# Characterizing Convolutional Neural Network Workloads on a Detailed GPU Simulator

Kwanghee Chang, Minsik Kim, Kyungah Kim and Won Woo Ro
School of Electrical and Electronic Engineering
Yonsei University
Seoul, Korea
{kwanghee.chang, minsik.kim, kyungah.kim, wro}@yonsei.ac.kr

***Abstract*— Recent frameworks on convolutional neural networks (CNNs) such as Caffe and MXNet have focused primarily on being compatible with CUDA software and hardware application. However, it was designed for GPU architecture of compute capability 3.0 and above. Therefore, it needs verification of function to perform GPGPU-Sim which is implemented as NVIDIA compute capability devices 2.x. We developed a framework which can make inferencing AlexNet on GPGPU-Sim. We also analyze the execution results of the GPGPU-Sim. The number of lines in one set of the L1 data cache is sensitive to influence performance of AlexNet inference**

***Keywords—CUDA; Compute Capability; AlexNet; GPU; GPGPU-Sim***

## I. INTRODUCTION

Convolutional neural network (CNN) is based on the concept of the human visual cortex to analyze the image data. CNN is widely used in the image classification and video analysis. Although the algorithm requires a large amount of computations, it can be processed efficiently on GPUs [1].

In this paper, we design a CNN application to perform inferencing image on a GPU micro-architecture simulator. By analyzing the micro-architectural simulation, we can figure out the operation process of CNN on GPUs and computational bottleneck. In order to resolve the observed problems, cache and DRAM configuration can be proposed. This study is based on GPGPU-Sim [2] which is a cycle-level many-core simulator compute capability 2.x.

However, current neural network frameworks such as Caffe and MXNet are designed to work at compute capability 3.0 and above. The main difference of compute capability between 3.0 and lower version is the number of CUDA thread and batch sizes due to hardware constraints which is shown in the Table 1. In addition, 32-bit registers per multiprocessor of 3.0 version is 2x bigger than that of compute capability 2.x [3]. For this reason, current framework cannot operate by using GPGPU-Sim v3.2.2 which is made to be suitable for compute capability 2.0.

We purpose new neural network frameworks that operate on compute capability 2.0, which can be worked on the GPGPU-Sim. We analysis performance as changing L1 data cache size using GPGPU-sim. The number of lines in one set of the L1 data cache is sensitive to influence performance of AlexNet inference

TABLE 1. TECHINICAL SPECIFICATIONS PER COMPUTE CAPABILITY

| Technical Specifications | Compute Capability | |
|---|---|---|
| | *2.1* | *3.0* |
| Maximum x-dimension of a grid of thread blocks | 65535 | $2^{31}$-1 |
| Maximum number of blocks per multiprocessor | 8 | 16 |
| Maximum number of warps per multiprocessor | 48 | 64 |
| Maximum number of threads per multiprocessor | 1536 | 2048 |
| Number of 32-bit registers per thread block | 32K | 64K |

## II. PROPOSED METHOD

### A. Forward Convolution function

We design forward convolution layer algorithm that input 4 types of data: input, bias, weight and stride.

$$\text{width of output} = \frac{\text{width of input} + \text{pad} * 2 - \text{kernel}}{\text{stride}} + 1 \quad (1)$$

As in (1), when the fractional part has remainder, the data loss can be occurred. In order to prevent the problem, we design new zero padding function that only add zero to the end of rows and columns. Unlike the max pooling function where zero values are added to all rows and columns of the matrix.

We use the weight and bias data which is trained from MXNet. We implemented the GPU code that performs convolution function which consists of matrix multiplier and reduction to obtain the output value in parallel.

### B. Max pooling function

In AlexNet, max pooling function is used after the 2nd, 3rd, and 6th convolution function. Max pooling function is designed as two sub-functions. The first sub-function optimizes the activation map as the same size with the input data to prevent loss of input data. The other selects the largest number in the kernel range.

### C. Fully Connected function

We design fully connected function that adds the result of matrix multiplication of input data and the bias as same as forward convolution function. However, unlike the Forward convolution function that use only values of the kernel from input data for matrix multiplication, the entire input data is used for matrix multiplication in fully connected function. Therefore, the number of weight data is determined by the product of the number of input data and the number of output data.

