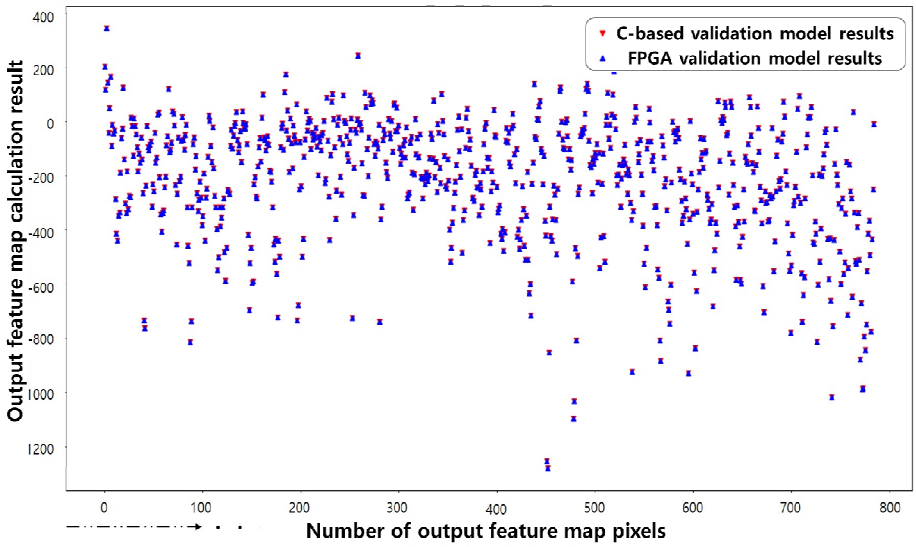

그림 9. 제안하는 하드웨어 구조 검증 결과 (1채널)
Fig. 9. Verification results of the proposed hardware architecture (1-channel)

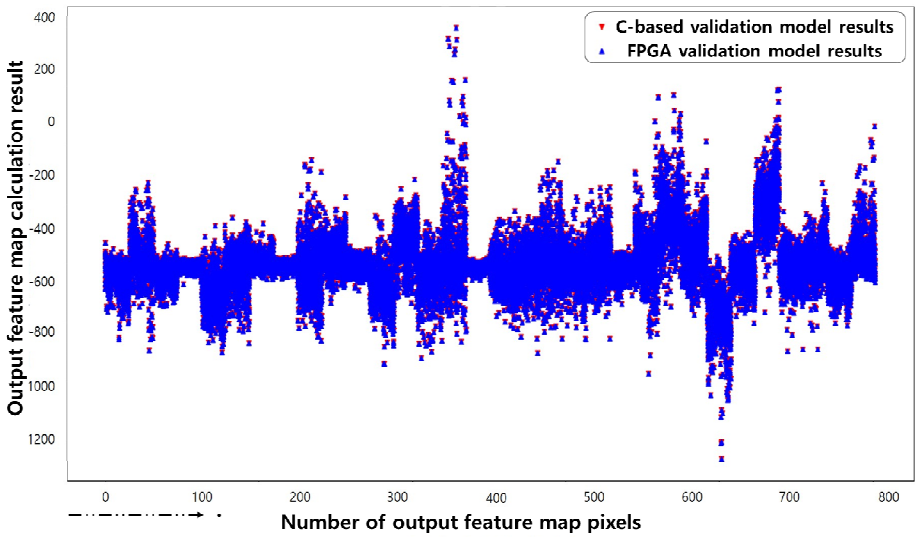

그림 10. 제안하는 하드웨어 구조 검증 결과 (32채널)
Fig. 10. Verification results of the proposed hardware architecture (32-channel)

V. 결론 및 향후 과제

본 논문에서는 합성곱 신경망의 추론 정확도 향상을 위해 배치 정규화가 내장된 하드웨어 구조를 제안하였다. 제안하는 하드웨어 구조는 내부 컨트롤 신호를 활용하여 배치 정규화 연산을 생략하였다.

제안하는 하드웨어 구조는 합성곱과 배치 정규화를 결합한 알고리즘을 소프트웨어로 검증을 수행하였으며, Verilog-HDL로 모델링한 후 정확한 검증을 위해 하드웨어 연산의 1채널 단위와 32채널 단위의 출력 결과값을 비교 평가함으로써 정상 동작을 확인하였으며, Samsung 28-nm 공정을 사용하여 합성을 진행하였다.

제안하는 하드웨어 구조는 다양한 합성곱 신경망에 유연하게 적용할 수 있을 것으로 기대되며, 이를 기반으로 저전력, 고속 네트워크 기반 AI 전용 가속기 개발에 사용될 수 있을 것으로 기대된다.

