POSTER: A Novel Memory-Efficient Deep Learning Training Framework via Error-Bounded Lossy Compression

Sian Jin
Washington State University
Pullman, WA, USA
sian.jin@wsu.edu

Guanpeng Li
University of Iowa
Iowa City, IA, USA
guanpeng-li@uiowa.edu

Shuaiwen Leon Song
University of Sydney
Sydney, NSW, Australia
shuaiwen.song@sydney.edu.au

Dingwen Tao*
Washington State University
Pullman, WA, USA
dingwen.tao@wsu.edu

Abstract

DNNs are becoming increasingly deeper, wider, and non-linear due to the growing demands on prediction accuracy and analysis quality. When training a DNN model, the intermediate activation data must be saved in the memory during forward propagation and then restored for backward propagation. Traditional memory saving techniques such as data recomputation and migration either suffers from a high performance overhead or is constrained by specific interconnect technology and limited bandwidth. In this paper, we propose a novel memory-driven high performance CNN training framework that leverages error-bounded lossy compression to significantly reduce the memory requirement for training in order to allow training larger neural networks. Specifically, we provide theoretical analysis and then propose an improved lossy compressor and an adaptive scheme to dynamically configure the lossy compression error-bound and adjust the training batch size to further utilize the saved memory space for additional speedup. We evaluate our design against state-of-the-art solutions with four widely-adopted CNNs and the ImangeNet dataset. Results demonstrate that our proposed framework can significantly reduce the training memory consumption by up to 13.5× and 1.8× over the baseline training and state-of-the-art framework with compression, respectively, with little or no accuracy loss. The full paper can be referred to at https://arxiv.org/abs/2011.09017.

CCS Concepts: • **Computing methodologies → Parallel algorithms**; • **Neural networks**;

Keywords: Neural Network, GPU Memory, Compression

*Corresponding author: Dingwen Tao (dingwen.tao@wsu.edu).

PPoPP '21, February 27-March 3, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8294-6/21/02.
https://doi.org/10.1145/3437801.3441597

1 Introduction

Training deep and wide neural networks has become increasingly challenging. While many state-of-the-art deep learning frameworks such as TensorFlow [1] can provide high computation throughput by leveraging the massive parallelism on general-purpose accelerators such as GPUs, one of the most common bottlenecks remains to be the high memory consumption during the training process, especially considering the limited on-chip memory available on modern DNN accelerators.

In recent years, several works have been proposed to reduce the memory consumption for DNN training, including activation data recomputation [2, 6], migration [11, 12], and compression [3, 5]. Recomputation takes advantage of the layers with low computational cost, and deallocate the activation data for those layers and recompute them based on their prior layer during the back propagation when needed. This method can reduce some unnecessary memory cost, but it can only be applied to limited types of layers with low performance overhead.

Another type of methods are proposed around data migration [11, 12], which sends the activation data from the accelerator to the CPU host when generated, and then loads it back from the host when needed. However, the performance of data migration heavily depends on the interconnect bandwidth available between the host and the accelerator(s), and the in-node interconnect technology applied.

Last but not least, data compression is another efficient approach to reduce the memory consumption, especially for conserving the memory bandwidth [5, 5]. The basic idea using data compression here is to compress the activation data when generated, hold the compressed data in the memory, and decompress it when needed. However, using lossless compression [3] can only provide a relatively low memory reduction ratio (i.e., compression ratio), e.g., typically within 2×. Some other studies such as JPEG-ACT [5] leverages the similarity between activation tensors and images for vision recognition tasks and apply a modified JPEG compressor to activation data. But it suffers from uncontrollable compression error and require a dedicated hardware component.

We note that all the three methods above are orthogonal to each other. Thus, in this paper, we mainly focus on

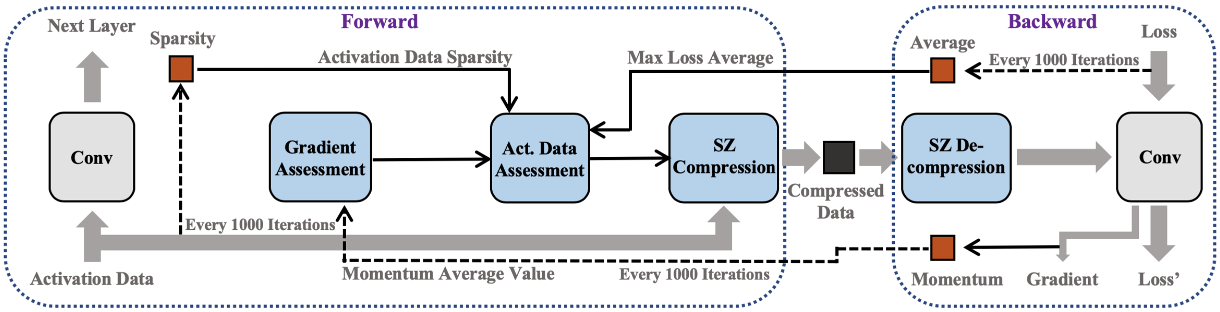

Figure 1. Overview of our proposed memory-efficient CNN training framework.

