Efficient Autonomous Driving System Design: From Software to Hardware

Yu Wang[1], Shulin Zeng[1], Kaiyuan Guo[2], Xuefei Ning[1], Yali Zhao[1,2], Zhongyuan Qiu[2],
Changcheng Tang[2], Shuang Liang[2], Huazhong Yang[1]
[1]Department of Electronic Enginnering, Tsinghua University [2]Novauto Co., Ltd.
Email: yu-wang@tsinghua.edu.cn

I. Introduction

Recent advancement in algorithm is pushing forward the application of autonomous driving in real life. The massive usage of deep learning in autonomous driving is making the system larger and more complex. On one hand, researchers are extending the usage of deep learning, from image process to lidar process, from sensing to control. On the other hand, people are designing larger models to achieve higher performance.

Such trend in algorithm design brings great challenges when it comes to deployment. As for autonomous driving, system latency is crucial to the safety issue. So a large platform with high computing capability is necessary. As the car itself is usually cost-sensitive and the heat dissipation is hard, the scale of the chips we can use on car is limited. To resolve the dilemma, we should take more efforts on the system's efficiency.

In this work, we try to increase the autonomous driving system's efficiency from both software and hardware aspects. For the software side, we focus on the deep learning model's design, optimize, and deployment stages, trying to reduce the system's workload and better fit the program with the hardware. For the hardware side, we propose customized accelerator for both neural network (NN) workload and non-NN workload to improve the overall system's latency. We will introduce our work on software and hardware respectively in section II and section III. Section IV concludes this paper.

II. Software

A. Model Design: Neural Architecture Search

The architecture design of a neural network is important for its performance and inference efficiency. As there exist diverse types of tasks and hardware platforms, it is costly to rely entirely on expert experiences to design the optimal architecture in every case. Zoph et al. [1] formulate the neural architecture search (NAS) problem and propose a reinforcement learning-based method to automatically design architectures.

We develope a NAS framework, aw_nas [2], and its open-source version is at https://github.com/walkerning/aw_nas. aw_nas implements popular NAS algorithms in a modularized manner. In addition, we develop our improved NAS algorithms, e.g., a predictor-based search strategy with higher sample efficiency [3], parameter sharing-based evaluation strategies with better ranking correlation [4], and so on.

In order to design an architecture that achieves a double-win of excellent task performance and high inference efficiency, hardware-aware NAS that considers the inference latency in the search process [5], [6] has received much attention. aw_nas also adopt hardware-aware NAS by introducing hardware profiling, hardware model building into the workflow. We also propose an example research work [7]. Further, for the co-exploration of hardware parameters and the architecture design, we have developed an efficient co-exploration method targeting the in-memory computing hardware platform in [8].

B. Model Optimization: Pruning & Quantization

Given a trained neural network, one usually needs to compress it before deploying it onto the hardware. The two most popular techniques for model compression are pruning and quantization. Structural pruning methods [9] aim at removing structured groups of weights from the model. To conduct structural pruning targeting given budgets (e.g., #FLOPs of the resulting model needs to be smaller than a threshold), we develop a budgeted pruning method [10] that decides a pruning plan with gradient-based optimization.

As for quantization, early in 2016 [11], we developed an 8-bit dynamic fixed-point format for efficient inference of NNs on FPGA. And recently, we develop a multi-level scaling format to improve the representational capability of low-bit tensors while ensuring the arithmetic of our low-bit tensors can be implemented efficiently on hardware [12]. This design can enable NN training (e.g., using 6-bit format is adequate to train a model on ImageNet), not limited to NN inference.

We develop toolkits based on PyTorch to compress and quantize a trained neural network model. The compression tool is able to parse PyTorch-format model and then iteratively compress a model to a target budget (FLOPs, parameters, or latency by fusing hardware models) automatically. Quantization tool takes ONNX-format model as input and exports a quantized ONNX model, which supports int8/uint8 high precision quantization and int8/fp32 fused quantization.

