Accelerating Convolutional Neural Networks Using Low Precision Arithmetic

Hiroki Naganuma
Department of Computer Science,
Tokyo Institute of Technology
hiroki11x@rio.gsic.titech.ac.jp

Rio Yokota
Global Scientific Information and Computing Center,
Tokyo Institute of Technology
rioyokota@gsic.titech.ac.jp

ABSTRACT

The recent trend in convolutional neural networks (CNN)[2] is to have deeper multilayered structures. While this improves the accuracy of the model, the amount of computation and the amount of data involved in learning and inference increases. In order to solve this problem, several techniques have been proposed to reduce the amount of data and the amount of computation by lowering the numerical precision of computation and data by utilizing the CNN's resistance to noise. However, there is a lack of discussion on the relationship between parameter compression and speedup within each layer of the CNN. In this research, we propose a method to speed up the inference by using half precision floating point SIMD instructions, by applying low precision to the learned model, in addition to reducing the data of the CNN model, and speeding up data access for layers that are computation-bound. We examined the influence of CNN recognition accuracy, the speedup for each layer, and its reason, when we apply our method.

KEYWORDS

image recognition, convolutional neural network, low precision arithmetic, half-precision, quantization

1 CNN WITH LOW PRECISION ARITHMETIC

1.1 float16 computation / data type

It is possible to reduce the amount of data of the CNN model by applying half-precision arithmetic / half-precision data type to the learned network model (in this case AlexNet[1]). Using half-precision data types enables faster data transfer rate by packing more variables into a given number of Bytes. In the present work, we take this one step further and accelerate not only the load/store of data, but also the arithmetic itself by utilizing half precision floating point SIMD instructions. This is especially effective for compute-bound layers such as the convolution layers. As a reference, we compare with the case where we use reduced precision for the data type only, and the case where we use it for both the computation and data type. When we use fp16 for only the data type we call this Dfp16, and when we used it for the math as well we call it Mfp16. Similarly, we call the fp32 counterparts Dfp32 and Mfp32. Simply using the data type fp16 reduces the memory footprint by half, and allows larger batch sizes or larger models without the need for Out of Core algorithms. In our case, we focus on reducing the time for inference by speeding up not only the data access, but also the computation by using half-precision floating point SIMD instruction for the compute-bound convolution layers.

HPCAsia2018, Tokyo, Japan
2017. N/A...$N/A
DOI: N/A

1.2 int8 quantization

In the method to be verified this time, the operation in the layer is performed by float 32, and the data type is set to int8 for the data propagating in each layer. In this research, we verify a method of mapping parameters to the range of -128, 127 based on the maximum of absolute value of parameters for each layer with respect to the quantization of 32 bit floating point as shown in Fig1. This method is called absmax. We also verified a method called minmax that quantizes the maximum and minimum value as a threshold.

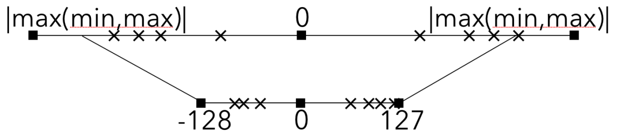

Figure 1: 8 bit quantization absmax method

2 EXPERIMENTS

In this research, we apply half-precision arithmetic to the learned model of AlexNet which is a simple CNN model, which won ILSVRC[3] in 2012. We investigated the degradation of recognition accuracy, the tendency of speed-up for each layer, and its cause. We also investigated several 8-bit quantization algorithms and their effect. The execution environment is as shown in the table1.

