EasyChair Preprint

№ 1831

Improve Image Classication by Convolutional Network on Cambricon

Peng He, Ge Chen, Kai Deng, Ping Yao and Li Fu

November 4, 2019

Improve Image Classification by Convolutional Network on Cambricon

Peng He[1,2*], Ge Chen[1,2*], Kai Deng[1,2], Ping Yao[1], and Li Fu[1]

[1] Institute of Computing Technology Chinese Academy of Sciences, HaiDian BeiJing 110108, China
[2] University of Chinese Academy of Sciences, Shijingshan BeiJing 14430, China
{hepeng18s,chenge18s,dengkai19s,yaoping,fuli}@ict.ac.cn

Abstract. Cambricon provides us with a complete intelligent application system, how to use this system for deep learning algorithms development is a challenging issue. In this paper, we exploit, evaluate and validate the performance of the ResNet101 image classification network on Cambricon with Cambricon Caffe framework, demonstrating the availability and ease of use of this system. Experiments with various operational modes and the processes of model inference show, the optimal running time of a common ResNet101 network that classifies the CIFAR-10 dataset on Cambricon is 1715ms. We hope that this work will provide a simple baseline for further exploration of the performance of convolutional neural network on Cambricon.

Keywords: Cambricon · Convolutional Neural Network · ResNet101.

1 Introduction

Image classification is a fundamental task in the field of computer vision that labels a picture to distinguish different categories visually and concisely. Classification task can also guide the development of other tasks: object detection [1], segmentation [8] and many more, such as instance segmentation, which performs per-pixel labeling of pictures at instance level. Therefore, since ResNet [10] was proposed as an effective image classification network at 2015, superior performance has enabled it to be applied as a backbone network to almost all other tasks [1, 2, 9]. Whether the ResNet network can run normally should be regarded as the basic standard to test whether an intelligent application system can run robustly.

As the calculation of neural network is a quite computationally intensive process, the computing capability of traditional CPU is far from meeting current computational complexity. Even the GPU is not designed originally specifically for artificial intelligence algorithms. The Cambricon chip is the first deep learning processor in the world as far as we know. It uses the hardware's digital logic structure, NFU (Neural Functional Units), to simulate the neural network

* Equal contribution.

connection structure to execute multiplication, addition, activation and other operations [17]. With ASIC (Application Specific Integrated Circuit) mode, which reduces a lot of unnecessary logic functions [24], Cambricon is extremely fast and consumes very low power, making it a superior alternative to GPU in video parsing, autonomous driving, and many more other real-world scenarios.

This work is implementing the ResNet101 classification network based on Cambricon with CIFAR-10 dataset [23]. By trying various operational modes, we find out the optimal operation strategy and running time to verify the performance and efficiency of Cambricon.

2 Related Work

Image Classification. Image classification, a fundamental problem in computer vision, can be described as categorizing images into one of several predefined classes [14]. It forms the basis for other computer vision tasks such as localization [5], detection [1–4, 6], and segmentation [7–9]. Traditionally, handcrafted features can be extracted from images using feature descriptors for the purpose of classification. The major disadvantage of this approach is that the accuracy of the classification result is profoundly dependent on the design of the feature extraction stage. In recent years, deep learning has developed to be a convenient, effective and robust tool to extract features from images, audio, etc, which does not require handcrafted features. Especially, DCNNs for image classification tasks achieved state-of-the-art results in the ImageNet Large Scale Visual Recognition Challenge since 2012 [14].

ResNet. In theory, the performance of the neural network should be positively related to the depth of the network, because the deeper the network, the more parameters it has, the more complicated it is. But the early experimenters observe that as the number of network layers increases, the model accuracy will rise first and then reach saturation, and continuing to increase layers will result in a decrease in accuracy sharply, which we called degradation [10]. ResNet's [10–13] proposal solves this problem very well. By introducing the residual network structure, it can make the network very deep, at the same time the final classification result is also very satisfactory. In this case, the depth of the network can be extended to tens, hundreds or even thousands of layers, providing the feasibility of extracting and classifying high-level semantic features. ResNet made a stunning appearance in the ILSVRC2015 competition, which raised the network depth to 152 layers, reducing the error rate to 3.57. In terms of image recognition error rate and network depth, it has greatly been improved compared with previous models. This also makes the network become the backbone network of the later convolutional neural network model, and many models with excellent performance subsequently are transformed on the basis of ResNet [1–4, 8, 9].

