Gradient Diversity:
a Key Ingredient for Scalable Distributed Learning

Dong Yin dongyin@berkeley.edu
Ashwin Pananjady ashwinpm@berkeley.edu
UC Berkeley
Max Lam maxlam@stanford.edu
Stanford University
Dimitris Papailiopoulos dimitris@papail.io
UW Madison
Kannan Ramchandran kannanr@berkeley.edu
Peter Bartlett peter@berkeley.edu
UC Berkeley

Abstract

It has been experimentally observed that distributed implementations of mini-batch stochastic gradient descent (SGD) algorithms exhibit speedup saturation and decaying generalization ability beyond a particular batch-size. In this work, we present an analysis hinting that high similarity between concurrently processed gradients may be a cause of this performance degradation. We introduce the notion of *gradient diversity* that measures the dissimilarity between concurrent gradient updates, and show its key role in the convergence and generalization performance of mini-batch SGD. We also establish that heuristics similar to DropConnect, Langevin dynamics, and quantization, are provably diversity-inducing mechanisms, and provide experimental evidence indicating that these mechanisms can indeed enable the use of larger batches without sacrificing accuracy and lead to faster training in distributed learning.

1 Introduction

In recent years, deploying algorithms on distributed computing units has become the *de facto* architectural choice for large-scale machine learning. Distributed optimization has gained significant traction with a large body of recent work establishing near-optimal speedup gains on both convex and nonconvex objectives [30, 15, 10, 41, 26, 20, 12, 4], and several state-of-the-art publicly available (distributed) machine learning frameworks, such as Tensorflow [1] and MXNet [5] offer distributed implementations of popular learning algorithms.

Mini-batch SGD is the algorithmic cornerstone for several of these distributed frameworks. During a distributed iteration of mini-batch SGD, a master node stores a global model, and P worker nodes compute gradients for B data points, sampled from a total of n training data (*i.e.,* B/P samples per worker per iteration), with respect to the same global model; the parameter B is commonly referred to as the batch-size. The master, after receiving these B gradients, applies them to the model and sends the updated model back to the workers; this is the equivalent of one round of communication.

These algorithms are typically used to solve empirical risk minimization problems, where we are interested in minimizing the *population* risk $R(\mathbf{w}) = \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[f(\mathbf{w};\mathbf{z})]$, but have access to it through i.i.d. samples from $\mathcal{D}$, denoted by $\mathcal{S} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$, and thus minimize the empirical risk $R_{\mathcal{S}}(\mathbf{w}) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w};\mathbf{z}_i)$. For simplicity, denote $R_{\mathcal{S}}(\mathbf{w})$ and $f(\mathbf{w};\mathbf{z}_i)$ by $F(\mathbf{w})$ and $f(\mathbf{w};\mathbf{z}_i)$, respectively. The iterations of mini-batch SGD then take the form $\mathbf{w}_{(k+1)B} = \mathbf{w}_{kB} - \gamma \sum_{\ell=kB}^{(k+1)B-1} \nabla f_{s_\ell}(\mathbf{w}_{kB})$, where each index s_i is drawn uniformly at random from $[n]$, with replacement, and the gradient computations are divided among the workers[1]. Here, we use $\mathbf{w}$ with subscript kB to denote the model we obtain after k distributed iterations, *i.e.,* a total of kB gradient updates.

Unfortunately, near-optimal scaling for distributed variants of mini-batch SGD is only possible for up to tens of compute nodes. Several studies [10, 32] indicate that there is a significant gap between ideal and realizable

[1] In related work, there is a normalization of $1/B$ included in the gradient step, here, *without loss of generality* we subsume that in the step-size γ. Our assumption of constant step-size is for convenience.

OPTML 2017: 10th NIPS Workshop on *Optimization for Machine Learning* (NIPS 2017).

speedups when scaling out to hundreds of compute nodes. This commonly observed phenomenon is referred to as *speedup saturation*. A key cause of speedup saturation is the communication overheads of mini-batch SGD.