Table 1: Execution environment

	Environment 1	Environment 2
OS	CentOS 7.3	Ubunts 16.04
GPU	Tesla P100-SXM2	NVIDIA GTX1080TI
Num of CUDA Core	3584	3584
Memory Band Width	732 GB/s	484 GB/s
Memory	16 GB	11 GB

2.1 Speed comparison of matrix multiply-accumulate

In the CNN inference and learning calculation, the product-sum calculation of the matrix occupies most of its operation time. With respect to the product-sum operation of the matrix, experiments were conducted for the case of each data type of int8, float16, and

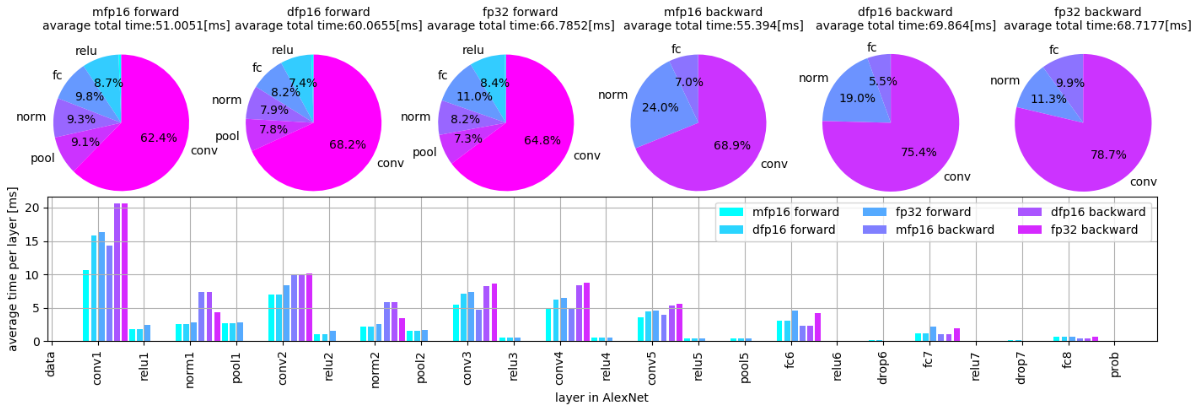

Figure 2: Improved speed by half-precision computation/data type application and influence on recognition accuracy of AlexNet

float32 using NVIDIA GTX1080TI and NVIDIA Pascal100 SXM2, respectively. In the current experiments, the matrix size is assumed to be m = n = k. cuBLAS[6] which is a dense matrix library also used for CNN operation was used in each product-sum operation. For int8, we used cuBLAS's GemmEx, for float16 we used cuBLAS's Hgemm, and for float32 we used cuBLAS's Sgemm. As shown in the table 2, it became clear that as the matrix size increases that if the result of float32 is taken as the baseline, float16 has twice its throughput and int8 has nearly four times its throughput.

Table 2: Comparison of elapsed time of multiply-add operation of matrix in each data type of int8, float16, float32

Matrix	ElapsedTime(P100 SXM2)		ElapsedTime(GTX 1080TI)	
Size	fp32 (sec)	fp16 (sec)	fp32 (sec)	int8 (sec)
256	5.53472e-05	6.92736e-05	3.00928e-05	2.37152e-05
1024	0.000379542	0.000318128	0.00024637	9.79936e-05
4096	0.0145966	0.00921268	0.0130537	0.00337301
16384	0.880288	0.462562	0.859033	0.217507

2.2 Influence of low-precision data type on CNN recognition performance

In this experiment, learned AlexNet is used as the model of CNN, by implementing it on chainer-v4. The operation in each layer is performed with float32, but the data type is set to the low-precision when propagating the data between the layers. Among the different methods for converting the data type to low precision, the method described in Section 1.2 showed the best recognition performance. As a result of the verification, it became clear that this method does not greatly deteriorate the recognition accuracy of CNN as shown in table3. In addition, verification was also carried out for VGG16 which is often used for transfer learning. As a result, the recognition performance of CNN degrades with 8-bit quantization using minmax, but not for absmax. The reason for achieving higher recognition performance when 8-bit quantization is used compared to when the data type is float32 is used, is most likely due to the effect of noise regularization[7] by application of low precision.

