From Cloud to Edge: Enabling Fast Inference Computation for Convolutional Neural Networks on ARM-based Multi-core and Many-core Architectures

Paper **#XX**

2 pages

Abstract

Neural networks is widely used in deep learning applications. As neural networks going deeper, model training is still affordable with larger clusters or more GPUs. However, deploying these networks on front-end mobile devices can exhaust their limited computing resources. This paper has presented a highly optimized inference computation library for ARM-CPUs, named as CNNForward. CNNForward is trying to improve the efficiency of inference computation for convolutional neural networks on ARM-based multi-core and many-core architectures using both mathematical formula reconstruction/simplification and deep NEON instruction optimization. Experimental result shows that, compared with Apple BNNS, CNNForward is 3 times and 20 times faster on processing light models and deep models on Apple A8 fusion respectively. Processing VGG16 on a server with 64 ARM A72 cores, CNNForward is 16.9 times and 8.17 times faster than Caffe with OpenBlas enabled using 1 thread and 8 threads respectively.

1. Introduction

Neural networks are becoming ubiquitous and widely used in deep learning applications, such as face recognition [1], speech recognition [2], object classification [3] and self-driving cars [4], etc. Mobile devices are these embedding chips in cars, mobile phones, watches and wearable equipments etc, they are close to end users, and can provided accurate, intelligent and effective services with deep learning techniques.

Deploy deep learning service at cloud side, needs large amount of data transmission, powerful computing infrastructure, and also may compromise users' privacy, so deploy them on mobile device is a better choice. However the computing capacity, memory size, and batter endurance of mobile devices will be exhausted when models going deep, this situation will in return affect the quality of deep learning services. Thus a large amount of work is focused on resolving this issue.

Previous work on accelerating deep models can be classified into three directions, the first is to compress deep models, the second one is algorithm optimization, and the last one is to accelerate inference computation on special hardware.

Model compression is the primary way to reduce the storage and computation costs [5, 6, 7, 8, 9]. Deep compression [9] report impressive compression rates on AlexNet [3] and VGGNet [10]) by pruning weight and then retraining without hurting the overall accuracy. However, pruning parameters does not necessarily reduce the computation time since the majority of the parameters removed are from the fully connected layers where the computation cost is low, for example, the fully connected layers of VGG-16 occupy 95% of the total parameters but only contribute less than 1% of the overall floating point operations (FLOP). They also demonstrate that the convolutional layers can be also compressed and accelerated with sparse BLAS libraries or even specialized hardware. But 10% of sparsity can result to limited or no speedup with modern sparse libraries for CNN computation [11]. Then instead of compress existing models, researchers start to design new light models such as SqueezeNet [12], Darknet [13], MobileNets [14], they inherently has less parameters, but at the cost of slightly accuracy loss compared with deep models.

Algorithms optimization is another important strategy to accelerate conventional neural network. Conventional neural network can be transformed into matrix multiplication using GEMM [15], FFT[16], Winograd[17]. However GEMM will introduce large memory footprint, FFT is better for filters larger than 7x7 and Winograd is fast but limited on 3x3 filters, those effective algorithms currently are implemented and optimized only for GPUs with high memory

access bandwidth. Approximate algorithms without retraining includes SVD [18], bit-compression or fixed-point implementation [19] and network pruning [20, 21] can be used to pursue higher computing speed but may induce prediction loss.

As only limited number of target algorithms is used for inference computation of convolutional neural networks, hardware acceleration is used to pursue higher computing performance and better energy efficiency. These algorithms (includeing naive algorithms, GEMM algorithms) can be executed on multicores using SIMD [22], on GPUs [23, 15], on FPGAs [24] or ASICs [25, 26]. Mobile devices however are rarely equipped with these special hardwares, or even some have these special hardwares the cost of these chips will increase dramatically. For ARM-CPUs, NNPACK [27] has integrated GEMM, FFT, and Winograd algorithms, they also implement these algorithms with NEON instructions but in a straightforward way, no mathematical formula reconstruction/simplification and memory access optimization has been conducted. Thus its performance on Caffe2 is worse than eigen[28]. Other libraries such as OpenBlas [29], eigen[28], atlas [30] can be used to support GEMM algorithms, but there still space for performance improvement on ARM-based Multi-core and Many-core Architectures.

In order to improve the usability for ubiquitous easy-accessible ARM-CPUs, this paper focuses on accelerate inference computation of convolutional neural networks on commercial ARM-CPUs for the following 3 reasons:

(1) ARM-based CPUs are available on most mobile devices and IoT devices. These ARM-based CPUs follow the same instruction sets, that means efficient optimization methods can benefit almost all these devices.

(2) GPUs on mobile devices is rare and still very diverse. The advantage of GPU's computing capacity is not overpowering compared with ARM-based CPUs. Also, the computing efficiency of GPU architectures is also difficult to improve due to its underline disordered programming languages, which loses its flexibility on hardware level optimization and general usability.