TABLE 2. EXECUTION TIME RATIO

| Kernel name | Execution time (%) | |
|---|---|---|
| | *Visual Profiler* | *GPGPU-Sim* |
| Fully Connected layer | 49.199 | 57.884 |
| Convolutional layer | 50.619 | 42.070 |
| Max pooling layer | 0.187 | 0.046 |

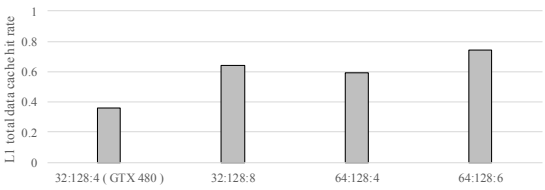


Figure 1. L1 cache hit rate on various L1 data cache config

### D. GPGPU-Sim

Visual Profiler is current widely used profiler offered by NVIDIA to measure performance [4]. Visual Profiler supports dependency analysis and provides vital feedback for users in order to optimize CUDA C/C++ applications.

Table 2 shows ratio of execution time of functions used in GPU with Visual Profiler and GPGPU-Sim. Though the difference of ratio between Visual Profiler and GPGPU-Sim is almost 10%, it is allowable range that can be considered sufficiently.

## III. PERFORMANCE EVALUATION

### A. Accuracy

In order to perform inference of AlexNet, we design framework that uses 224 sized input data, weight and bias that are pre-trained in MXNet. The accuracy result is 51.3% which is equal to MXNet classification. The result of accuracy is affected by that the second highest probability label *Walker hound* is very similar to a correct answer *beagle* in human view.

### B. Cache hit rate

We analysis L1 data cache hit ratio as changing data cache size in GPGPU-Sim. Larger data cache size can resolve the bottleneck of AlexNet. However, increasing the data cache size is not a recommended solution because it increases the performance but also increases the cost. To resolve this problem, it is necessary to determine which part of the data cache size to increase is a solution, by changing each of the three parameters (number of sets, batch size and number of lines in one set) respectively. Figure 1. shows that execution time is sensitive to the number of lines in one set of the L1 data cache.

### C. Execution time

We analyze the execution time in the same way as measuring the L1 data cache hit rate. We observe that the execution time decreases as size of data cache increase. Moreover, we analyze which parameter of L1 data cache affects execution time as depicted in Figure 2.. It shows that execution time is sensitive to the number of lines in one set of the L1 data cache.

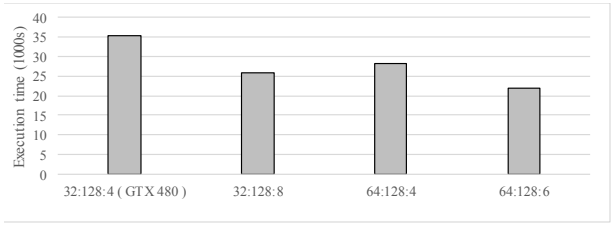


Figure 2. Execution time on various L1 data cache config

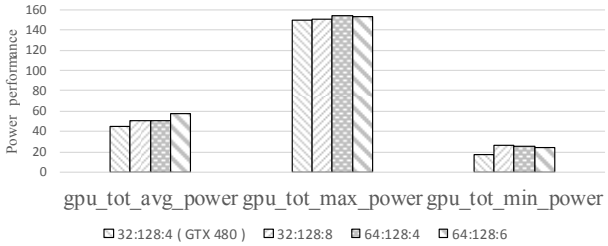


Figure 3. Execution power performance on various L1 data cache config

### D. Power Consumption

In addition to performance of IPC, power consumption is also very important to design GPU architecture. GPUWattch is a power model that addresses all of the aforementioned requirements [5]. It enables widespread architecture-level power analysis and optimization. Figure 3. shows the power consumption as changing L1 data cache size.

## IV. CONCLUSION

In this paper, we design framework for AlexNet inference that run AlexNet on GPGPU-Sim instead of GPU. We analyze data cache sensitivity of AlexNet by using GPGPU-Sim. As our result, the number of lines in one set of the L1 data cache is sensitive to influence performance of AlexNet inference. This means that developer can easily design gpu architecture to optimize for AlexNet as well as other neural networks.

## REFERENCES

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP)(No. 2016-0-00140, Development of Application Program Optimization Tools for High Performance Computing Systems), by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP)(No. NRF-2015R1A2A2A01008281)

## REFERENCES

[1] Nvidia, C. U. D. A. "Toolkit Documentation (2014)." (7).

[2] Bakhoda, Ali, et al. "Analyzing CUDA workloads using a detailed GPU simulator." Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on. IEEE, 2009.

[3] NVIDIA. 2015. CUDA 7.0 Performance Report. http://developer.download. nvidia.com/compute/cuda/compute-docs/cuda-performance-report.pdf

[4] Janssen, C. "The visual profiler." URL http://aros. ca. sandia. gov/cljanss/perf/vprof (1999).

[5] Leng, Jingwen, et al. "GPUWattch: enabling energy optimizations in GPGPUs." ACM SIGARCH Computer Architecture News. Vol. 41. No. 3. ACM, 2013.