OpenCL-Darknet: An OpenCL Implementation for Object Detection

Yongbon Koo
Autonomous Driving System Research Group
ETRI
Daejeon, Republic of Korea
ybkoo@etri.re.kr

Chayoung You
Research and Development Laboratory
RTst
Daejeon, Republic of Korea
yous@rtst.co.kr

SungHoon Kim
Autonomous Driving System Research Group
ETRI
Daejeon, Republic of Korea
saint@etri.re.kr

Abstract— **Object detection is a technology that deals with recognizing classes of objects and their location. It is used in many different areas, such as in face-detecting digital cameras, surveillance tools, or self-driving cars. These days, deep learning-based object detection approaches have achieved significantly better performance than the classic feature-based algorithms. Darknet [1] is a deep learning-based object detection framework, which is well known for its fast speed and simple structure. Unfortunately, like many other frameworks, Darknet only supports NVIDIA CUDA [2] for accelerating its calculations. For this reason, a user has only limited options for graphic card selection. OpenCL™ (open computing language) [3] is an open standard for cross-platform, parallel programming of heterogeneous systems. It is available not only for CPUs, GPUs (graphics processing units), but also for DSPs (digital signal processors), FPGAs (field-programmable gate arrays) and other hardware accelerators. In this paper, we present the OpenCL-Darknet, which transforms the CUDA-based Darknet into an open standard OpenCL backend. Our goal was to implement a deep learning-based object detection framework that will be available for the general accelerator hardware and to achieve competitive performance compared to the original CUDA version. We evaluated the OpenCL-Darknet in AMD R7-integraged APU (accelerated processing unit) with OpenCL 2.0 and AMD Radeon RX560 with OpenCL 1.2 using a VOC 2007 dataset [4]. We also compared its performance with the original Darknet for NVIDIA GTX 1050 with CUDA 8.0 and cuDNN 6.0.**

Keywords- OpenCL, deep learning, object detection

I. INTRODUCTION

Object detection is a technology that deals with recognizing classes of objects and their location. Many areas, including face detection, surveillance, and a self-driving car's vision system, need object detection as their core functionalities. Classic object detection systems use the scheme of feature-based methods, such as Haar features extraction [5], and histogram of oriented gradients (HOG) and linear support vector machine (SVM) algorithms [6]. These days, deep learning is applied to object detection and has archived significant precision improvement. To execute deep learning algorithms, various deep learning frameworks are released. Darknet [1] is one of the most widely used frameworks for object detection. It is well known for its fast processing speed and simple architecture. Unfortunately, like many other frameworks, Darknet only supports NVIDIA CUDA [2] for accelerating its deep learning calculations. For this reason, a user has only limited options for graphic card selection, and this leads her to having restricted system configurations.

OpenCL™ (open computing language) [3] is a low-level open standard API for cross-platform, parallel programming of heterogeneous systems. It is similar to CUDA but available not only for CPUs and GPUs (graphics processing units), but also for DSPs (digital signal processors), FPGAs (field-programmable gate arrays) and other hardware accelerators. In this paper, we present the OpenCL-Darknet, which transforms a CUDA-based Darknet into OpenCL backend. Our goal was to implement a deep learning-based object detection framework that is available for general accelerator hardware and achieve competitive performance compared to the original CUDA version. We evaluated the OpenCL-Darknet in AMD R7-integrated APU (accelerated processing unit) with OpenCL 2.0 and AMD Radeon RX560 with OpenCL 1.2 using the VOC 2007 dataset [4]. We also compared its performance with the original Darknet for NVIDIA GTX 1050 with CUDA 8.0 and cuDNN 6.0.

Our main contributions are the following:

- Implementing a deep learning-based object detection framework that is available for OpenCL, the open standard for heterogeneous architectures.
- Demonstrating the validity of the framework on two disparate GPU configurations, AMD R7 APU and AMD Radeon RX560, and comparing its performance with NVIDIA GTX 1050.