References

[1] I. S. Eo, H. S. Mo, Y. M. Park, and U. J. Han, "HPC technology trends seen through SC20", Electronic communication trend analysis, Vol. 36, No. 3, pp. 133-144, Jun. 2021. https://doi.org/10.22648/ETRI.2021.J.360313.

[2] G. I. O, S. E. Kim, Y. H. Bae, and Y. S. Kwan, "Artificial Intelligence Neuromorphic Semiconductor Technology Trend", Electronic communication trend analysis, Vol. 35, No. 3, pp. 76-84, Jun. 2020. https://doi.org/10.22648/ETRI.2020.J.350308.

[3] M. Y. Lee, J. Chung, J. H. Lee, J. H. Han, and Y. S. Kwon, "Trends in AI processor technology", Electronics and Telecommunications Trends, Vol. 35, No. 3, pp. 66-75, Jun. 2020. https://doi.org/10.22648/ETRI.2020.J.350307.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR), Las Vegas, NV, USA, pp. 770-778, Jun. 2016. https://doi.org/10.1109/CVPR.2016.90.

[5] S. Santurkar, D. Tsipras, A. Llyas, and A. Madry, "How does batch normalization help optimization?", Proceedings of the 32nd international conference on neural information processing systems, Red Hook, NY, USA, pp. 2488-2498, Dec. 2018.

[6] H. Yonekawa, H. Haruyoshi, and H. Nakahara. "On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an FPGA", 2017 IEEE International Parallel and Distributed Processing Symposium Workshops(IPDPSW), Lake Buena Vista, FL, USA, pp. 98-105, May 2017. https://doi.org/10.1109/IPDPSW.2017.95.

[7] Y. J. Wai, Z. bin Mohd Yussof, and S. I. bin Salim, "Fixed point implementation of tiny-yolo-v2 using opencl on fpga", International Journal of Advanced Computer Science and Applications, Vol. 9, No. 10, pp. 506-512, Sep. 2018. http://dx.doi.org/10.14569/IJACSA.2018.091062.

[8] T. Sledevic, "Adaptation of convolution and batch normalization layer for CNN implementation on FPGA", 2019 Open Conference of Electrical, Electronic and Information Sciences (eStream), IEEE, Vilnius, Lithuania, pp. 1-4, Apr. 2019. https://doi.org/10.1109/eStream.2019.8732160.

[9] C. Luo, Y. Wang, W. Cao, P. H. W. Leong, and L. Wang, "RNA: An Accurate Residual Network Accelerator for Quantized and Reconstructed Deep Neural Networks", 28th International Conference on Field Programmable Logic and Applications (FPL), Dublin, Ireland, pp. 60-603, Aug. 2018. https://doi.org/10.1109/FPL.2018.00018.

[10] Seong-Ho Jeon and Seung-Ho Ok, "A C-based Verification Model for FPGA-based CNN Accelerators", 2020 Conference on Korean Institute of Information Technology(KIIT), Cheongju, Korea, pp. 528-530, Dec. 2020.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", In: International conference

on machine learning, PMLR, Lille, France, pp. 448-456, Jun. 2015.

[12] Y. Wu and C. T. Huang, "Efficient Dynamic Fixed-Point Quantization of CNN Inference Accelerators for Edge Devices", 2019 International Symposium on VLSI Design, Automation and Test, pp. 1-4, Apr. 2019. https://doi.org/10.1109/VLSI-DAT.2019.8742040.

[13] X. Zhao, Y. Wang, C. Liu, C. Shi, K. Tu, and L. Zhang, "BitPruner: Network Pruning for Bit-serial Accelerators" 2020 57th ACM/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, pp. 1-6, Mar. 2020. https://doi.org/10.1109/DAC18072.2020.9218534.

[14] http://vis-www.cs.umass.edu/lfw/ [accessed: Oct. 29. 2019]

저자소개

전 성 호 (Seong-Ho Jeon)

2021년 2월 : 동의대학교 로봇·자동화공학전공(공학사)
2021년 3월 ~ 현재 : 동의대학교 인공지능학과(석사과정)
관심분야 : On-device AI, FPGA, NPU, Edge AI-System

옥 승 호 (Seung-Ho Ok)

2008년 2월 : 경북대학교 전자공학과(공학석사)
2014년 2월 : 경북대학교 전자공학부(공학박사)
2014년 3월 ~ 2017년 8월 : 삼성전자 책임연구원
2017년 9월 ~ 현재 : 동의대학교 로봇·자동화공학전공 조교수
관심분야 : Robot Vision, Edge AI, SoC, VLSI