C. Deployment: NOVA-3D

The special characteristics of LiDAR point cloud data such as sparsity, large data quantity and unstructured data pose huge challenges that oftentimes lead to highly inefficient algorithms. For example, the farthest point sampling (FPS) operator [13] is used for resampling point cloud data. Although its sampling effect is excellent, the time complexity is O(N^2). A point cloud of a 64-channel LiDAR takes 100 to 150ms for a factor-two down-sampling even on a high-end NVIDIA RTX 2080 Ti GPU. Operators such as sparse convolutions (spconv) [14] consist of unique and gather/scatter operations and a large number of memory allocations (mask) which consumes much time and memory. Some of the more complex spconv-based 3D point cloud perception models need to use mixed precision to be able to perform normal model training and optimization even on large training clusters.

To address the above problem, we propose an optimized operator library on GPU with dedicated implementation using the CUDA cores and tensor cores, including point cloud voxelization, spconv, ball query and point sample. We compare the optimized operators with the open source implementation spconv on NVIDIA Xavier AGX platform with KITTI dataset. Experimental results show that we achieve 7.4x speedup on 8 different operators and reduce memory footprint by 50%.

We further developed a PyTorch toolkit that comprises a complete set of operators, covering pre-processing and post-processing of the point cloud model, backbone, model middleware and more. Combined with PyTorch's own official operators, we can optimize the entire model construction and training process of any 3D point cloud model. To handle the dynamic data size of the point cloud data, a novel constraint method is adopted which makes the static deployment of 3D point cloud perception models possible. On the deployment side, we expand TensorRT to a toolkit with customized operator plug-ins that is completely consistent with the PyTorch toolkit. So we can ensure identical results across training and deployment stage for all operators.

III. Hardware

A. *Efficient Spatial Multi-Task DNN Accelerators*

The design goal of traditional DNN accelerators is to pursue the ultimate performance and energy efficiency in the single-task scenario. These single-task optimized DNN accelerators, *e.g.*, Google TPU [15], NVDLA [16], and Eyeriss [17], usually put more and more computing units (*i.e.*, multiply-accumulate units (MACs)) and memory inside a single large core with the specialized dataflow (*e.g.*, weight stationary [16], output stationary [18], row stationary [17], *etc.*). However, as deep learning is widely used in autonomous driving, DNN accelerators in autonomous driving system are facing a shift from single-task to multi-task workloads [19]. For example, DNN accelerators need to simultaneously process different DNN models applied to scene perception and localization [20], path planning and behavior arbitration [21], and motion controller [22].

Recently, the community has paid much attention to the importance of introducing the multi-task feature into DNN accelerators. In this talk, we will start with a review of recent studies on multi-task DNN accelerators. Then, we will introduce the basic idea of our proposed multi-task DNN accelerator, that is, the shift from temporal sharing a single large core to spatial sharing multiple small cores [23]. We show that the system throughput of spatial multi-core virtualized design outperforms that of temporal single-core static design by 3-6×. Finally, we will discuss the design space exploration of spatial multi-task DNN accelerators from a coordinated architecture, mapping, and scheduling perspective. Our proposed co-exploration framework can find the optimal design under different workloads, which rivals Planaria [24] and Herald [25] by 3.0-7.5× and 1.8-3.6× Energy-Delay-Product (EDP) reduction.

B. *Efficient Non-NN accelerators*

When DNN accelerators cover the 99% of computation requirement in autonomous driving, the rest part, usually runs on CPU, may become the bottleneck. Prototype systems to evaluate algorithms usually contains not only high-end GPUs, but also server level CPUs. Tasks like feature extractions or point cloud registrations neither runs efficiently on GPUs or DNN accelerators and affects the system's latency. We propose a set of FPGA-based accelerators to solve the problem, including:

- An ORB feature extraction accelerator with dedicated image pyramid and descriptor pipeline.
- An image mosaicing accelerator supporting parking space detection, using end-to-end pixel mapping.
- Two Lidar point cloud pre-processors for BEV-based detection with Complex YOLO [26] and PointPillars [27] respectively.
- A point cloud registration accelerator used for car localization.

Experimental results show that the proposed accelerators achieves 5 to 50 times of speed up over embedded CPUs on FPGA, as shown in Table I.

