ISSN:1226-7244 (Print)
ISSN:2288-243X (Online)
논문번호 17-03-10
j.inst.Korean.electr.electron.eng.Vol.21,No.3,244~247,September 2017
http://dx.doi.org/10.7471/ikeee.2017.21.3.244

GP-GPU를 이용한 보행자 추론 CNN

Pedestrian Inference Convolution Neural Network Using GP-GPU

정 준 모*

Junmo Jeong*

Abstract

In this paper, we implemented a convolution neural network using GP-GPU. After defining the structure, CNN performed inferencing using the GP-GPU with 256 threads, which was the previous study, using the weight obtained from the training. Training used Intel i7-4470 CPU and Matlab. Dataset used Daimler Pedestrian Dataset. The GP-GPU is controlled by the PC using PCIe and operates as an FPGA. We assigned a thread according to the depth and size of each layer. In the case of the pooling layer, we used over warpping pooling to perform additional operations on the horizontal and vertical regions. One inferencing takes about 12 ms.

요 약

본 논문에서는 GP-GPU를 활용한 보행자 추론 컨볼루션 뉴럴 네트워크를 구현했다. CNN은 구조를 정한 후, 학습에서 얻은 가중치를 이용해 기존 연구인 256개의 스레드를 가지는 GP-GPU를 활용해 추론을 수행했다. 학습에는 Inter i7-4470 CPU와 Matlab을 사용했다. Dataset은 Daimler Pedestrian Dataset을 사용했다. GP-GPU는 PCIe를 이용해 PC로부터 제어를 받으며, FPGA로 동작한다. 각 레이어의 depth와 size에 따라 스레드를 할당했다. 풀링 레이어의 경우는 over warpping pooling을 사용했기 때문에 횡영역과 종영역에 추가적인 연산을 수행했다. 한 번의 추론에는 약 12ms가 걸린다.

Key words : Pedestrian Inference, CNN, GP-GPU, Multithread, SIMT

Ⅰ. 서론

최근 CNN(Convolution Neural Network)는 영상 분류, 객체 검출, 자연어 인식 등의 영역에서 사용되고 있다. 이미지와 같은 2차원 데이터를 분류 하는 데 CNN이 많이 사용 되고 있으며, 2차원 데이터 형태로 변환해, 음성 등에도 사용하기도 한다.

CNN은 정방향으로 수행할 경우 추론을 수행하고 역방향으로 수행할 경우 학습을 수행한다.

CNN은 뉴런들이 모여 레이어를 구성하고, 레이어의 크기와 역할을 바꿔 네트워크를 구성한다. 레이어는 컨볼루션 레이어와 풀링 레이어, 풀리 커넥티드 레이어로 나눌 수 있다. 다음 그림 1은 뉴런과 레이어와 네트워크를 단순하게 도식화했다.

* Dept. of Electronics Engineering, Seokyeong University
e-mail:jjmo@skuniv.ac.kr, tel:02-940-7732
※ Acknowledgment
This research was supported by Seokyeong University in 2017
Manuscript received, Sep.13, 2017; revised, Sep.18, 2017; accepted, Sep.20, 2017

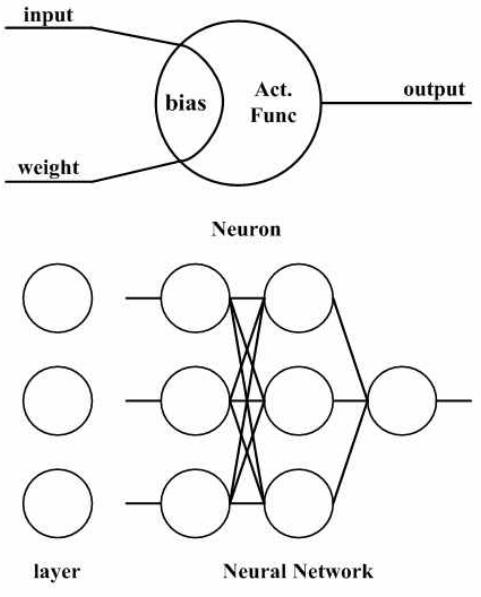

Fig. 1. Neuron, layer, neural Network
그림 1. 뉴런, 레이어, 뉴럴 네트워크

보행자 인식은 ADAS(Advanced Driver Assistance System, 운전자 보조 시스템)에서 중요한 역할을 한다. 보행자 인식 알고리즘은 HOG(Histogram of Oriented Gradient), Cascade HOG, ChnFtrs 등이 있으며 현재는 CNN을 사용해 보행자를 검출하는 알고리즘으로 많이 사용한다[1][2].

