Object Detection with Deep Learning on Drive PX2

Duong Nguyen-Ngoc Tran, Huy-Hung Nguyen, Long Hoang Pham, and Jae Wook Jeon
Department of Electrical and Computer Engineering, Sungkyunkwang University, Korea
duongtran@skku.edu, huyhung91@skku.edu, phlong@skku.edu, jwjeon@yurim.skku.ar.kr

Abstract

Many successful deep learning networks for object detection have been proposed in recent years, but direct comparison is difficult due to different resolutions, training validation data, and frameworks used to train these networks. In particular, we use DRIVE PX2 as a reference computing platform, which is manufactured by NVIDIA Corporation for development of autonomous vehicles and evaluate the performance of methods of autonomous driving on ARM-based embedded processing cores and Tegra-based embedded graphics processing units (GPUs).

Keywords: object detection, autonomous car, drive px2, edge devices,

1. Introduction

Autonomous vehicle navigation to a given destination and advanced driver support will be one of the next great milestones for the automotive industry. The discovery and implementation of appropriate algorithms are integral parts of meeting this challenge. These facts led to attract several researchers and automakers to focus on the evolutionary process of the automotive industry, where the development of Advanced Driving Assistance Systems (ADAS) will allow the technology to mature and gradually lead to the development of autonomous vehicles. Perception techniques, and more precisely those related to lane detection, are not only a first step towards semi-autonomous driving but will then open the way to fully advanced automated vehicles. However, there are some challenges in the development and inference of available deep learning model into the edge device:

1. The higher accuracy model would require a more powerful processing device, and a large internal memory.
2. The deep learning model must be run stably on a small device over a long time.
3. The implementation step must balance the dilemma between accuracy detection and speed performance.

To meet the computational demands of deep learning applications, supplier offerings are usually based on various application-specific integrated circuits (ASICs) such as field programmable gate arrays (FPGAs), or on digital signal processors (DSPs) or graphics processing units (GPUs). Since GPUs are proprietary systems, the vendors generally do not disclose internal information. The prerequisite for setting up the emulation environment is the extraction of intrinsic properties from real hardware. In this study, we use the embedded computing platform marketed by NVIDIA as the DRIVE PX Parker AutoChauffeur. The PX2 incorporates multicore processors along with multiple GPUs to accelerate computer vision (CV) applications. To conclude, we illustrate a summary of the contributions as follows:

1. We modify state-of-the-art object detection methods and make them more efficient and suitable for Drive PX2.
2. We show the time-consuming comparison between the methods.

The remainder of this paper is organized as follows. In Section 2, the related works will be discussed. The detail of the experimental method is described and discussed in Section 3. In Section 4, the authors show qualitative results and benchmark results to corroborate the advantages of the new method. Conclusions are drawn in Section 5.

2. Related work

Figure 1: NVIDIA Drive PX2 AutoChauffeur

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2020R1A2C3011286).

Table 1: Detail of Drive PX2 iGPU and dGPU

	Integrated GPU (iGPU)	Discrete GPU (dGPU)
Computing units	256 CUDA Cores	1152 CUDA Cores
Accessible memory	6668 MBytes	3840 MBytes
Memory bandwidth	≈50 GBytes/s	≈80 GBytes/s
Shared L2 cache size	512 KBytes	1,024 KBytes

A modern detector is usually composed of two parts: a backbone that is pre-trained on ImageNet, and a head is used to predict classes and bounding boxes of objects. A high performancehigh-performance computer could run the backbone as VGG [1], ResNet [2], or DenseNet [3]. For the small and edge devices, their backbone could be small such as SqueezeNet [4], MobileNet [5], or ShuffleNet [6]. For the head part, it is usually classified into two types, i.e., one-stage object detector and two-stage object detector. The most representative two-stage object detector is the R-CNN [7] series. It is also possible to make a two-stage object detector an anchor-free object detector, such as RepPoints. As for the one-stage object detector, the most representative models are YOLO [8], SSD [9], and RetinaNet [10].

In some cases, models should run based on the support of the development company, such as the version of operation, driver software, and the open source of the community. Specifically, we must build all of the framework and open source for testing on the old operation system and NVIDIA Driver if the company and community discontinue support.

3. Methodology

In this section, we provide detailed descriptions of our hardware platform, the DRIVE PX2 [11], the deep learning models, and the details of input image.