Cambricon. CPU and GPU are designed to handle many different computing tasks originally. They have multiple functional logic units inside, which are widely applicable. But for a computationally intensive computing task, they are not so efficient [15, 16]. Current artificial intelligence algorithm mainly in-

cludes two aspects: convolutional neural network and recurrent neural network. From the point of view of decomposition, they are composed of a lot of matrix multiplication or tensor element-by-element multiplication, so the CPU will no longer be suitable for this algorithm, and the GPU will be better, but there is still a lot of room for improvement. Since the introduction of the Cambricon chip, it has sparked a wave of research and application of deep learning accelerators. It is designed for the local and computational characteristics of artificial intelligence algorithms and neural network models to achieve better performance acceleration ratio and computing capability consumption ratio. On this basis, the application scenarios targeted by the deep learning chip are further divided, so the high-performance computing architecture DaDianNao [17] for the server side, the ShiDianNao [18] for the edge-end device application scenario, the PuDianNao [19] for the more generalized machine learning algorithms, all appeared. And the Cambricon instruction set [20] for a wider range of machine learning accelerators and the Cambricon-X [21] for hardware acceleration using data sparsity have been proposed for better use of these architectures.

3 Experiments

3.1 Multiple Operating Modes of Cambricon

To support the Cambricon machine learning processor, Cambricon modifies the open source deep learning programming framework Caffe, and adds some functions like offline, multi-core forward inference and so on, to form Cambricon Caffe. It is compatible with native Caffe's python/C++ programming interface and the native Caffe network model [22]. Besides, it provides a convenient interface to run various types of deep learning applications and a series of APIs provided by the Cambricon Neuware Machine Learning Library (CNML) for efficient inference. CNML interacts with the Cambricon machine learning processor by calling the Cambricon Neuware Runtime Library (CNRT) and drivers. Applications can also call CNML or CNRT directly to use the Cambricon Machine Learning Processor [20].

In the Cambricon operating environment, a variety of different programming models are supported. Firstly, the Cambricon machine learning processor is a multi-core processor architecture with 32 cores and supports two parallel modes: model parallelism and data parallelism. The setting of two parallelism parameters is based on the specific model. Reasonable model parallelism and data parallelism can optimize the performance of MLU (Machine Learning Unit). The MLU also supports multi-card mode, such as multiple MLU cards installed on a single server, allowing the model to run on different cards or distributing the calculations to multiple MLU cards. MLU supports two different modes of operating mode: online and offline. Online mode refers to the mode that depends on the Cambricon Caffe framework to run, and offline mode refers to the mode that uses the runtime function interface directly from the framework.

3.2 Design Details

In this work, our main concern is the running time of the model, that is, selecting the most suitable model settings, and completing the inference process of the ResNet101 classification model on the CIFAR-10 dataset in the shortest time. As we mentioned above, in order to speed up the inference process of the model, we try to select multi-core, multi-card, offline operation mode.

First of all, for programming model, we can choose multi-core and multi-card. The multi-core is determined by model parallelism and data parallelism. MLU provides us with up to 32 cores. The degree of parallelism of model and data decides the number of cores used. We use grid computing to choose the best combination of model parallelism and data parallelism for this problem. On the other hand, the BenchCouncil provides only one MLU in the runtime environment, we can't try the multi-card programming process. In terms of the model's operating mode, the online process depends on the operation of the Cambricon Caffe framework, while the offline mode is independent of the framework, so we choose the offline mode. The final result of the operation is shown in the figure 1.

