RESEARCH-ARTICLE

Quantization and Acceleration of YOLOv5 Vehicle Detection Based on GPU Chips

YUXI YANG, Rutgers University–New Brunswick, New Brunswick, NJ, United States

Published: 03 July 2024

GAIIS 2024: 2024 International Conference on Generative Artificial Intelligence and Information Security
May 10 - 12, 2024
Kuala Lumpur, Malaysia

https://doi.org/10.1145/3665348.3665420
ISBN: 9798400709562

Quantization and Acceleration of YOLOv5 Vehicle Detection Based on GPU Chips

Yuxi Yang*
Department of Computer Engineering, Rutgers University, New Brunswick, America
yuxiyang0204@gmail.com

ABSTRACT

The development of Artificial Intelligence (AI) has raised new requirement for computing chips. The great amount of calculation in deep learning model needs specially designed computing platforms, which is impossible on terminal chips and needs model compression. Model quantization is one way of model compression that can reduce the amount of calculation effectively. This paper is written on the INT8 acceleration of YOLOv5 vehicle detention based on Nvidia Tesla T4 GPU: after YOLOv5 vehicle detention model is trained, one can use TensorRT to accelerate and quantize it. With MAP reducing only 0.6, the overall performance improves by 418%, and the deployable off-line model generated is only 30% of the original size.

CCS CONCEPTS

• **Hardware** → Integrated circuits; Semiconductor memory; Dynamic memory.

KEYWORDS

quantization, acceleration, NVIDIA chips, detection

ACM Reference Format:
Yuxi Yang*. 2024. Quantization and Acceleration of YOLOv5 Vehicle Detection Based on GPU Chips. In *2024 International Conference on Generative Artificial Intelligence and Information Security (GAIIS 2024), May 10–12, 2024, Kuala Lumpur, Malaysia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3665348.3665420

1 INTRODUCTION

AI technology has been a new focus of countries around the world. And AI chip is a general term referring to hardware that can accelerate AI application, specifically in deep learning based on neural network. As commonly acknowledged, "No chip, no AI." Calculation based on AI chips is an important metric of the level of AI development. At present, the four mainstream chips for AI calculation's special acceleration are GPU, ASIC, FPGA and brain-inspired chips, each one of which has its advantages and disadvantages showed as follow in Table 1

With technical becoming more mature, the application of AI chips can expand from cloud computing and big data centers to edge computing in smart homes, smart production, smart finance, etc.; they can also be put on smart terminals like smart phones, security cameras, and self-driving cars. More varieties of smart products will appear in the future with AI computing will be omnipresent.

The detection mission of deep learning is to find the objects in pictures, which requires drawing bounding boxes first. For example, in the street, further analysis is only possible after the vehicle or the person has been detected.

RCNN gives out the method of Region Proposals, that is to first search the possible region proposals selectively in the picture, and then conduct object identification for each region, which significantly increases the effectiveness of object identification and location. Nevertheless, the speed of Region Proposals is still slow, even with the subsequent Fast RCNN and Faster RCNN that constantly improves the algorithm of neural network structure targeting and region proposal.

YOLO (You Only Look Once) is an algorithm for object identification and location based on deep neural network, whose most distinguishable feature is its fast running speed allowing it to be applied in real-time systems. YOLO creatively combines the two steps of region proposal and object identification together, applying predefined ones instead of eliminating region proposals–that is, to divide the pictures into several grids to calculate their frames. It is less precise but more rapid, therefore, YOLO is now widely applied in industry.

Complicated models mean better performance, but also indicate more occupied spaces and more consumption of computing resources, which makes it hard to be used on hardware platforms. Hence, the growing depth and size of convolutional neural network has imposed a great challenge for deploying deep learning on mobiles, and the compression and acceleration of deep learning models is now a focus of academic and industrial circle.

2 RELEVANT RESEARCH

2.1 Current Situation of Chips

Internet and traditional IT companies are now investing more on chips' research and development due to the customized application of AI and IP protection. The four international giants Google, Apple, Facebook and Amazon, and their Chinese counterparts such as Baidu and Alibaba have all started work on their chips. For examples, Google announced its TPU3.0 with its development framework called TensorFlow to create a closed environment; Facebook announced PyTorch 1.0 software and hardware; Microsoft launched Project Brainwave, a low-lagging deep learning cloud platform based on FPGA; Amazon produced customized AI chips for future Echo equipment. As for Chinese companies, Baidu announced Kunlun chip targeting AI application; Alibaba

GAIIS 2024, May 10–12, 2024, Kuala Lumpur, Malaysia

ACM ISBN 979-8-4007-0956-2/24/05
https://doi.org/10.1145/3665348.3665420

Table 1: The advantages and disadvantages of GPU, FPGA, ASIC and brain-inspired chips [Owner-draw].