3.1. Architecture

For deep learning development in a practical way, NVIDIA shows that their device is more advanced in both training and inference [12]. Beside the Jetson device for research and testing, they offer the Drive device, which has a discrete GPU for processing, which will not affect the operation system while inferencing the model. NVIDIA Drive PX2 is a computing platform for autonomous driving, which promises a powerful and easy-to-develop platform for algorithm research and rapid prototyping. In the AutoChauffeur configuration, it consists of two Parker SoCs, two discrete GPUs (dGPU) and an Aurix TC297. Each Parker has one smaller integrated GPU (iGPU). Some relevant differences

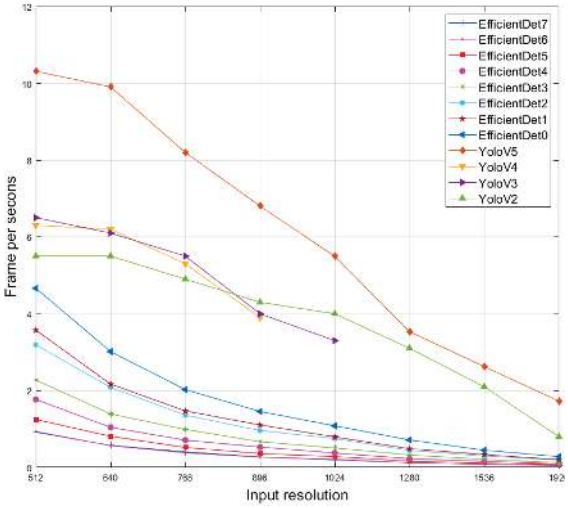

Figure 2: Speed performance of all methods on Drive PX2

can be found in Table 1. The PX2 contains two identical and independent "Parker" systems on a chip (SoCs), called Tegra A and B, connected by a 1 Gbps Controller Area Network (CAN) bus. We show one of the identical halves of the PX2. Each SoC contains six CPUs (four ARMv8 Cortex A57 cores, two ARMv8 Denver 2.0 cores), and an integrated Pascal GPU (iGPU). Each Parker SoC also connects to a discrete Pascal GPU (dGPU) over its PCIe x4 bus. While the iGPU and CPU cores share main DRAM memory, the dGPU has its own DRAM memory. Memory transfers between CPU memory and dGPU memory use the PCIe bus. The remainder of this paper, including the design descriptions and evaluation experiments, use only one of the PX2's two independent systems: iGPU vs. dGPU. As previously mentioned, the PX2 has two types of GPUs – the iGPU and the dGPU – using the same Pascal architecture. While they share an architecture, they share little else. Consider their specifications as shown in Table 1. The dGPU has more cores, faster memory, and a larger L2 cache. The iGPU lags behind, advantaged only by its access to the larger, albeit shared, main memory DRAM banks. In this paper, we consider and use the Drive PX2 for testing and a benchmark current state-of-the-art method, which are flexible in the covert model for implementation the in-edge device. The Drive PX2 has two main boards, each board has two GPU processes for running a Linux OS and running the application, which is shown in detail in Table 1. In the Drive PX2, operation system is Ubuntu 16.04, CUDA 9.2 and CuDNN 5.

3.2. Models

Beside the Jetson version in NVIDIA, which is supported annually, the Drive PX2 is rarely upgraded to the new version of both the operation system and

a) v5_512 b) v5_1920

c) D7_512 d) D7_1920

Figure 3: The visualization result of EfficientDet and YOLOv5 in 512 and 1920 resolution input image.

the driver of CUDA and TensorRT. We must build and compile the framework and open source for testing. For the EfficientDet, Mobilenet, R-CNN, we used the Tensorflow 2.2.0 framework built from the source, after upgrading the GCC from 5.4 to 7.3. For the YOLOv5, we run it on Pytorch 1.5.1. Lastly, for the YOLOv2, YOLOv3, and YOLOv4, we compile the Darknet.

3.3. Datasets

We consider several images of varying resolution from 512, 640, 712, 863, 1024, 1536, and 1920 of KATECH (Korea Automotive Technology Institute). The purpose for the various resolutions in the input image is that the current camera stream has various sizes, which is based on the bandwidth and the processing power of the edge device. For the small device or the low bandwidth, we should consider the low resolution for a short time consuming in both transfer and inferred image. And for the high bandwidth connection or the edge device, which has a processor capable of handling the heavy loading process, we are able to run the high-resolution model. The high and low-resolution input image has shown the difference in results. Figure 3 shows the number of detected objects on different models with the two lowest and highest resolutions.

4. Evaluation

In this section, we show and discuss the quantitative and visualization result of the models run on Drive PX2.

4.1. Various input size

We consider the Efficient-Det model and Yolo model for inference, and present the results with various input image sizes without retraining the model. As can be seen in Table 3, for the YOLO version, only version 2 with the C++ language and

Table 2: Speed performance of constant input size models

Models	PC	PX2
MobileNetV2	23	10
MobileNetV3	20	9
Fast R-CNN	16	3
Mask R-CNN	15	4

version 2 using Python with the Pytorch framework could run in high resolution at 1920 x 1920 pixels. However, the YOLOv3 and YOLOv4 could reach high resolution, because of out of memory Figure 2 shows more detail with respect to time consumption for each algorithm in Drive PX2. On the other size, of the EfficientDet with the Tensorflow framework, the latter version, the higher in accuracy. Additionally, with the high resolution, the model will still run on the Drive PX2. Moreover, for the visualization results shown in Figure 3, the bigger size results in more detected objects, which is the reason we should consider preprocessing in practice aside from the bandwidth and memory of the device.

