Streaming Batch Gradient Tracking for Neural Network Training (Student Abstract)

Siyuan Huang,[1] **Brian D. Hoskins,**[2] **Matthew W. Daniels,**[2] **Mark D. Stiles,**[2] **Gina C. Adam**[1]*
[1]School of Engineering & Applied Science, George Washington University, Washington, DC, USA
[2]Physical Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, USA

Abstract

Faster and more energy efficient hardware accelerators are critical for machine learning on very large datasets. The energy cost of performing vector-matrix multiplication and repeatedly moving neural network models in and out of memory motivates a search for alternative hardware and algorithms. We propose to use streaming batch principal component analysis (SBPCA) to compress batch data during training by using a rank-k approximation of the total batch update. This approach yields comparable training performance to minibatch gradient descent (MBGD) at the same batch size while reducing overall memory and compute requirements.

Introduction

Recent assessment of the energy necessary to train deep networks highlights the high financial and environmental costs associated with this fast growing field (Strubell, Ganesh, and McCallum 2019). New hardware accelerators are needed for efficient local and cloud computing (Ambrogio et al. 2018).

To accelerate training, research has increasingly focused on variable learning rate methods that add additional hyperparameters to the training of neural networks. Though these additional hyperarameters decrease the training time, they double or triple the memory overhead compared to a unit batch size stochastic gradient descent (SGD). These approaches are nevertheless preferred over SGD, however, since they can accelerate training on the rapidly fluctuating gradient vector to provide a superior learning trajectory.

In our approach, we accelerate training and reduce overhead by generating a stochastic low-rank approximation of the gradient using streaming principal component analysis (SPCA). SPCA was proposed by Oja in 1982(Oja 1982), and recent proposals combine SPCA with adaptive algorithms to optimize the learning rate while using only a single pass over the data (Henriksen and Ward 2019). (Burrello et al. 2019) proposed a parallel implementation of streaming History-PCA to save memory when backpropagating. SPCA remains an active research area which is promising for edge computing and other applications where memory usage is critical. To our knowledge, Hoskins et al. are first to use SPCA to compress batch gradients (Hoskins et al. 2019). Here we propose the SBPCA training algorithm, which approximates minibatch gradient descent with a slowly varying thin singular value decomposition of the overall gradient. Initial results show that SBPCA can accelerate training while reducing memory overhead and energy costs.

*Correspondence: GinaAdam@gwu.edu

Method Details

In any SGD-derived method, the weights in a given layer are updated as $\Theta \leftarrow \Theta - \alpha\nabla\Theta$ where Θ is the weight matrix trained for that layer, α is the learning rate, and $\nabla\Theta$ is a stochastic approximation of $\partial\ell/\partial\Theta$ over a batch B, with ℓ the loss function evaluated after the feedforward step.

We consider an alternative calculation of $\nabla\Theta$ through a streaming low rank approximation based on the Singular Value Decomposition (SVD). Using Algorithm 1, the stochastic approximation is given by $\nabla\Theta = X\Sigma\Delta^T$, where X is the left singular matrix, σ is the vector of singular values, $\Sigma = \text{diag}(\sigma)$, and Δ is the right singular matrix. Restricting σ to its top k singular values, the memory cost of Algorithm 1 is $m \times k + k + n \times k$. We generally take $k \ll \min\{m, n\}$ to approximate $\nabla\Theta$ with efficient memory usage. Algorithm 1 updates this approximation using a novel block-averaged bi-iterative implementation of Oja's rule, and uses QR decomposition to re-orthogonalize the singular vectors.

Algorithm 1 Streaming Batch PCA (SBPCA) Update

Require: σ, X, Δ and b
for $i = 1, 2, ..., B/b$ **do**

X **step:**	Δ **step:**
$y = \delta_i\Delta/b$	$z = x_iX/b$
$X \leftarrow \frac{iX}{i+1} + \frac{x_i^T y}{(i+1)\sigma}$	$\Delta \leftarrow \frac{i\Delta}{i+1} + \frac{\delta_i^T z}{(i+1)\sigma}$
$X \leftarrow \text{QR}(X)$	$\Delta \leftarrow \text{QR}(\Delta)$

X-Δ **step:** $\sigma \leftarrow \frac{i\sigma}{i+1} + \sum_{\text{rows}} \frac{(\delta_i\Delta)\odot(x_iX)}{(i+1)b}$
end for
Calculate $\nabla\Theta = X \cdot \text{diag}(\sigma) \cdot \Delta^T$

The X matrix (and Δ respectively) is updated as $X \leftarrow cX + (1-c)d$ where c is a convergence coefficient and d represents the block-averaged update of X over a minibatch of the input data, represented as x (and δ respectively), a matrix

