Multiple Algorithms Against Multiple Hardware Architectures: Data-Driven Exploration on Deep Convolution Neural Network

Chongyang Xu, Zhongzhi Luan, Lan Gao, Rui Wang, Han Zhang, Lianyi Zhang, Yi Liu, Depei Qian

Multiple Algorithms Against Multiple Hardware Architectures: Data-Driven Exploration on Deep Convolution Neural Network

Chongyang Xu[1], Zhongzhi Luan[1], Lan Gao[1], Rui Wang[1⋆], Han Zhang[2], Lianyi Zhang[2], Yi Liu[1], and Depei Qian[1]

[1] Beihang University, Beijing, China
{xuchongyang1995,07680,lan.gao,wangrui,yi.liu,depeiq}@buaa.edu.cn
[2] Secience and Technology on Special System Simulation Laboratory, Beijing Simulation Center, Beijing, China
{xia_mei2000,lyzhang117}@163.com

Abstract. With the rapid development of deep learning (DL), various convolution neural network (CNN) models have been developed. Moreover, to execute different DL workloads efficiently, many accelerators have been proposed. To guide the design of both CNN models and hardware architectures for a high-performance inference system, we choose five types of CNN models and test them on six processors and measure three metrics. With our experiments, we get two observations and conduct two insights for the design of CNN algorithms and hardware architectures.

Keywords: Convolutional neural network · Hardware architecture · Performance evaluation.

1 Introduction

CNN models have large computation and consume much energy, putting significant pressure on CPUs and GPUs. To execute CNN models more efficiently, many specific accelerators are proposed (e.g., Cambricon-1A [11] and TPU [9]).

Due to the complexity of both sides, it is challenging to design high-performance processors for various CNN models and design CNN models with different types of processors. To tackle this, we perform a lot of evaluations, and we get two observations. Based on observations, we get two insights for the design of CNN algorithms and hardware architectures.

Following of this paper includes related work, experiments methodology, experiments result and analysis, conclusion and acknowledgements.

2 Related Work

Related evaluation work of CNN inference systems is as follows.

⋆ corresponding author

AI benchmark [7] measures only latency and one type of processors. Fathom [2] and SyNERGY [13] test two different types of processors and one metric. However,they do not compare the same type of processors with different versions. DjiNN and Tonic [3] measure the latency and throughput.They only use one CPU and one GPU without comparing different versions of same processor. BenchIP [16] and [9] use three metrics and three types of processors.BenchIP focuses on the design of hardware and use prototype chips instead of prodution level accelerators. [9] focus the performance of hardware architecture only, we give insights for the design of both CNN models and hardware architectures.

3 Experiment Methodology

In this section, we present the principles of workloads choosing, processors choosing, and software environment. We also give details of measurement.

Table 1. Selected Models

Model	Conv	FC	Weights(10^6)	Gflops[a]	input size	dataset
AlexNet [5]	5	3	60.1	0.62	227×227×3	Imagenet [14]
MobileNetv1 [6]	27	1	4.2	0.57	227×227×3	Imagenet
ResNet50 [4]	49	1	25.6	3.89	227×227×3	Imagenet
Vgg16 [15]	13	3	138.3	15.47	227×227×3	Imagenet
Yolov2 [12]	19	0	67.4	17.51	416×416×3	COCO [10]

Table 2. Selected Processors and Software Environment

Processor	GHz	TDP(W)	#TFLOPS/s	#core	GB /s	Numeric Library
CPU-E5	2.40	240	2.15	28	76.8	Intel MKL 2017 update 4
CPU-I5	3.40	65	0.20	4	34.1	Intel MKL 2017 update 4
GPU-P100	1.33	300	9.30	3584	732	Cuda8,CuDNN7
GPU-970	1.05	145	3.50	1664	224	Cuda8,CuDNN7
Cambricon	-	-	1.92	1	27.8	Libipu
TPU	-	-	180.00	8	600	-

Workloads are chosen from widely used tasks, with different layers, of different depths, of different size and of different topology as shown in table 1.