본 논문에서는 컨볼루션 뉴럴 네트워크를 학습과 추론을 다른 하드웨어를 사용했으며, 보행자를 추론하는 컨볼루션 뉴럴 네트워크를 구현했다.

II. 본론

1. 컨볼루션 뉴럴 네트워크를 활용한 보행자 추론

Daimler Pedestrian Dataset을 사용해 컨볼루션 뉴럴 네트워크에 학습 시켜 보행자를 추론했다. 학습 이미지는 흑백 이미지이며, 학습에는 Intel i7-4470 CPU과 Matlab을 사용했다[3].

보행자는 일반적으로 세로영역이 긴 이미지 이므로 그에 따라 컨볼루션 뉴럴 네트워크도 정방형의 형태가 아닌 세로가 긴 형태의 구조를 갖는다. 보행자 추론 컨볼루션 뉴럴 네트워크의 구조는 다음 그림 2와 표 1과 같다.

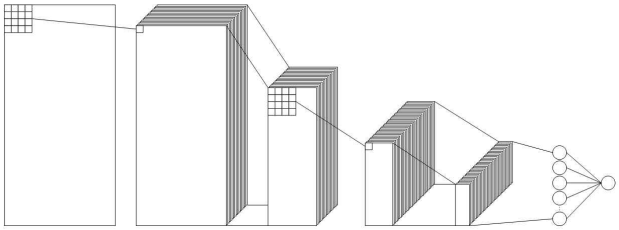

Fig. 2. Architecture of pedestrian Inference CNN
그림 2. 보행자 추론 컨볼루션 뉴럴 네트워크의 구조

Table 1. Architecture of pedestrian Inference CNN
표 1. 보행자 추론 컨볼루션 뉴럴 네트워크의 구조

Layer	Depth	Size	Kernel
Input	1	16×32	-
Conv. 1	16	13×29	4×4 Stride 1
Pool. 1	16	7×15	3×3 Stride 2
Conv. 2	32	4×12	4×4 Stride 1
Pool. 2	32	2×6	3×3 Stride 2
FC. 1	-	64	-
FC. 2	-	1	-

컨볼루션 레이어는 총 두 개를 사용하며 커널 사이즈는 4×4이며, Stride는 1이다.

풀링 레이어에는 Over wrapping pooling 기법을 사용하여 첫 번째 컨볼루션 레이어에는 2 pixel 만큼, 두 번째 컨볼루션 레이어에는 1 pixel 만큼 0으로 채워줘야 한다. 각 커널의 사이즈는 3×3이며, Stride는 2이다.

풀리 커넥티드 레이어는 두 개로, 64개의 뉴런과 1개의 뉴런을 가진 레이어로 구성했다. 최종 뉴런에서는 보행자가 맞는지 확률을 계산한다.

2. 멀티 스레드 GP-GPU

256개의 멀티 스레드 GPU는 총 256개의 스레드를 가지며, 4개의 스트림 멀티 프로세서, 16개의 스트림 프로세서를 가진다. 스트림 멀티 프로세서 당 스레드는 64개 이며, 스트림 프로세서 당 스레드는 16개이다. 워프 스케쥴러는 스트림 프로세서의 갯 수 만큼 가지고 있다. 스트림 프로세서 하나에는 ALU, FPU, 오퍼랜드 컬렉터를 한 개씩 가지고 있다. 다음 내용을 정리한 것이 표 2에 해당한다.[4][5]

Table 2. Architecture of pedestrian Inference CNN
표 2. 멀티 스레드 GP-GPU의 구성 요소

Stream multiple processor (SM)	Stream processor (SP)	Warp Scheduler
4	16	16
Max threads	threads per SM	threads per SP
256	64	16
ALU per SP	FPU per SP	OC per SP
1	1	1

3. 멀티 스레드 GP-GPU를 활용한 보행자 추론 컨볼루션 뉴럴 네트워크

256개의 스레드를 가지는 GP-GPU를 활용해 컨볼루션 뉴럴 네트워크를 구현했다.