II. RELATED WORK

OpenCL caffe [7] transformed the CUDA-based caffe deep learning framework into OpenCL-based framework. They described OpenCL porting strategies that guaranteed algorithm convergence and examined the performance bottlenecks. They also proposed three key optimization techniques, kernel caching to avoid OpenCL online compilation overheads, a batched manner data layout scheme to boost data parallelism and multiple command queues to boost task parallelism.

clSpMV [8] optimized the SpMV (sparse matrix vector multiplication) kernel, which is a key computation in linear algebra, in OpenCL. They proposed a new sparse matrix format, the Cocktail Format, to take advantage of the

2375-9356/18/$31.00 ©2018 IEEE
DOI 10.1109/BigComp.2018.00112

strengths of many different sparse matrix formats. They found that clSpMV provided the best representations of the input sparse matrices on both NVIDIA and AMD platforms.

There are several tools to help porting CUDA programs to OpenCL [9][10][11]. In this study, we did not use any of them because they are general tools and not compatible with large programs. However, we plan to utilize them to generate reference codes.

III. PORTING STRATEGIES

The Darknet framework was originally written in C and CUDA. CUDA and OpenCL have different hardware abstractions, memory managements, and data transfer mechanisms. First, we targeted a layer-wise porting. Each layer in Darknet has both CPU and GPU implementation, and the user can select one with compiler options. In order to maintain each layer's correctness, we collected temporary data of a layer's input and output by running CPU implementations with several images. Then, we built a test framework to verify the OpenCL implementation of each layer with these data. Next we replaced CUDA kernels with OpenCL ones.

For convolutional layers, we used a clBLAS's clblasSgemm function for GEMM (general matrix-matrix multiplication) implementation [12]. We also used clRNG to generate random stream array in parallel on an OpenCL device [13]. Basic arithmetic functions were also ported to OpenCL. They included filling, scaling, and normalizing matrices. We needed to transform simple layers, such as activation to OpenCL to reduce unnecessary data transfers between a host and devices. Not only deep learning layers but also pre- and post-processing functions needed to be implemented.

In this study, we only implemented techniques for a generic inference situation. Almost all the embedded devices only needed to run inference processes. Large-scale clusters were responsible for the training.

IV. PERFORMANCE EVALUATION

We evaluate the performance of the OpenCL-Darknet on an AMD embedded board with various GPU configurations. Because deep learning processing is not only affected by GPU acceleration, but also by CPU computation, we retained the same hardware configuration except for discrete GPU cards. AMD R7 was integrated within an AMD Embedded RX-421BD chip and called the APU (accelerated processing unit). In an APU environment, CPU cores and GPU cores are connected internally and share the main memory. AMD Radeon RX560 and NVIDIA GTX 1050 are discrete GPU cards attached via a PCI-Express interface. We turned off an integrated GPU when we were using a discrete GPU.

TABLE I shows our test platform specifications. TABLE II, TABLE III and TABLE IV show the specifications for target GPUs and software configurations.

The detection model we tested was YOLOv2 and TinyYOLO [14] for a publicly available PASCAL VOC2007 (Visual Object Classes Challenge 2007) [4] test data set. VOC2007 contains 4952 images and 20 classes to be recognized. In Fig. 1, you can see one of detection results of VOC2007 with YOLOv2 using the OpenCL-Darknet. YOLOv2 and TinyYOLO were trained with VOC2007 and VOC2012 training data sets, and had a mAP (mean average precision) on VOC2007 of 78.6% and 57.1% respectively.

As shown in Fig. 2 and Fig. 3, we also compared the performance of Darknet with CUDA with or without cuDNN. They are average time consumed per image, and are

TABLE I. EVALUATION BOARD SPECIFICATION

CPU	AMD Embedded RX-421BD 4 cores / 2100MHz / 28nm
RAM	16GB DDR4
Storage	256GB SSD / SATA III interface
Interface	PCI Express 3.0

TABLE II. CONFIGURATION FOR AMD R7-INTEGRATED APU

Hardware Specification	8 compute units 512 shading units 800 MHz 819 GFLOPS
OS	Ubuntu 14.04.5
Kernel	3.16.0-77-generic
Graphic Driver	fglrx-15.302.1101
OpenCL	2.0 (AMDAPPSDK-3.0)
clBLAS	2.12
clRNG	1.0.0-beta

TABLE III. CONFIGURATION FOR AMD RADEON RX 560

Hardware Specification	16 compute units 1024 shading units 1275 MHz 2611 GFLOPS 1GB GDDR5 128bits
OS	Ubuntu 16.04.2
Kernel	4.4.0-97-generic
Graphic Driver	AMDGPU 17.10
OpenCL	1.2
clBLAS	2.12
clRNG	1.0.0-beta