Type	Level of customization	Programmability	Computing power	Price	Advantages	Disadvantages	Application scene
GPU	Universal	unprogrammable	medium	high	Universal, suitable for large scale computing, mature design and production	Unable to fullu perform its parallel computing in reasoning end	Advanced complex algorithm and universal AI platform
FPGA	Semi-customized	Easy to program	high	medium	Flexible iteration through programming, better average performance, low power consumption, shorter development time (6 months).	High cost in mass production, low peak computing capability, high difficulty in hardware coding	Various industries
ASIC	Customized	Hard to program	high	high	Able to give out full performance through fixed algorithm, strong averageness, low energy consumption, small size, low cost in mass production	High spending in early stages, long development time, high technological risk	Providing customized and specific smart algorithm software for users in certain settings
Brain-inspired chip	Simulation of human brain	unprogrammable	high	-	Lowest energy consumption, highly effective communication, strong cognitive ability	Still at exploring stage	Various industries

established T-Head Semiconductor Co., Ltd. Which aims to provide customized AI chips for Alibaba's business; Huawei launched end-to-side chip Kirin 980 and cloud chip Ascend. At present, US companies, like NVIDIA and AMD, enjoy an obvious advantage in AI chips. AMD has launched the first GPU of 7mm targeting mobile devices. NVIDIA has launched brand new Volta structure GPU — the NVIDIA TeslaV100 GPU. GPU is mainly used to conduct parallel computing. It is faster than CPU and cheap than other AI specific CPU [1], [2].

And this paper focuses mainly on neural network acceleration based on NVIDIA's deep learning accelerators GPU.

2.2 Quantization Algorithm

Model compression refers to parameter compression or dimensionality reduction, or redesign of simple networks aiming at higher speed of network training and inference. In the industry, there are growing numbers of different ways to conduct model compression, like pruning [3], knowledge distillation [4], quantization [5], etc. As one of the universal ways to optimize neural network, model compression can reduce the size of deep neural network model and the time of model-based inference. It is applicable on most models and different hardware. If the computing capacity is limited, acceleration of INT8, FLOAT16 and even INT4 on chips will be of vital importance.

The definition of quantization is the process of converting data in Float32 to low bit like that in Float 32, INT8 or INT4. Why quantization? The aim is to lower the cost and to increase the bandwidth, that is, saving data with less bits to decrease the dependence on storage resources. Typical trained floating-point models, like Alexnet, are as big as 200MB, posing great pressure on saving space. Most neural networks have a fixed weight range for each layer with only light fluctuation, which is suitable for quantization compression. From FLOAT to INT8, the weight range can be reduced by 1/4. The power consumption can be reduced, as moving 8bit data is 4 times as effective as it is to move 32bit floating point model. The computing speed will be improved, as most CPU can support rapid process of 8bit data. Generally speaking, quantization can reduce both memory access and the amount of calculation, as well as energy consumption [6].

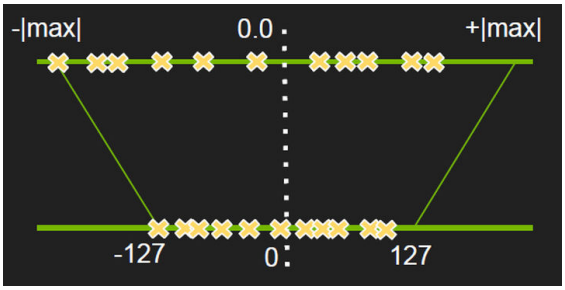

Figure 1: Principle of Neural Network Quantization [Owner-draw].

The principle of quantization is shown below in Figure 1. Find the range of activation or weight like the convolution layer, and then take the maximum as threshold, or use other statistic algorithm like KL to map this range into the range of plus-minus 128.

The formulas are below:

R stands for real floating-point value; Q stands for the fixed-point value after quantization; Z stands for the quantified fixed-point of 0 floating point; S stands for the minimum scale after fixed quantization. The quantization formula from floating point to fixed point is:

$$Q = \frac{R}{S} + Z \tag{1}$$

Formula for inverse quantization is:

$$R = (Q - Z) * S \tag{2}$$

The evaluation formula of S and Z is:

$$s = \frac{R_{max} - R_{min}}{Q\_max - Q_{min}} \tag{3}$$

$$Z = Q_{max} - \frac{R_{max}}{S} \tag{4}$$

Rmax, Rmin, Qmax, and Qmin respectively mean the maximum floating point, the minimum floating point, the maximum fixed-point and the minimum fixed-point. If the floating point R ascertained from Q after quantization still exceeds the maximum range, the truncation will be necessary.