Table 3: Influence of low-precision data type CNN on recognition performance

	float32	float16	minmax_int8	absmax_int8
AlexNet top1-accuracy	51.025%	51.024%	50.116%	50.945%
VGG16 top1-accuracy	70.888%	70.885%	70.876%	70.902%

2.3 Impact and speedup of half-precision computation / data type application on CNN

We investigated the speedup of CNN's inference time when changing data type and computation type on NVIDIA Tesla P100 SXM2. We target AlexNet, which has a simple structure is simple and we used NVIDIA-Caffe which provides improved GPU implementation of the open-source deep learning framework Caffe. As shown in the Section1.1, when only the data type is set to float16 (Dfp16 Mfp32), the speed is not improved because the calculation is performed by float32. In the case where computation is done in float16(Dfp16 Mfp16) using SIMD instruction, speed of the convolution layer observed. This is because the convolution layer is compute-bound. The results of the experiment is shown in Fig. 2 and table4.

Table 4: Inference time and recognition accuracy by adapting half-precision computation / data type, batchsize = 512

Computation Precision	Inference Time	GPU UseRate	Memory Usage	top-1 acc	top-5 acc
fp32	66.785ms	99%	12845MiB	0.56828	0.79950
Dfp16Mfp32	60.065ms	99%	7475MiB	0.56813	0.79962
Dfp16Mfp16	51.005ms	99%	7475MiB	0.56821	0.79944

In the case of Dfp16 Mfp32, the memory usage has been reduced to 0.58%. The Fc7 layer, which is memory-bounded, achieves a 1.789x speedup. In the case of Dfp16 Mfp16, despite a decrease in Top1-accuracy of 0.007%, the data is reduced as much as the Dfp16 Mfp32 case, while speed up for the compute-bound convolution layer is achieved, especially in the conv1 layer, the speed up reaches 1.548x.

3 CONCLUSIONS AND FUTURE WORK

We propose a method to speed up the inference by using half precision floating point SIMD instructions. We found that no significant deterioration of AlexNet's recognition accuracy was observed, when half-precision computation is applied to the learned model and when the data type of int8 is applied. In addition, even when only data type is half-precision, it shows that the maximum speed of 1.789x can be attained in all the coupling layers and the like which are rate-limiting for memory access. By applying half-precision computation, it was shown that the maximum speed of 1.548x can be attained by the convolution layer, which is the computation-bound.

On the other hand, in order to make the technique of this research even faster, at this time we verified the implementation of int8 only for the data type and not the computation. Regarding the quantization, besides the methods verified this time, it is necessary to compare the method with fixed-point[5] etc., and to investigate not only the recognition accuracy but also the execution speed. It is necessary to consider a technique using 8-bit SIMD instruction for speeding up the computation as well. Additionally, we will try to speed up by adapting the 8-bit SIMD instruction to a method that effectively uses the weight of the compression model that uses 8-bit quantization such as Deep Compression[4] in the future.

Acknowledgments. The authors would like to thank A. Sekiya for technical assistance with the experiments. This work was supported by JST CREST Grant Number JPMJCR1687, Japan.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems 25*, pp. 1106–1114, 2012.
[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner., "Gradient-based learning applied to document recognition.", *Proc. of the IEEE*, pp. 2278fi?!-2324, 1998.
[3] O. Russakovsky,J. Deng,H. Su,et al. , "ImageNet Large Scale Visual Recognition Challenge", *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[4] S. Han,H. Mao,W. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding ", *International Conference on Learning Representation*, pp. 74–76, 2016.
[5] S. Gupta,A. Agrawal,K. Gopalakrishnan,P. Narayanan , "Deep Learning with Limited Numerical Precision.", *International Conference on Machine Learning*, 2015.
[6] NVIDIA, CUBLAS. *https://developer.nvidia.com/cublas.*
[7] Y. Luo, F. Yang, "Deep Learning With Noise", *http://www.andrew.cmu.edu/user/fanyang1/deep-learning-with-noise.pdf*, 2014.