Implemetation of Image Classification CNN using Multi thread GPU

Seong-Hyeon Han
Dept. of Computer Engineering
Seokyeong University
Seoul, Korea
cromcruach@skuniv.ac.kr

Kwang-Yeob Lee
Dept. of Computer Engineering
Seokyeong University
Seoul, Korea
kylee@skuniv.ac.kr

***Abstract*—This study implemented an image classification CNN using a multi-thread GPU. For the CNN, the CIFAR10 dataset was used, and the multi-thread GPU had 256 threads. Using the 256 threads limited to each layer, allocation and parallel processing were conducted. The image classification CNN took 807 ms for computation.**

Keywords—CNN, Multi-thread, GPU, Image classification

I. INTRODUCTION

In recent years, CNN has been used for various purposes. Especially, it has been widely used for image recognition and it shows good performance when applied to two-dimensional data such as images.

The forward pass of CNN performs inferencing and the backward pass training. The latter takes more than twice longer computation time than the former[1]. Therefore, if inferencing and training are conducted in separate hardware, it may result in good performance. For instance, Google TPU performs only inferencing and show better performance than others using a GPU[2].

In this study, training was carried out using NVIDIA GTX 1070 and for inferencing, an object recognition CNN was implemented using a GPU with 256 threads.

II. IMAGE CLASSIFICATION CNN

The image classification CNN was trained on the CIFAR-10 data set[3]. The CIFAR-10 data set has a RGB-3 channel with a size of 32x32 and ten images consisting of the airplane, automobile, bird, cat, deer, dog, frog, house, ship, and truck.

The image classification CNN used RGB-3 channel images with a size of 32x32 as input images. The first convolution layer has 16 depths and a size of 32x32. The second convolution layer has the same structure as the first and convolution padding was used for this. The first pooling layer has 16 depths and a size of 16x16. The third pooling layer has a size of 16x16 and 32 depths. The second pooling layer has a size of 8x8 and 32 depths. The first fully-connected layer has 2048 neurons. The second fully-connected layer has 1024 neurons, and the third-fully connected layer has 512. The output layer has 10 layers. Table 1 shows the architecture of the CNN.

TABLE I. IMAGE CLASSIFICATION CNN ARCHITECTURE

Layer	Depth	Size
Input layer	3	32X32
1st Convolution layer	16	32X32
2nd Convolution layer	16	32X32
1st Pooling layer	16	16X16
3rd Convolution layer	32	16X16
2nd Pooling layer	32	8X8
1st Fully connected layer	-	1,024
2nd Fully connected layer	-	512
Output layer	-	10

For training, the CUDA of NVIDIA GTX 1070 was used. To allocate blocks and threads to the CUDA, the performance of the GPU must be checked first. Table 1 shows the specifications of NVIDIA GTX 1070[4].

TABLE II. NVIDA GTX 1070 SPECIFICATION

memory	SP	max SMs	max blocks	threads per SP	threads per SM	threads per SP	max threads
8GB	1,920	15	30	16	2,048	1,024	30,720

In training, as many blocks as depths and as many threads as sizes were used. For instance, 16 blocks and 1024 threads were used on the first convolution layer. Figure 1 shows the allocation of blocks and threads.

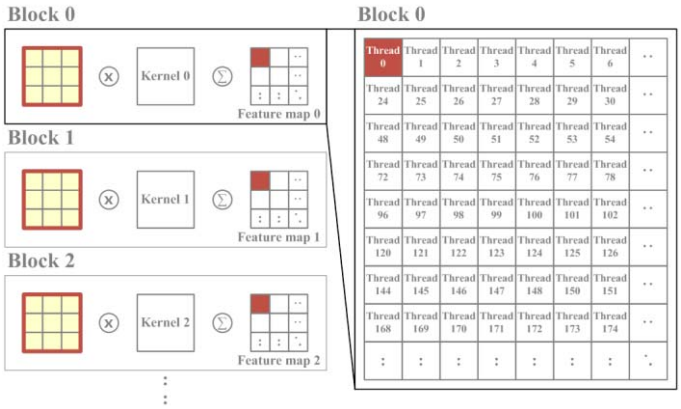

Fig. 1. Allocation of Blocks and Threads

III. MULTI THREAD GPU

The GPU with 256 threads has four stream multiprocessors (SM). Each SM has four stream processors (SP) and each SP has 16 threads. The 16 threads of an SP are collectively called "warp." So the GPU has 16 warps. Figure 2 shows the structure of the GPU.