첫 번째 컨볼루션 레이어의 크기는 16개의 depth에 13×29의 아웃맵 크기를 가진다. 아웃맵의 원소 하나를 스레드 하나가 계산한다. 즉 377개의 아웃맵의 원소를 256개의 스레드가 처리하려면 모든 thread가 처리 된 후 121개의 스레드가 컨볼루션 연산을 한번 더 수행해야 한다. 또한 피처맵의 갯수 만큼 연산을 더 수행해야 하므로 하나의 스레드는 16개 혹은 32개의 아웃맵의 원소를 연산해야 한다. 다음 그림 3은 첫 번째 컨볼루션 레이어의 연산에 스레드가 어떻게 할당 되었는지를 보여준다.

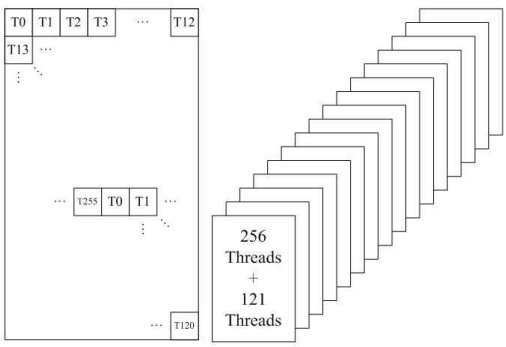

Fig. 3 Assigning the first convolution layer thread
그림 3. 첫 번째 컨볼루션 레이어의 스레드 할당

풀링 레이어 연산의 경우는 아웃맵의 크기 만큼 연산 하되 Over wrapping pooling 기법이 들어갔기 때문에 횡영역과 종영역에 연산을 더 해줘야 한다. 첫 번째 풀링 레이어를 예를 들어 설명하면 아웃맵의 크기가 105이지만 Over wrapping Pooling 연산을 위해 84개의 스레드를 먼저 연산한 후 횡영역과 종영역에 대해 연산을 추가로 수행한다. 횡영역은 96개의 스레드를 사용하고 종영역은 240개의 스레드를 사용했다. 다음 그림 4는 두 번째 풀링 레이어의 연산에 스레드가 어떻게 할당되었는지를 보여준다.

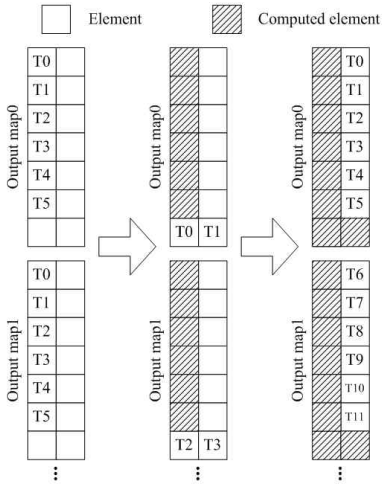

Fig. 4 Assigning the second pooling layer thread
그림 4. 두 번째 풀링 레이어의 스레드 할당

그림 5와 같이 풀리 커넥티드 레이어의 경우는 뉴런의 개수에 맞게 스레드를 할당 했다.

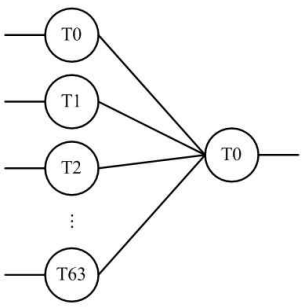

Fig. 5 Assigning the fully connected layer thread
그림 5. 풀리 커넥티드 레이어의 스레드 할당

다음 표 3은 스레드 당 연산하는 아웃맵의 원소에 대한 내용이다.