designing an efficient lossy compression based solution for convolutional layers, to achieve the memory reduction ratio beyond the state-of-the-art approach on CNN models.

2 Design of Proposed Framework

The overview of our proposed memory-driven framework is shown in Figure 1. We iteratively repeat the process shown in Figure 1 for each convolutional layer in every iteration.

Parameter Collection. We collect the parameters of current training status for the following adjustment of lossy compression configurations. Our framework mainly collects two types of parameters: (1) offline parameters in CNN architecture, and (2) semi-online parameters including activation data samples, gradient, and momentum. Note we only extract semi-online parameters every W iterations to reduce the computation overhead and improve the overall training performance.

Gradient Assessment. We estimate the limit of the gradient error that would result in little or no accuracy loss to the training curve, Based on our theoretical analysis, we need to determine the acceptable σ in the gradient error distribution that minimizes the impact to the overall training accuracy curve.

Activation Assessment. We dynamically configure the lossy compression for activation data based on the gradient assessment in the previous phase and the collected parameters. We simplify our estimator to the following:

$$eb = \frac{\sigma}{a\bar{L}\sqrt{NR}} \tag{1}$$

where eb is the absolute error bound for activation data, σ describes the acceptable error distribution in the gradient, a is the empirical coefficient, $\bar{L}$ is the average value of current layer's loss, N is the batch size, and R is the sparsity ratio of activation data.

Adaptive Compression. In the last phase, we deploy the lossy compression with our optimized configuration to the corresponding convolutional layers. We use the GPU version of SZ lossy compression [4, 10, 14] (i.e., cuSZ [16]) in this paper because of its high compression ratio and high throughput with absolute error bound [15]. In addition, we propose to modify cuSZ for the case of compressing continuous zeros to avoid generating a series of small values in decompressed data by SZ, which would reduce the efficiency of our framework.

3 Experimental Evaluation

Our evaluation are conducted with Caffe [8] and Tensorflow [1]. Our experiment platform is the TACC Longhorn system, of which each GPU node is equipped with 4 Nvidia Tesla V100 GPUs per node. Our evaluation dataset is the ImageNet-2012 [9]. We use the CNN models for image classification including AlexNet [9], VGG-16 [13], and ResNet-18/50 [7].

Figure 2 illustrates the result with AlexNet. We can observe that our framework does not obviously affect the training accuracy. In the early stage of the training, compression ratio can be slightly unstable because of the relatively large change to the model. Note that the compression ratio will change slightly when the learning rate changes, because the learning rate only matters when updating the weights.

Table 1 shows the compression ratio of convolutional layers that our framework can provide. There is almost no accuracy loss or only little, with up to 0.31% . This thanks to our careful control of compression error and thorough theoretical analysis and modeling of error impact. Our framework can deliver a promising compression ratio without heavy efforts of fine-tuning any parameter for different models. Overall, our proposed framework can provide up to 13.5× compression ratio with little or no accuracy loss.

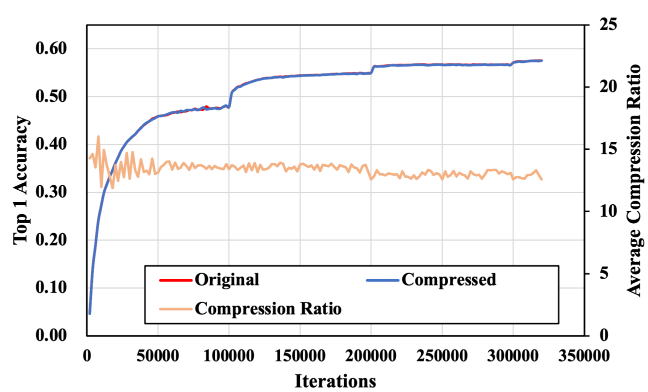

Figure 2. Training accuracy curve comparison between the baseline and our proposed framework (batch size = 256).

Table 1. Comparison of accuracy and activation size between baseline training and our proposed framework.