3.3 Results in Challenge

In the subsequent experiments, we test model inference on pre-trained ResNet101 with the CIFAR-10 dataset. The three parameters that need to be adjusted are: parallelism of the model, parallelism of the data, and number of threads. The degree of parallelism of model and data together determines the number of cores used by the model. Because BenchCouncil provides us with a total of 32 cores, single card MLU, in order to get the best inference time, we use the most number of cores. In addition, we did some experiments to find out the influence of threads on the speed of model inference. In order to fully use the 32 cores, our model parallel number and data parallel number will be set as 2/16, 4/8, 8/4, 16/2 and 32/1 in order. Meanwhile, we select 2, 4, 6, ..., 14, 16 to test the optimal number of threads. The experimental results are shown in figure 1. According to it, we can summarize as fllow:

With the same degree of parallelism, the more threads, the longer the time of model inference: we believe that when the degree of parallelism of the model and the data both are 1, a total of 32 threads can be created, so the degree of parallelism and threads are mutually exclusive. Now, in order to maximize the number of cores, the degree of parallelism is also set to a maximum of 32, and then increase the number of threads, which will cause resource preemption among different threads, reducing the speed of reasoning.

Under the specific number of threads, when the model parallelism and data parallelism are 4 and 8, model inference time is the shortest, and it can be considered that the performance matching of the two factors is optimal.

4 Conclusion

In order to solve the problem of image classification on Cambricon, we find out a model setting that is most suitable for the execution of ResNet101 network,

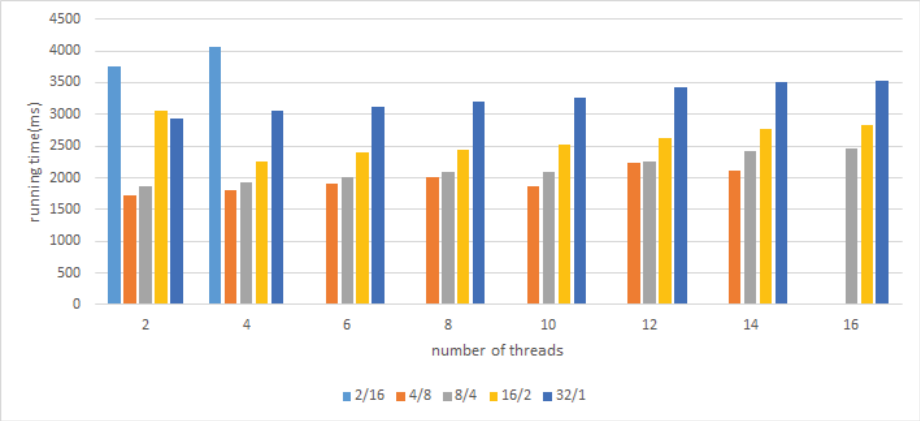

Fig. 1. Illustration of results of model inference under the influence of three factors. The abscissa represents the number of threads, and the ordinate represents the time of model inference. Different colors are different model and data parallelism matching. For example, 2/16 indicates that the model parallelism is 2 and the data parallelism is 16. Under certain thread numbers, the combination of different parallelism will cause MLU out of memory. For example, in 16 threads, if 2/16 is used, the memory will overflow, we don't display these results in this figure.

and adopt the multicore and multithreading method to transplant the network to the Cambricon operating environment for faster speed. Our transplantation is effective and efficient. We also record the differences among these combinations, and hope the implementation details publicly available can help the community adopt these useful strategies for object detection, scene parsing and semantic segmentation and advance related techniques.

5 Acknowledgements

The authors would like to thank BenchCouncil for BenchCouncil Testbed. The whole team is supported by Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No.XDA19020400), Equipment Pre-Research Fund (Grant No.61403120405, Grant No.6141B07090131) and Spaceborne Equipment Pre-Research Project(Grant No. 305030704).