4.2. Constant input size

The state-of-the-art models have pre-trained checkpoints, in which the input image is a fixed size. In some cases, we usually convert the model from the original lite version, which has fewer parameters and smaller size. To compare with the flexible input size model, the constant input size shows the drawback with respect to the inference with the edge device, such as the hard adaption of the hardware design in the processor.

5. Conclusion

In this paper, we illustrated the process of edge devices such as Drive PX2 in several state-of-the-art autonomous driving methods. We also benchmarked the execution speed of these networks on the NVIDIA Drive PX2. Furthermore, we showed an experimental analysis of some of the main aspects that influence the detection accuracy and speed of networks. This will help in choosing an appropriate method for deploying object detection in the real world based on requirements.

References

[1] K. Simonyan and A. Zisserman, "Very deep convo- lutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014..

[2] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV,

Table 3: Speed performance of various input size models.

	Methods	Resolutions	512	640	768	896	1024	1280	1536	1920
1	YoloV2	PX2	5.50	5.50	4.90	4.30	4.00	3.10	2.10	0.80
		PC	20.10	19.20	15.30	12.10	9.30	6.30	4.60	2.90
2	YoloV3	PX2	6.50	6.10	5.50	4.00	3.30	OOM	OOM	OOM
		PC	14.50	10.40	7.60	5.60	4.30	2.80	2.00	OOM
3	YoloV4	PX2	6.30	6.20	5.30	3.90	OOM	OOM	OOM	OOM
		PC	10.80	10.10	7.40	5.60	4.30	2.90	OOM	OOM
4	YoloV5	PX2	10.31	9.90	8.20	6.80	5.49	3.53	2.62	1.72
		PC	27.03	24.39	22.73	21.28	17.24	12.05	9.90	6.76
5	EfficientDet-0	PX2	4.66	3.01	2.02	1.46	1.08	0.71	0.45	0.28
		PC	35.25	26.74	21.99	18.31	14.50	10.34	7.50	5.06
6	EfficientDet-1	PX2	3.57	2.17	1.47	1.11	0.79	0.49	0.34	0.20
		PC	27.65	20.43	17.31	12.78	10.99	7.53	5.54	3.62
7	EfficientDet-2	PX2	3.19	2.08	1.36	0.96	0.75	0.44	0.30	0.19
		PC	25.26	19.15	15.86	12.22	10.00	6.81	4.99	3.21
8	EfficientDet-3	PX2	2.28	1.39	0.99	0.67	0.51	0.33	0.21	0.13
		PC	19.89	15.35	10.56	9.11	7.33	5.02	3.59	2.36
9	EfficientDet-4	PX2	1.77	1.04	0.71	0.53	0.38	0.23	0.16	0.09
		PC	15.51	11.06	8.79	6.76	5.46	3.58	2.58	1.67
10	EfficientDet-5	PX2	1.25	0.81	0.52	0.36	0.28	0.16	0.11	0.07
		PC	11.95	8.57	6.59	5.02	4.01	2.64	1.88	1.22
11	EfficientDet-6	PX2	0.94	0.57	0.38	0.27	0.21	0.12	0.08	0.05
		PC	9.50	6.61	4.94	3.74	2.99	1.94	1.38	0.88
12	EfficientDet-7	PX2	0.92	0.57	0.41	0.27	0.20	0.13	0.08	0.05
		PC	9.44	6.58	4.96	3.76	2.99	1.95	1.38	0.88

USA, 2016.

[3] G. Huang, Z. Liu, L. V. D. Maaten and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017.

[4] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," in *arXiv preprint arXiv:1602.07360*, 2016.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, "MobileNets: Efficient con- volutional neural networks for mobile vision applications," in *arXiv preprint arXiv:1704.04861*, 2017.

[6] N. Ma, X. Zhang, H.-T. Zheng and J. Sun, "ShuffleNetV2: Practical guidelines for efficient cnn architecture design," in *the European Conference on Computer Vision (ECCV)*, 2018.

[7] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, OH, USA, 2014.

[8] A. Bochkovskiy, C.-Y. Wang and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," in *arXiv preprint arXiv:2004.10934*, 2020.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, "SSD: Single shot multibox detector," in *the European Conference on Computer Vision (ECCV)*, 2016.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Doll´ar, "Focal loss for dense object detection," in *the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[11] "NVIDIA DRIVE™ Developer Program for DRIVE PX 2," [Online]. Available: https://developer.nvidia.com/DRIVE-PX2-program. [Accessed 04 04 2020].

[12] "NVIDIA DRIVING INNOVATION," [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/. [Accessed 04 04 2020].