Hardware architectures are chosen from scenarios such as user-oriented situation, datacenter usage and mobile devices as shown in table 2, (1) Intel Xeon E5-2680 v4, (2) Intel Core I5-6500, (3) Nvidia TESLA P100, (4) Nvidia GeForce GTX 970, (5) Cambricon is a typical neural processor and the actual processor is HiSilicon Kirin 970 SoC in Huawei Mate 10. and (6) TPUv2, a publicly available DL accelerator from Google Cloud.

The same frame framework(tensorflow v1.6 [1]) and pre-trained models (.pb file) are used except Cambricon. Cambricon has its inference API and model format. Tensorflow 1.8 is provided for TPU by Google Cloud.

Three metrics are measured. Latency, the average milliseconds spent for an image. Throughput, the average images processed in a second. Energy efficiency, the amount of computation when a processor consumes 1 joule of energy.

To measure latency, we (1) load 100 images into memory and perform preprocessing, (2) run once to warm up, (3) infer one image each time, record time of 100 times inference and compute average latency. It is similar to throughput but using 1000 images and inferring one batch each time. Max throughput is achieved by tuning batch size. Measuring energy efficiency is similar to measuring the maximum throughput. Power is sampled via sysfs powercap interface at 1Hz, nvidia-smi at 10Hz on CPU and GPU respectively. We take energy consumption as energy consumed when the processor is under workload minus when the processor is idle. For Cambricon, we use MC DAQ USB-2408.

4 Experiment Results and Analysis

Figure 1 shows the result. As shown in figure 1, for most cases, the more is the computation, the higher is the latency or lower is the throughput. However, there are exceptions; we summarize them into two observations.

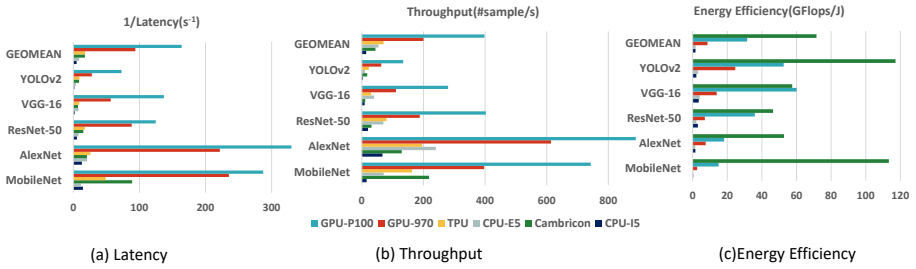
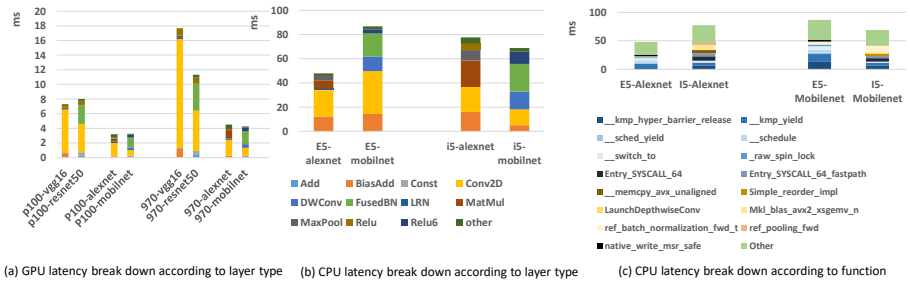


Fig. 1. Measured data. The longer the bar is, the better the performance is.

Observation 1: CNN models that have more computation may not incur higher latency or lower throughput. Models have more computation are expected to take more computing time, thus have higher latency and lower throughput. However, in figure 1(a), on CPU-E5, AlexNet with more computation has lower latency than MobileNetv1; on GPU-P100, Vgg16, AlexNet with more computation has lower latency than ResNet50, MobileNetv1 respectively.