(3) Mainstream deep learning frameworks such as Caffe [31] and Tensorflow [32] support general CPUs including ARM-CPUs currently. Their efficiency, however, need to be further improved. An efficient inference computing library is urged to narrow the gap between deep models with heavy computation workloads and limited computing capacity on mobile devices.

With the above concerns, this paper present a highly optimized inference computation library, named as CNNForward, for ubiquitous mobile devices with ARM-CPUs. CNNForward is trying to improve the computing efficiency of ARM-CPUs in mobile devices using both mathematical formula reconstruction/simplification and NEON instruction optimization. As conventional neural network cost almost 98% of computing time, improving the computation efficiency of convolutional neural networks layers is critical.

We optimized two algorithms to promote computing efficiency of the inference procedure in convolutional neural networks, the GEMM approach for general filter sizes, and the Winograd approach for specific filter size of 3x3.

- The GEMM algorithm transforms input data into columns by the order of convolutions, and stack different input channels together in order to perform efficient matrix-matrix multiplication against the filter matrix to yield final outputs. In order to achieve peak efficiency, we have developed our own GEMM subroutines for the ARM architectures with cache block techniques introduced in [33]. We also reorder the kernel parameters as they are needed in the GEMM subroutines in the preprocess stage, so that sequential vector load could be performed for even better efficiency.
- The Winograd convolutional algorithm [17] transforms the input and convolutional kernels to reduce complexity of the element-wise multiplication step, which conducts most of the computation, especially for models with manny channels. Previous optimizations on the Winograd algorithm stack data in the same minibatch together to form an efficient GEMM. For inference problem where minibatch size is only 1, however, the GEMM is reduced to GEMV which is much less efficient. We propose a GEMM variant kernel function, which follows the GEMM framework as well as the optimization techniques, but handles respective elements in the multiplication step in a SIMD fashion. Input transform and kernel transform also forms a sequential data layout for the GEMM vartiant kernel function to achieve best efficiency in the multiplication.

Experimental result shows that, On iPad min 4 with A8 fusion processor, CNNForward is 3 times faster than Apple BNNS [34] on light models, and 20 times faster on larger models. Processing VGG16 [10] on a server with 64 ARM A72 cores, CNNForward is 16.9 times faster than Caffe [31] (enabling OpenBlas [29]) using 1 thread and 8.17 times faster with 8 thread.

References

[1] Wenyi Zhao, R Chellappa, P J Phillips, and A Rosenfeld. Face recognition: A literature survey. *ACM Computing Surveys*, 35(4):399–458, 2003.

[2] Dong Yu and Li Deng. *Automatic Speech Recognition: A Deep Learning Approach*. Springer Publishing Company, Incorporated, 2014.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[4] Dean A Pomerleau. Alvinn: an autonomous land vehicle in a neural network. 89(77):305–313, 1989.

[5] Yann Le Cun, John S Denker, and A Sara. Optimal brain damage. Advances in neural information processing systems (NIPS), 1989.

[6] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. Advances in neural information processing systems (NIPS), 1993.

[7] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[8] Zelda Mariet and Suvrit Sra. Diversity networks. *arXiv preprint arXiv:1511.05077*, 2015.

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *International Conference on Learning Representations (ICLR)*, 2016.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *international conference on learning representations*, 2015.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (NIPS)*, pages 1–9, 2015.

[12] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[13] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[14] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[15] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014.

[16] Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. 2015.

[17] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4013–4021, 2016.

[18] Jian Xue, Jinyu Li, and Yifan Gong. Restructuring of deep neural network acoustic models with singular value decomposition. In *Interspeech*, pages 2365–2369, 2013.

[19] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[20] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[21] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. *arXiv preprint arXiv:1412.5474*, 2014.

[22] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. Improving the speed of neural networks on cpus. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, volume 1, page 4, 2011.

[23] Adam Coates, Brody Huval, Tao Wang, David Wu, Bryan Catanzaro, and Ng Andrew. Deep learning with cots hpc systems. In *International Conference on Machine Learning*, pages 1337–1345, 2013.

[24] Srimat Chakradhar, Murugan Sankaradas, Venkata Jakkula, and Srihari Cadambi. A dynamically configurable coprocessor for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 38, pages 247–257. ACM, 2010.

[25] Clément Farabet, Berin Martini, Benoit Corda, Polina Akselrod, Eugenio Culurciello, and Yann LeCun. Neuflow: A runtime reconfigurable dataflow processor for vision. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 109–116. IEEE, 2011.

[26] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*, 2017.

[27] NNNPACK. https://github.com/maratyszcza/nnpack.

[28] Eigen. http://eigen.tuxfamily.org.

[29] OpenBLAS. https://github.com/xianyi/openblas.

[30] R Clinton Whaley and Jack Dongarra. Automatically tuned linear algebra software. pages 38–38, 1998.

[31] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[32] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[33] Kazushige Goto and Robert A Geijn. Anatomy of high-performance matrix multiplication. *ACM Transactions on Mathematical Software (TOMS)*, 34(3):12, 2008.

[34] BNNS. https://developer.apple.com/documentation/accelerate/bnns.