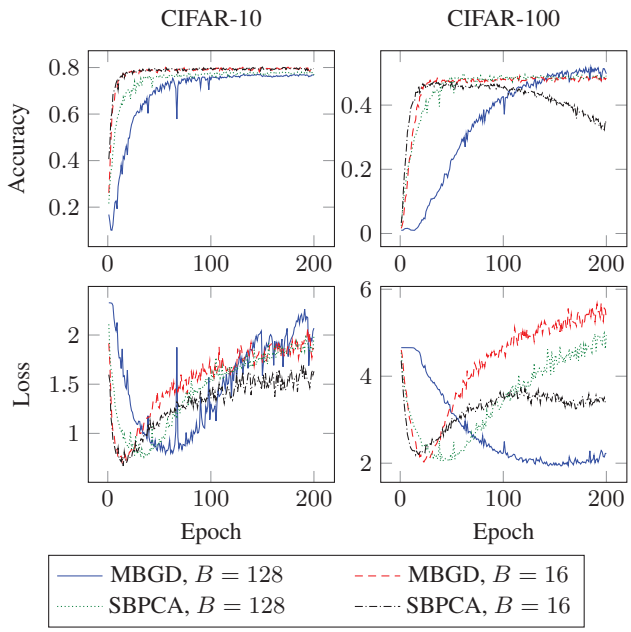

Figure 1: Comparative accuracy and loss for SBPCA and MBGD for CIFAR-10 (rank $k = 3$) and CIFAR-100 ($k = 30$) for $B \in \{16, 128\}$, with $b = B/4$. SBPCA recreates the loss minimum artifact in AlexNet present in MBGD.

with with rows b representing a subset of a minibatch of data with block size b examples. We describe two approaches. In the fixed version of Algorithm 1, we vary the convergence coefficient from block to block as $c_i = i/(i+1)$ while keeping the block size b of the minibatch fixed. However, we find that fixed values of b can lead to poor sampling of the space, so we introduce an alternative version (SBPCA with variable b or SBPCAV), adapted from Algorithm 1, with fixed $c = 1/2$ (implementable in hardware as a single bit-shift) but variable block size $b_i = 2^{(i-1)}$, with i the block index. The rank-1 updates corresponding to each singular vector of X and Δ are rescaled by their respective values in σ.

Experiments

We evaluate the SBPCA and SBPCAV algorithms on the CIFAR-10, CIFAR-100, and ImageNet datasets using an AlexNet modified to accommodate the smaller input size while maintaining the five convolutional and three fully connected layer structure. These algorithms are used only for fully connected layers; convolutional layers have less memory overhead, and so benefit less. We compare with MBGD results.

Since the initialization condition of the gradient approximation for the next batch is the end condition of the prior batch, the gradient estimation includes significant prior gradient history. This acts as a form of momentum, accelerating the training convergence compared to MBGD. In our experiments, we observed this is especially powerful at low learning rates. At high learning rates, the gradient changes more rapidly than the gradient estimation can be updated, destabilizing the training process. As seen in Fig. 1, the SPCA consistently achieves faster convergence than the classic MBGD for equivalent learning rates. Since AlexNet's structure poorly solves CIFAR, it leads to a non-monotonic behavior in the loss due to overtraining and can converge to an undesired local minimum. For small datasets, SBPCA approaches yield-equivalent accuracy to MBGD, shown in Table 1. Even for ImageNet, the algorithm can reach 85% of MBGD accuracy. We also see a dramatic difference in the performance of SBPCAV compared to SBPCA.

Table 1: Accuracy of training methods and ranks for CIFAR-10, CIFAR-100, and ImageNet. CIFAR hyperparameters same as before. For ImageNet, $B = 256$ and $b = 64$. *Fixed* refers to SBPCA and *Varied* to SBPCAV.

	CIFAR-10		**CIFAR-100**		**ImageNet**	
Rank	*Fixed*	*Varied*	*Fixed*	*Varied*	*Fixed*	*Varied*
1	0.7644	0.7391	0.4103	0.4163	0.3382	0.3660
3	0.7817	0.7729	0.4288	0.3783	0.3747	0.4065
10	0.7913	0.7840	0.4652	0.4252	0.4329	0.4454
30			0.4988	0.4563	0.4698	0.4602
100			0.5159	0.4712		
MBGD:	**0.7712**		**0.5185**		**0.5434**	

Conclusions

Our proposed SBPCA and SBPCAV can produce stochastic approximations of the gradient updates sufficient to train functionally relevant neural networks. We find that approximations of rank less than about ten are typically good enough to capture most relevant information about the local loss function. We verify these methods' effectiveness on three image datasets. While results are better on CIFAR-10 and CIFAR-100, even on ImageNet, SBPCAV can reach 85% of MBGD accuracy at significantly lower memory overhead. These results suggest future research exploring detailed memory analysis, impact of dropout and low rank approximations of more sophisticated training algorithms.

References

Ambrogio, S.; Narayanan, P.; Tsai, H.; et al. 2018. Equivalent-accuracy accelerated neural-network training using analogue memory. *Nature* 558(7708):60–67.

Burrello, A.; Marchioni, A.; Brunelli, D.; et al. 2019. Embedding principal component analysis for data reduction in structural health monitoring on low-cost iot gateways. *International Conference on Computing Frontiers* 235–239.

Henriksen, A., and Ward, R. 2019. Adaoja: Adaptive learning rates for streaming pca. *arXiv:1905.12115*.

Hoskins, B. D.; Daniels, M. W.; Huang, S.; et al. 2019. Streaming batch eigenupdates for hardware neural networks. *Frontiers in neuroscience* 13:793.

Oja, E. 1982. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology* 15(3):267–273.

Strubell, E.; Ganesh, A.; and McCallum, A. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.