TABLE IV. CONFIGURATION FOR NVIDIA GTX 1050

Hardware Specification	640 CUDA cores 1354 MHz 1862 GFLOPS 2GB GDDR5 128bits
OS	Ubuntu 14.04.5
Kernel	4.4.0-97-generic
Graphic Driver	375.66
CUDA	8.0
cuDNN	6.0

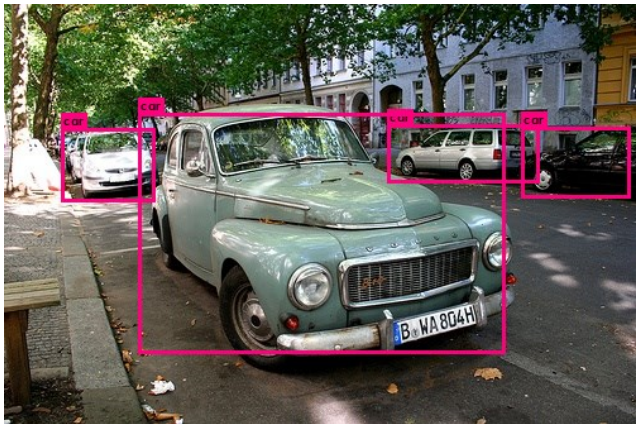

Figure 1. Example of an object detection using YOLOv2.

normalized with cuDNN performance. Fig. 2 is the result of YOLOv2, and Fig. 3 is of TinyYOLO. We investigated the performance in two steps. The first was inferencing with a deep learning network, and the second was calculating the bounding boxes. Currently, OpenCL-Darknet's performance is far less than its competitor's CUDA version. Compared with CUDA and cuDNN, the OpenCL-Darknet had a performance gap of 2.5x to 4.4x for YOLOv2, and of 3.2x to 6.4x for TinyYOLO.

The goal of this comparison was to investigate further optimization directions. As you can see in Fig. 4, the deep learning network calculation is the dominant time consumer. It consumed 10x to 23x the time than the processing of bounding boxes. Thus, reducing deep learning network computing time is the most important optimization direction. We expect more sophisticated optimization will reduce the current performance gap. Furthermore, using OpenCL will facilitate applying FPGA (field- programmable gate arrays), and will reduce the cost compared with GPUs. We will study these aspects as future work.

V. CONCLUSIONS

In this paper, we presented the OpenCL-Darknet, which transforms the CUDA-based Darknet – a deep learning-based object detection framework – into an open standard OpenCL backend. Our goal was to implement a deep learning-based object detection framework for general accelerator hardware and achieve competitive performance over the original CUDA version. We evaluated the OpenCL-Darknet in AMD R7-integrated APU with OpenCL 2.0 and AMD Radeon RX560 with OpenCL 1.2 using the VOC 2007 dataset. We also compared its performance with the original Darknet for NVIDIA GTX 1050 with CUDA 8.0 and cuDNN 6.0.

As future work, we will optimize the OpenCL-Darknet using profiling tools to identify the bottleneck. OpenCL is generally known as slower than CUDA [15]. However, we can make a more cost-effective deep learning framework with OpenCL because OpenCL supports various types of hardware. We plan to facilitate FPGA to reduce the cost to run object detection systems.

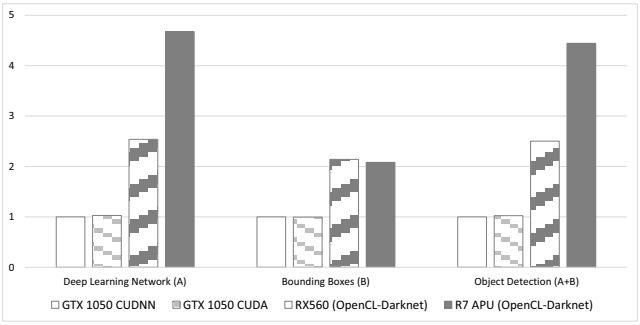

Figure 2. Average processing time of YOLOv2.