Table 3. Thread per output map element
표 3. 스레드 당 아웃 맵 원소 연산 갯 수

Layer	Element	Thread per Element	
Conv. 1	16×377 (6,032)	0 ≤ tid ≤ 120	32
		121 ≤ tid ≤ 255	16
Pool. 1	16×105 (1,680)	0 ≤ tid ≤ 83	16
		0 ≤ tid ≤ 95	1
		0 ≤ tid ≤ 239	1
Conv. 2	32×48 (1,536)	0 ≤ tid ≤ 47	32
Pool. 2	32×12 (384)	0 ≤ tid ≤ 5	32
		0 ≤ tid ≤ 160	1
		0 ≤ tid ≤ 64	1
FC. 1	64	0 ≤ tid ≤ 63	1
FC. 2	1	tid = 1	1

4. 실험 결과

실험은 FPGA로 동작하는 GP-GPU를 활용하여 보행자를 추론하였다. 실험에 사용한 FPGA 보드는 Xilinx VC-707 FPGA Board이다. 한 번의 추론을 수행하는데 걸린 시간은 표 4와 같다[6].

Table 4. Processing time
표 4. 연산 시간

Layer	Processing time
Conv. 1	1.73 ms
Pool. 1	1.5 ms
Conv. 2	5.23 ms
Pool. 2	1.2 ms
FC. 1	2.3 ms
FC. 2	0.13 ms
Total	12.09 ms

표 5는 기존의 보행자를 인식하는 다른 연구결과와 비교했다[7][8][9]. 다른 연구결과와 비교를 위해 픽셀 당 시간으로 비교했다.

Table 5. Processing time per pixel
표 5. 픽셀 당 연산 시간

	Image size	Processing time per pixel (us)
CNP	42X42	56
DC-CNN	20X20	105
CNNP	16X32	37
Proposed GPU	16X32	23

첫 번째 컨볼루션 레이어와 두 번째 컨볼루션 레이어는 인맵의 16개와 커널의 16개를 메모리로부터 읽어와 아웃맵을 생성하기 때문에 연산이 많으며, 이미지가 크기 때문에 thread가 많다. 그에 따라 연산시간이 길다. 풀링 레이어는 9개의 인맵을 읽어와 가장 큰 값을 아웃맵을 생성하기 때문에 연산이 적다. 그래서 컨볼루션 레이어보다 연산시간이 적게 든다. 두 개의 풀리 커넥티드 레이어는 뉴런의 개수 만큼의 스레드를 할당했기 때문에 연산 시간은 길지 않다.

III. 결 론

본 논문에서 보행자를 추론하는 컨볼루션 뉴럴 네트워크를 Daimler Pedestrian Dataset을 활용하여 학습시켰다. 학습에는 Intel i7-4470 CPU와 Matlab을 사용하여 학습했다. 추론은 FPGA로 동작하는 256개의 thread를 가지는 GP-GPU으로 수행했다. 각 레이어는 아웃맵의 원소의 갯 수 만큼의 스레드를 할당해서 연산했다. GP-GPU를 활용한 보행자 추론 컨볼루션 뉴럴 네트워크의 추론 연산 시간은 12.09 ms가 걸렸다.

References

[1] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans.* on PAMI, 2012.
DOI : 10.1109/TPAMI.2011.155
[2] http://darkpgmr.tistory.com/
[3] http://www.gavrila.net/Research/Pedestrian_Detection/Daimler_Pedestrian_Benchmark_D/Daimler_Mono_Ped__Detection_Be/daimler_mono_ped__detection_be.html
[4] Dohyun Kim, Chi-yong Kim, “Design of a SIMT architecture GP-GPU using Tile based on graphic pipeline structure”, *j.inst.Korean.electr.electron.eng,* Vol 20. No 1.
DOI : 10.7471/ikeee.2016.20.1.075
[5] Kwanho Lee, Chi-yong Kim, “A Design of a High Performance Stream Processor without Superscalar Architecture”, *j.inst.Korean.electr.electron.eng,* Vol 21. No 1.
[6] https://www.xilinx.com
[7] Farabet, Clément, Cyril Poulet, Jefferson Y. Han, and Yann LeCun. "Cnp: An fpga-based processor for convolutional networks." *In 2009 International Conference on Field Programmable Logic and Applications*, pp. 32-37. IEEE, 200
DOI : 10.1109/FPL.2009.5272559
[8] Chakradhar, Srimat, et al. "A dynamically configurable coprocessor for convolutional neural networks." *ACM SIGARCH Computer Architecture News.* Vol. 38. No. 3.ACM, 2010.
DOI : 10.1145/1816038.1815993
[9] Heekyeong Jeon, “A Design of Convolutional neural network Processor for ADAS”, *Master thesis*, Seokyeong University, 2017