References

1. R. Girshick, F., J. Donahue, S., T. Darrell, T.: Rich feature hierarchies for accurate object detection and semantic segmentation. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(2014)
2. R. Girshick, F.: Fast R-CNN. IEEE International Conference on Computer Vision (ICCV) (2015)

3. K. He, F., X. Zhang, S., S. Ren, T., et al: Spatial pyramid pooling in deep convolutional networks for visual recognition. European Conference on Computer Vision(ECCV), 346-361(2014)
4. Lin. Tsung Yi, F., Dollár. Piotr, S., Girshick. Ross, T., et al: Feature pyramid networks for object detection. IEEE Conference on Computer Vision and Pattern Recognition(CVPR)(2017)
5. Zhou. Bolei, F., Khosla. Aditya, S., Lapedriza. Agata, T., et al: Learning Deep Features for Discriminative Localization. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition(2016)
6. Lin. Tsung Yi, F., Goyal. Priya, S., Girshick. Ross, T., et al: Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision(2017)
7. Ronneberger. Olaf, F., Fischer. Philipp, S., Brox. Thomas, T., et al: U-net: Convolutional networks for biomedical image segmentation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)(2015)
8. K. He, F., G. Gkioxari, S., P. Doll'ar, T., et al: Mask R-CNN. IEEE International Conference on Computer Vision (ICCV) (2017)
9. H. Zhao, F., J. Shi, S., X. Qi, T., at al: Pyramid scene parsing network. IEEE Conference on Computer Vision and Pattern Recognition(CVPR) (2017)
10. He. Kaiming, F., Zhang. Xiangyu, S., Ren. Shaoqing, T., et al: ResNet. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition(2016)
11. He. Kaiming, F., Zhang. Xiangyu, S., Ren. Shaoqing, T., et al: Identity mappings in deep residual networks. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)(2016)
12. Wang. Fei, F., Jiang. Mengqing, S., Qian. Chen, T., et al: Residual attention network for image classification. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition(CVPR)(2017)
13. He. Kaiming, F., Zhang. Xiangyu, S., Ren. Shaoqing, T., et al: Deep Residual Learning for Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition(2016)
14. Krizhevsky. Alex, F., Sutskever. Ilya, S., Hinton. Geoffrey T.: ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems(2012)
15. Bergstra. James, F., Breuleux. Olivier, S., Bastien. Frederic Frédéric, T., et al: Theano: a CPU and GPU math compiler in Python. Proceedings of the Python for Scientific Computing Conference (SciPy)(2010)
16. A. Coates, F., B. Huval, S., T. Wang, T., et al: Deep learning with cots hpc systems. In International Conference on Machine Learning(2013)
17. Chen. Yunji, F., Luo. Tao, S., Liu. Shaoli, T., et al: DaDianNao: A Machine-Learning Supercomputer. Proceedings of the Annual International Symposium on Microarchitecture(MICRO)(2015)
18. Du. Zidong, F., Fasthuber. Robert, S., Chen. Tianshi, T., et al: ShiDianNao: Shifting vision processing closer to the sensor. Proceedings - International Symposium on Computer Architecture(2015)
19. Liu. Daofu, F., Chen. Tianshi, S., Shaoli. Liu, T., et al: PuDianNao: A Polyvalent Machine Learning Accelerator. ACM SIGPLAN Notices, 369-381
20. Liu. Shaoli, F., Du. Zidong, S., Tao. Jinhua, T., et al: Cambricon: An Instruction Set Architecture for Neural Networks. Proceedings - 2016 43rd International Symposium on Computer Architecture(ISCA)(2016)

21. Zhang. Shijin, F., Du. Zidong, S., Zhang. Lei, T., et al: Cambricon-X: An accelerator for sparse neural networks. Proceedings of the Annual International Symposium on Microarchitecture(MICRO)(2016)
22. Jia. Yangqing, F., Shelhamer. Evan, S., Donahue. Jeff, T., et al: Caffe: Convolutional architecture for fast feature embedding. MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia
23. Krizhevsky. Alex: Learning Multiple Layers of Features from Tiny Images. (2009)
24. C. Farabet, F., B. Martini, S., B. Corda, T., et al: NeuFlow: A runtime reconfigurable dataflow processor for vision. CVPR Workshop, 109–116(June 2011)
other custom neural network algorithms
25. J.-y. Kim, F., S. Member, S., M. Kim, T., et al: Real-Time Multi-Object Recognition Processor With Bio-Inspired Neural Perception Engine. IEEE Journal of Solid-State Circuits 45(1), 32–45 (Jan. 2010)