Tens of thousands of datasets are necessary for training, but for quantization calibration, 500 datasets are enough, which make it very easy to deploy. Users only needs to quickly run through the original FP32 model on a calibration dataset, and then collect the data distribution on every layer, find a projection, and then an FP32 will be quantized into INT8. Precision and performance test can be carried out later.

On TensorRT, quantizable layers are convolution/Matrix multiplication/feedforward/up-sampling. The weight values and inputs of these layers will all be recorded as quantization. Some layers can but set as Float16 to lower their saving space and computing, even if no quantization is conducted on them.

2.3 Target Detection Algorithm

YOLO (You Only Look Once) [7] is an algorithm based on deep neural network which is used to identify and locate the target. Its biggest advantage lies in its fast running speed that allows its application in real-time systems. YOLO is widely adopted in the industrial circle. The backbone network of the YOLOv5 model comprises Focus, BottlenceckCSP, SSP networks, which mainly include Focus, Conv Convolution, BottelneckCSP and SSP modules [8].

This paper applied YOLOv5-s network from an open source community to detect vehicles Figure 2.

3 EXPERIMENT

For verification of the actual effect of acceleration by AI chips, this paper uses the BDD100K dataset [10] from the pre-trained YOLOv5-s model to train the model. It also adopts Nvidia Tesla T4 GPU to accelerate Tensorrt-INT8 and test its performance.

3.1 Introduction of the Training Environment

The BDD100K dataset used in this experiment is by far one of the largest and the most content-diverse public dataset of driving, consisted of more than 100,000 videos with footnotes such as image-level marks, object bounding box, drivable areas, driving lane marks and full-frame instance segmentation. The dataset is geographically, environmentally, and weather-wise diverse, and thus more robust in a new environment.

This experiment applied the vehicle bounding box for training and inference to convert BDD100K format to YOLO format for the following training [9].

The version of Python in the training is 3.9; Pytorch's version is 1.10.0. Some of the parameters are as followed: batch-size is 32, epochs is 200, 1r0 is 0.01, weight decay is 0.0005.

3.2 Introduction of Acceleration Environment of the Chip

This paper is mainly based on NVIDIA's GPU deep learning acceleration card, Nvidia Tesla T4 GPU, with power of 70 watts, memory of 16GB, mixed precision (FLOAT16/FLOAT32) of 65TFLOPS, INT8 Precision of 130TOPS. The card has provided acceleration services for many different cloud workloads and many industries Figure 3.

TensorRT is a deep learning framework that only propagates forward. The framework can analyze the trained Caffe, TensorFlow, ONNX network models, map them with each corresponding layer in TensorRT, convert all the models in other frameworks into TensorRT. Then, the optimization strategy on NVIDIA's own GPU can be initiated together with mixed-precision quantization and deployed acceleration. After having used TensorRT to accelerate, the host side and the device side executed tasks simultaneously and this model was not sensitive to the CPU type.

The version of TensorRT of this experiment is 8.0. Export ONNX from the trained Pytorch model and use TensorRT to analyze. Select 128 pictures randomly from BDD100K training data selection as selection for calibration and use trt.IInt8EntropyCalibrator2 to do quantitative calibration [11].

Trtexec provides TensorRT with a benchmark tool to test the function of the model.

3.3 Evaluation Standard

QPS (queries-per-second) in this paper refers to the number of pictures that can go through detection per second on Nvidia Tesla