Neural Network		Top-1 Accuracy	Convolutional Act. Size	Compress Ratio
AlexNet	baseline	57.41%	407 MB	
	compressed	57.42%	**30 MB**	13.5×
VGG-16	baseline	68.05%	9.30 GB	
	compressed	68.02%	**0.83 GB**	11.1 ×
ResNet-18	baseline	67.57%	3.42 GB	
	compressed	67.43%	**0.32 GB**	10.7 ×
ResNet-50	baseline	71.49%	10.28 GB	
	compressed	71.18%	**0.93 GB**	11.0 ×

Compared with the lossless compression based solution [3], which reduces the memory usage by within 2×, our framework outperforms it by over 9×; compared with the the current state-of-the-art lossy compression based solution [5], which uses an image based lossy compressor to provide up to 7× compression ratios, our framework outperforms it by 1.5× and 1.8× on ResNet-18 and ResNet-50, respectively.

Our framework introduces relatively small overhead to the training process while can greatly reduce the memory utilization and allow larger and wider neural networks to be trained with limited GPU memory. Moreover, the saved memory can also be further utilized for larger batch size, which improves the overall performance. Another potential way to improve the performance from increasing the batch size is faster convergence speed to well trained status [17]. More batch size can lead to a more precise direction for the gradient instead of just rely on methods such as the momentum to reduce the impact of gradient uncertainty.

Overall, our proposed framework introduces about 17% overhead the training process while keeping the same training batch size. Moreover, our framework can further utilize the saved memory to increase the batch size and improve the training performance to offset this performance overhead. For example, our framework can achieve as low as 7% overhead on VGG-16 by increasing the batch size from 32 to 256 with the similar memory consumption. In comparison, the state-of-the-art migration solution such as Layrub on average achieves a memory reduction of 2.4× but with a high training performance overhead of 24.1% [11].

Acknowledgments

This research is supported by the National Science Foundation under Grants OAC-2034169 and OAC-2042084. The authors acknowledge the Texas Advanced Computing Center (TACC) for providing HPC resources that have contributed to the research results reported within this paper.

References

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

[2] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174* (2016).

[3] Esha Choukse, Michael B Sullivan, Mike O'Connor, Mattan Erez, Jeff Pool, David Nellans, and Stephen W Keckler. 2020. Buddy compression: Enabling larger memory for deep learning and HPC workloads on gpus. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture*. 926–939.

[4] Sheng Di and Franck Cappello. 2016. Fast error-bounded lossy HPC data compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium*. 730–739.

[5] R David Evans, Lufei Liu, and Tor M Aamodt. 2020. JPEG-ACT: Accelerating Deep Learning via Transform-based Lossy Compression. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture*. 860–873.

[6] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. 2017. The reversible residual network: Backpropagation without storing activations. In *Advances in neural information processing systems*. 2214–2224.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 770–778.

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

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

[10] Xin Liang, Sheng Di, Dingwen Tao, Sihuan Li, Shaomeng Li, Hanqi Guo, Zizhong Chen, and Franck Cappello. 2018. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *2018 IEEE International Conference on Big Data*. 438–447.

[11] Bo Liu, Wenbin Jiang, Hai Jin, Xuanhua Shi, and Yang Ma. 2018. Layrub: layer-centric GPU memory reuse and data migration in extreme-scale deep learning systems. In *Proceedings of the 23rd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 405–406.

[12] Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen W Keckler. 2016. vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 18.

[13] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[14] Dingwen Tao, Sheng Di, Zizhong Chen, and Franck Cappello. 2017. Significantly improving lossy compression for scientific data sets based on multidimensional prediction and error-controlled quantization. In *2017 IEEE International Parallel and Distributed Processing Symposium*. 1129–1139.

[15] Dingwen Tao, Sheng Di, Xin Liang, Zizhong Chen, and Franck Cappello. 2019. Optimizing lossy compression rate-distortion from automatic online selection between sz and zfp. *IEEE Transactions on Parallel and Distributed Systems* 30, 8 (2019), 1857–1871.

[16] Jiannan Tian, Sheng Di, Kai Zhao, Cody Rivera, Megan Hickman Fulp, Robert Underwood, Sian Jin, Xin Liang, Jon Calhoun, Dingwen Tao, and Franck Cappello. 2020. cuSZ: An Efficient GPU-Based Error-Bounded Lossy Compression Framework for Scientific Data. In *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*. 3–15.

[17] Yang You, Igor Gitman, and Boris Ginsburg. 2017. Scaling sgd batch size to 32k for imagenet training. *arXiv preprint arXiv:1708.03888* 6 (2017).