Observation 2: Optimizations on CNN models are only applicable to specific processors. As shown in figure 1(a), MobileNetv1 has lower latency than AlexNet on CPU-I5 but higher latency on CPU-E5. MobileNetv1 is an optimized model but only performs well on a less powerful CPU.

To explain these two observations, we measure latency breakdown by layer types and functions, the result is shown in figure 2.



(a) GPU latency break down according to layer type (b) CPU latency break down according to layer type (c) CPU latency break down according to function

Fig. 2. Experiments for Observations

For observation 1, as shown in figure 2(a),2(b) BatchNorm layers have large execution time with low computation, which cause higher latency of MobileNetv1 than AlexNet on CPU-E5, higher latency of ResNet50 and MobileNetv1 than Vgg16 and AlexNet respectively on GPU-P100. Thus, we give insight 1.

Insight 1: BatchNorm layers have a low ratio of computation but a disproportionately high ratio of computing time on CPUs and GPUs. This suggests a trade-off between using more BatchNorm layers to achieve faster convergence for training [8] and using less BatchNorm to achieve faster inference.

For observation 2, as shown in figure 2(c), for MobileNetv1, the runtime overhead (kmp_yield(), sched_yield(), switch_to(), raw_spin_lock(), etc) on CPU-E5 occupies more than 40ms of 86.8ms in total, while the runtime overhead on CPU-I5 is about 20ms of 68.8ms in total. More cores of CPU-E5 increase the runtime overhead of DL frameworks.

Insight 2: The runtime overhead of modern DL frameworks increases with the increment of the core number on CPU. This suggests improving the computing capability of individual cores rather than increasing the number of cores to reduce latency.

5 Conclusion

In this work, we choose five CNN models and six processors and measure the latency, throughput, and energy efficiency. We present two observations and conclude two insights. These insights might be useful for both algorithms and hardware architectures designers.

- For algorithm designers, they need to balance the usage of BatchNorm layers for which can accelerate the training process but slow down inference.
- For hardware designers, BatchNorm layers deserve more attention; to reduce latency, it is more critical to improve the computing capability of individual cores than increasing the number of cores.

6 Acknowledgements

This work is supported by the National Key Research and Development Program of China under grant 2017YFB0203201. This work is also supported by the NSF of China under grant 61732002.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: OSDI 2016
2. Adolf, R., Rama, S., Reagen, B., Wei, G.Y., Brooks, D.: Fathom: Reference workloads for modern deep learning methods. In: IISWC 2016
3. Hauswald, J., Kang, Y., Laurenzano, M.A., Chen, Q., Li, C., Mudge, T., Dreslinski, R.G., Mars, J., Tang, L.: Djinn and tonic: Dnn as a service and its implications for future warehouse scale computers. In: ISCA 2015
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition
5. Hinton, G.E., Krizhevsky, A., Sutskever, I.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems
6. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
7. Ignatov, A., Timofte, R., Chou, W., Wang, K., Wu, M., Hartley, T., Van Gool, L.: Ai benchmark: Running deep neural networks on android smartphones. In: European Conference on Computer Vision (2018)
8. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML 2015
9. Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al.: In-datacenter performance analysis of a tensor processing unit. In: ISCA 2017
10. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014 (2014)
11. Liu, S., Du, Z., Tao, J., Han, D., Luo, T., Xie, Y., Chen, Y., Chen, T.: Cambricon: An instruction set architecture for neural networks. In: ACM SIGARCH Computer Architecture News (2016)
12. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition (2017)
13. Rodrigues, C.F., Riley, G.D., Luján, M.: Fine-grained energy profiling for deep convolutional neural networks on the jetson TX1. CoRR **abs/1803.11151** (2018)
14. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (2015)
15. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
16. Tao, J.H., Du, Z.D., Guo, Q., Lan, H.Y., Zhang, L., Zhou, S.Y., Xu, L.J., Liu, C., Liu, H.F., Tang, S., et al.: B ench ip: Benchmarking intelligence processors. Journal of Computer Science and Technology (2018)