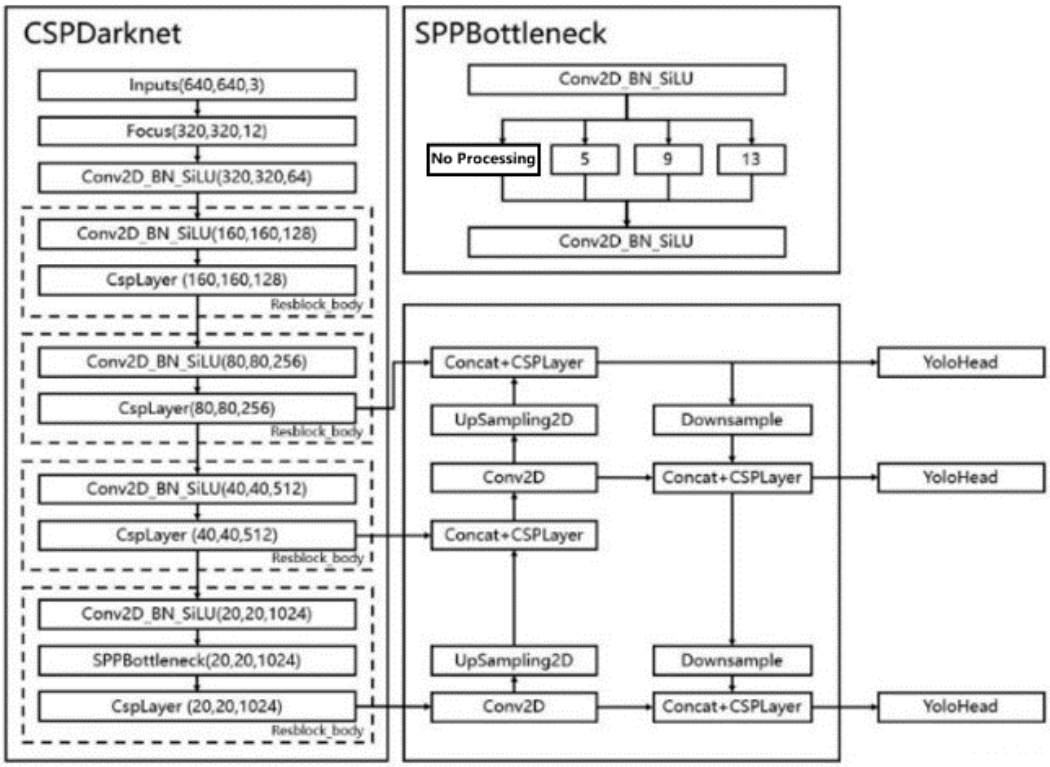

Figure 2: Network Structure of YOLOv5 [from CSDN].

Figure 3: Nvidia Tesla T4 GPU [from internet].

T4 GPU in YOLOv5-s accelerated by TensorRT.

$$FPS = FrameNum/WallTime$$

This experiment used the official tool trtexec of TensorRT to test the network's performance. Troughout is our parameter. The test result says the QPS batch-size is 32. For a GPU accelerator card, the bigger batch-size for one inference, the better QPS it has Figure 4.

The experiment result in the following uses MAP 0.5 (IoU=0.5) as the parameter of the precision of YOLOv5-s before and after quantization. Higher numerical value indicates higher precision [12].

3.4 Experiment Result

Table 2: The YOLOv5-s Network's Precision and Performance Model Before and After Quantization [Owner-draw, data comes from experiment].

	MAP 0.5	QPS	Model Size
YOLOv5-s-int8	45.5	789	11M
YOLOv5-s-float32	46.1	189	37M

3.4.1 ***Display of Performance Memory.*** From Table 2, it could be seen that the model's performance improved by 417% with deployable off-line model turned to 30% of its original size. MAP0.5 (IoU=0.5) dropped 0.6. For regions out of the bounding box, there seems to be no distinction, which is within an acceptable range.