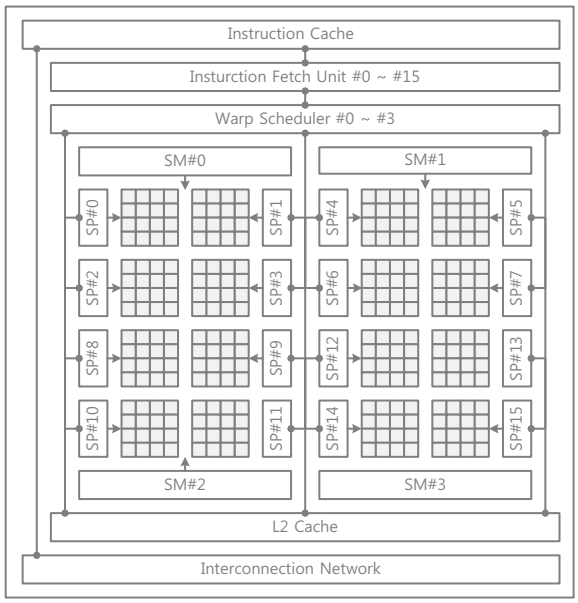

Fig. 2. GPU Architecture'

It is a warp scheduler that manages warps. Activated warps send activation information to the instruction fetch through the warp scheduler; the instruction fetch receives instructions through the instruction cache; and the instructions are delivered to the activated warps through the instruction fetch. Figure 3 shows this process.

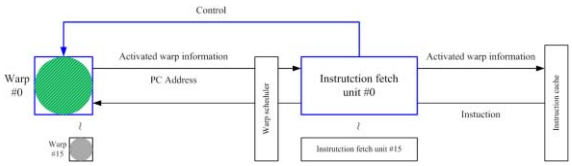

Fig. 3. Processing of Warp scheduler

IV. EXPERIMENTAL RESULT

This study implemented an image classification CNN using a multi-thread GPU. The Fully connected layer with a large number of parameters took a long time to read and write memory, which was found to be related to high computational complexity. Table 3 shows the image classification CNN processing time.

TABLE III. PROCESSING TIME

Layer	Processing time (ms)
1st Convolution layer	9
2nd Convolution layer	51
1st Pooling layer	1
3rd Convolution layer	10
2nd Pooling layer	1
1st Fully connected layer	498
2nd Fully connected layer	236
Output layer	1
Total	807

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP). (No. 2016-0-00204, Development of mobile GPU hardware for photo-realistic realtime virtual reality) and This work was supported by the "Development of the high efficiency GPU structure for artificial intelligence" of the Korea Evaluation Institute of Industrial Technology(KEIT) granted financial resource from the Ministry of Trade, Industry & Energy, Republic of Korea(No. 10067394)

REFERENCES

[1] https://github.com/Bvlc/caffe/issues/1317

[2] Jouppi, Norman P., et al. "In-datacenter performance analysis of a tensor processing unit.", the 44th International Symposium on Computer Architecture (ISCA), 2017

[3] https://www.cs.toronto.edu/~kriz/cifar.html

[4] http://www.nvidia.co.kr/graphics-cards/geforce/pascal/kr/gtx-1070

[5] Kwanho Lee, "A Design of a SIMT architecutre based GP-GPU for parallel acceleration of algorithms", Master thesis, Seokyeong University, 2017