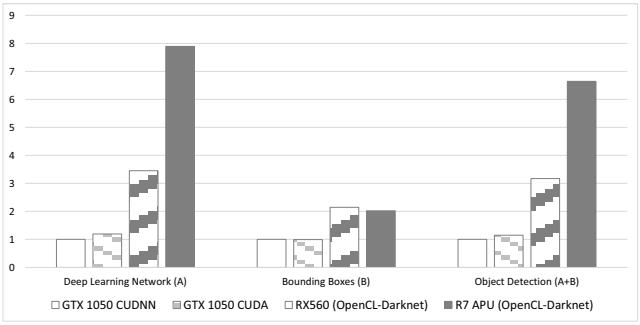

Figure 3. Average processing time of TinyYOLO.

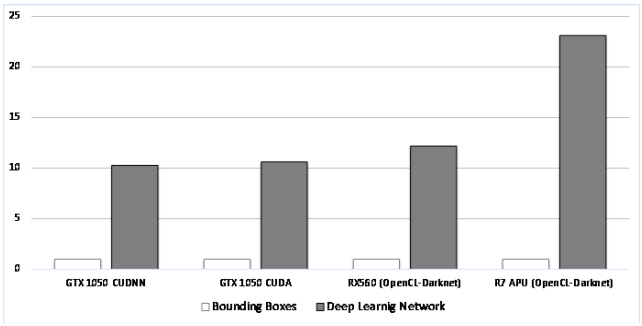

Figure 4. Comparison between bounding box processing and deep learning network processing.

ACKNOWLEDGMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. 2016-0-00004, Development of Driving Computing System Supporting Real-time Sensor Fusion Processing for Self-Driving Car).

REFERENCES

[1] J. Redmon, Darknet, https://github.com/pjreddie/darknet

[2] NVIDIA, CUDA Technology, http://www.nvidia.com/CUDA

[3] The Khronos Group Inc, OpenCL, https://www.khronos.org/opencl

[4] M. Everingham, L. Van~Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2007 (VOC2007) results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

[5] P. Viola, and M. Jones, “Robust real-time object detection,” International Journal of Computer Vision, vol. 57, issue 2, 2004, pp. 137-154, https://doi.org/10.1023/B:VISI.0000013087.49260.fb

[6] N. Dalal, and B. Triggs, “Histograms of oriented gradients for human detection,” Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 05), San Diego, CA, USA, 2005, pp. 886-893. vol. 1, doi: 10.1109/CVPR.2005.177 2005.

[7] J. Gu, Y. Liu, "OpenCL caffe: Accelerating and enabling a cross platform machine learning framework," Proc. the 4th International Workshop on OpenCL (IWOCL 16), Vienna, Austria, 2016, pp. 8:1-8:5, doi: 10.1145/2909437.2909443.

[8] B-Y. Su, and K. Keutzer, “clSpMV: A Cross-Platform OpenCL SpMV Framework on GPUs,” Proc. the 26th ACM International Conference on Supercomputing (ICS 12), Venice, Italy, 2012, pp. 353-364, doi:10.1145/2304576.2304624.

[9] M.J. Harvey, and G. De Fabritiis, “Swan: A tool for porting CUDA programs to OpenCL,” Computer Physics Communications, vol. 182, issue 4, Apr. 2011, pp. 1093-1099, https://doi.org/10.1016/j.cpc.2010.12.052

[10] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, and J. Dongarra, "From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming," Parallel Computing, vol. 38, issue 8, Aug. 2012, pp. 391-407, doi:10.1016/j.parco.2011.10.002.

[11] G. Martinez, M. Gardner, and W. Feng, “CU2CL: A CUDA-to-OpenCL translator for multi- and many-core architectures,” Proc. 2011 IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS 11), Dec. 2011, pp. 300-307, doi: 10.1109/ICPADS.2011.48.

[12] clMathLibraries, clBLAS, https://github.com/clMathLibraries/clBLAS

[13] clMathLibraries , clRNG, https://github.com/clMathLibraries/clRNG

[14] J. Redmon, and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” arXiv preprint arXiv:1612.08242, 2016.

[15] K. Karimi, N. G. Dickson, and F. Hamze, “A Performance comparison of CUDA and OpenCL,” arXiv preprint arXiv:1005.2581, 2010.