3.4.2 Display of Precision (Figure 5, Figure 6).

4 COUNCLUSION

Chip is the core element that supports the development of AI, because neural network is a resource-intensive algorithm that does not only requires huge amount of calculation cost but also memory. Even though the computing resources are increasing by day, optimizing the training and inference for deep neural network is still vital to the realization of models. For that purpose, this paper applied the quantization and calibration function of TensorRT to fully release the calculating power of int8 on GPU which is 4 times faster than floating-point operation. Within an acceptable drop

```
[TRT] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +319, GPU +224, now: CPU 897, GPU 508 (MiB)
[TRT] [MemUsageChange] Init cuDNN: CPU +114, GPU +52, now: CPU 1011, GPU 558 (MiB)
[TRT] [MemUsageChange] TensorRT-managed allocation in engine deserialization: CPU +0, GPU +35, now: CPU 0, GPU 35 (MiB)
Engine loaded in 6.51875 sec.
[TRT] [MemUsageChange] Init cuBLAS/cuBLASLt: CPU +0, GPU +10, now: CPU 974, GPU 550 (MiB)
[TRT] [MemUsageChange] Init cuDNN: CPU +1, GPU +8, now: CPU 975, GPU 558 (MiB)
[TRT] [MemUsageChange] TensorRT-managed allocation in IExecutionContext creation: CPU +0, GPU +826, now: CPU 0, GPU 861 (MiB)
Using random values for input images
Created input binding for images with dimensions 24x3x640x640
Using random values for output output
Created output binding for output with dimensions 24x25200x85
Starting inference
Warmup completed 2 queries over 200 ms
Timing trace has 26 queries over 3.34375 s

=== Trace details ===
Trace averages of 10 runs:
Average on 10 runs - GPU latency: 123.389 ms - Host latency: 148.63 ms (end to end 246.411 ms, enqueue 1.21917 ms)
Average on 10 runs - GPU latency: 123.621 ms - Host latency: 148.862 ms (end to end 247.238 ms, enqueue 1.19711 ms)

=== Performance summary ===
Throughput: 7.77569 qps
Latency: min = 145.091 ms, max = 164.479 ms, mean = 149.272 ms, median = 147.031 ms, percentile(99%) = 164.479 ms
End-to-End Host Latency: min = 239.108 ms, max = 264.9 ms, mean = 247.878 ms, median = 244.444 ms, percentile(99%) = 264.9 ms
Enqueue Time: min = 1.14624 ms, max = 1.25189 ms, mean = 1.19661 ms, median = 1.20038 ms, percentile(99%) = 1.25189 ms
H2D Latency: min = 9.58838 ms, max = 9.59471 ms, mean = 9.59066 ms, median = 9.59039 ms, percentile(99%) = 9.59471 ms
GPU Compute Time: min = 119.841 ms, max = 139.234 ms, mean = 124.031 ms, median = 121.788 ms, percentile(99%) = 139.234 ms
D2H Latency: min = 15.6416 ms, max = 15.6694 ms, mean = 15.6506 ms, median = 15.6467 ms, percentile(99%) = 15.6694 ms
Total Host Walltime: 3.34375 s
Total GPU Compute Time: 3.22479 s
Explanations of the performance metrics are printed in the verbose logs.
```

Figure 4: Performance Test of YOLOv5-s by trtexec (screenshot of experiment).

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.45539
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.68118
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.46212
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.29199
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.43768
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.59558
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.39871
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.57360
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.62775
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.41549
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.62589
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.74360
```

Figure 5: Verifying MAP Results Based on the Dataset by YOLOv5-s int8 Quantization [screenshot of experiment].

Figure 6: Effect of Vehicle Detection Based on YOLOv5-s int8 Quantization [from internet].

in precision, an expected progress in function and reduction in memory can be achieved.

REFERENCES

[1] Prasanna S. Deep learning deployment with NVIDIA TensorRT[J]. NVIDIA Deep Learning Institute, New York, 2019.

[2] Vanholder H. Efficient inference with tensorrt[C]//GPU Technology Conference. 2016.

[3] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv: 1510.00149, 2015.

[4] Wen T, Lai S, Qian X. Preparing lessons: Improve knowledge distillation with better supervision[J]. Neurocomputing, 2021.

[5] Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: A whitepaper[J]. arXiv preprint arXiv:1806.08342, 2018.

[6] Nvidia, "8 bit inference with TensorRT. " 2017.http://on-demand.gputechconf.com/gtc/2017/presentation/s7310-8-bit-inferencewith-tensorrt.pdf.

[7] Redmon J, Farhadi A. Yolov3: An incremental improvement[J]. arXiv preprint arXiv:1804.02767, 2018.

[8] Zhang Z D, Tan M L, Lan Z C, *et al.* CDNet: a real-time and robust crosswalk detection network on Jetson nano based on YOLOv5[J]. Neural Computing and Applications, 2022: 1-12.

[9] Kasper-Eulaers M, Hahn N, Berger S, *et al.* Detecting heavy goods vehicles in rest areas in winter conditions using YOLOv5[J]. Algorithms, 2021, 14(4): 114.

[10] Yu F, Chen H, Wang X, *et al.* Bdd100k: A diverse driving dataset for heterogeneous multitask learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 2636-2645.

[11] Wang X, Yue X, Li H, *et al.* A high-efficiency dirty-egg detection system based on YOLOv4 and TensorRT[C]//2021 International Conference on Advanced Mechatronic Systems (ICAMechS). IEEE, 2021: 75-80.

[12] Stepanenko S, Yakimov P. Using high-performance deep learning platform to accelerate object detection[C]//Proceedings of the International Conference on Information Technology and Nanotechnology, Samara, Russia. 2019: 26-29.