TABLE I
Customized IP acceleration over embedded CPU.

task	latency(ms)		Acc.
	ARM	FPGA	
ORB Feature Extraction	696	33	21x
Image mosaicing	70	13.5	5.2x
Complex Yolo preprocess	90.9	9.6	9.7x
PointPillars preprocess	88.7	8	11x
Point Cloud Registration	2130	42.5	50x

IV. Summary

In this paper, we introduce our work on improving the efficiency of an autonomous driving system. For software, we use NAS to improve the model architecture's efficiency at the design stage and further compress the model with pruning and quantization. We also propose an optimized operator library to increase the hardware utilization ratio at runtime. For hardware, we propose customized accelerators for both NN and non-NN workloads to improve the system's efficiency. In the future, research should focus on system level's view, where there are more task-oriented optimization opportunities.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. U19B2019, 61832007, 62104128), National Key RD Program of China (No. 2017YFA02077600); China Postdoctoral Science Foundation (No. 2019M660641); Tsinghua EE Xilinx AI Research Fund; Beijing National Research Center for Information Science and Technology (BNRist); Beijing Innovation Center for Future Chips. The authors thank Toyota for the support.

References

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2017.

[2] X. Ning, C. Tang, W. Li, S. Yang, T. Zhao, N. Zhang, T. Lu, S. Liang, H. Yang, and Y. Wang, "aw_nas: A modularized and extensible nas framework," 2020.

[3] X. Ning, Y. Zheng, T. Zhao, Y. Wang, and H. Yang, "A generic graph-based neural architecture encoding scheme for predictor-based nas," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[4] X. Ning, C. Tang, W. Li, Z. Zhou, S. Liang, H. Yang, and Y. Wang, "Evaluating efficient performance estimators of neural architectures," in *Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[5] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2815–2823.

[6] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations (ICLR)*, 2020.

[7] S. Zeng, H. Sun, Y. Xing, X. Ning, Y. Shan, X. Chen, Y. Wang, and H. zhong Yang, "Black box search space profiling for accelerator-aware neural architecture search," *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 518–523, 2020.

[8] H. Sun, C. Wang, Z. Zhu, X. Ning, G. Dai, H. Yang, and Y. Wang, "Gibbon: Efficient co-exploration of nn model and processing-in-memory architecture," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.

[9] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[10] X. Ning, T. Zhao, W. Li, P. Lei, Y. Wang, and H. Yang, "Dsa: More efficient budgeted pruning via differentiable sparsity allocation," in *ECCV*, 2020.

[11] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," in *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2016, pp. 26–35.

[12] K. Zhong, X. Ning, G. Dai, Z. Zhu, T. Zhao, S. Zeng, Y. Wang, and H. Yang, "Exploring the potential of low-bit training of convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[14] Y. Yan and B. Li, "Spconv: Pytorch spatially sparse convolution library," website, 2021, https://github.com/traveller59/spconv/tree/v1.2.1.

[15] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[16] NVIDIA, "Nvdla deep learning accelerator," website, 2017, http://nvdla.org.

[17] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 367–379, 2016.

[18] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," in *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, 2015, pp. 92–104.

[19] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[20] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2584–2601, 2017.

[21] S. D. Pendleton, H. Andersen, X. Du, X. Shen, M. Meghjani, Y. H. Eng, D. Rus, and M. H. Ang, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, p. 6, 2017.

[22] M. A. Kamel, A. T. Hafez, and X. Yu, "A review on motion control of unmanned ground and aerial vehicles based on model predictive control techniques," *Journal of Engineering Science and Military Technologies*, vol. 2, no. 1, pp. 10–23, 2018.

[23] S. Zeng, G. Dai, H. Sun, K. Zhong, G. Ge, K. Guo, Y. Wang, and H. Yang, "Enabling efficient and flexible fpga virtualization for deep learning in the cloud," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2020, pp. 102–110.

[24] S. Ghodrati, B. H. Ahn, J. K. Kim, S. Kinzer, B. R. Yatham, N. Alla, H. Sharma, M. Alian, E. Ebrahimi, N. S. Kim *et al.*, "Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2020, pp. 681–697.

[25] H. Kwon, L. Lai, M. Pellauer, T. Krishna, Y.-H. Chen, and V. Chandra, "Heterogeneous dataflow accelerators for multi-dnn workloads," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 71–83.

[26] M. Simony, S. Milzy, K. Amendey, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[27] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.