Ultimately, the batch-size B controls a crucial performance trade-off between communication costs and convergence speed, as observed and analyzed in several studies [34, 37, 16]. When using large batch-sizes, we observe large speedup gains per pass (*i.e.,* per n gradient computations), as shown in Figure 1, due to fewer communication rounds. However, as shown in Figure 2, to achieve a desired level of accuracy for larger batches, we may need a larger number of passes over the dataset, resulting in *overall* slower computation that leads to speedup saturation. Furthermore, recent work shows that large batch sizes lead to models that generalize worse [22], and efforts have been made to improve the generalization ability [19]. Here, generalization is measured by the gap $|R_{\mathcal{S}}(\mathbf{w}) - R(\mathbf{w})|$ that quantifies the performance discrepancy of the model $\mathbf{w}$ between the empirical and population risks.

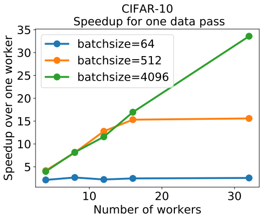

CIFAR-10
Speedup for one data pass

Figure 1: Speedup gains for a single data pass and various batch-sizes (cuda-convnet, CIFAR-10)

Our contributions: We introduce the notion of *gradient diversity* that measures the dissimilarity between concurrent gradient updates, and show that mini-batch SGD does not suffer from speedup saturation and generalization degradation as long as we choose the batch-size no more than a fundamental bound implied by gradient diversity. We also establish that some heuristics in large scale optimization are provably diversity-inducing, and provide experimental evidence to show their effectiveness.

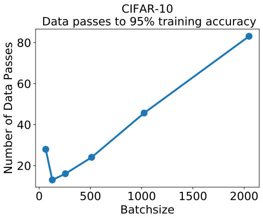

CIFAR-10
Data passes to 95% training accuracy

Figure 2: Number of data passes to reach 95% accuracy (cuda-convnet variant, CIFAR-10)

2 Related Work

We focus on some key papers that are closest to our work. Dekel et al. [11] analyze mini-batch SGD on non-strongly convex functions and propose $B = \mathcal{O}(\sqrt{T})$ as an optimal choice for batch-size. In contrast, our work provides a data-dependent principle for the choice of batch-size, and it holds without the requirement of convexity. Even in the regime where the result in [11] is valid, depending on the problem, our result may still provide better bounds on the batch-size than $\mathcal{O}(\sqrt{T})$ (*e.g.,* in the sparse conflict setting [30]). Other papers that analyze minibatch SGD show weak linear convergence rate for strongly convex functions [13], propose optimization algorithms for choosing the batch-size [9], and develop weighted sampling techniques [28, 42]. In empirical studies, it has been observed that more diversity in the data allows more parallelism [6]. Data-dependent thresholds for batch-size have been developed for some specific problems such as least squares [21] and SVM [34]. In particular, for least square problems, Jain et al. [21] propose a bound on batch-size similar to our measure of gradient diversity; however, our result holds for a much wider range of problems including nonconvex setups, and can be used to motivate several heuristics.

Several other mini-batching algorithms have been proposed; including mini-batch proximal algorithms [25, 37, 38], accelerated methods [7], mini-batch SDCA [33, 35], and the combination of mini-batching and variance reduction such as Acc-Prox-SVRG [29] and mS2GD [23]. We emphasize that although different mini-batching algorithms can be designed for particular problems and may work better in particular regimes, especially in the convex setting, these algorithms are usually more difficult to implement in distributed learning frameworks, and can introduce additional communication costs. A few other algorithms have been recently proposed to reduce the communication cost by inducing sparsity in the gradients, *e.g.,* , QSGD [2] and TernGrad [40].

On the generalization side, bounds for SGD were shown by Hardt et al. [17] and Jain et al. [21] (for least squares regression) via stability analysis and operator methods, respectively. Variance reduction methods are also used to develop algorithms with good generalization performance [14, 8].

3 Main Results

Convergence Our results for convergence are dependent on definitions of *gradient diversity*, which measures the dissimilarity between individual gradients.

Definition 1 (gradient diversity)**.** *We refer to the following ratio as gradient diversity:*

$$\Delta_D(\mathbf{w}) := \frac{\sum_{i=1}^n \|\nabla f_i(\mathbf{w})\|_2^2}{\|\sum_{i=1}^n \nabla f_i(\mathbf{w})\|_2^2}, = \frac{\sum_{i=1}^n \|\nabla f_i(\mathbf{w})\|_2^2}{\sum_{i=1}^n \|\nabla f_i(\mathbf{w})\|_2^2 + \sum_{i \neq j} \langle \nabla f_i(\mathbf{w}), \nabla f_j(\mathbf{w}) \rangle}.$$

The gradient diversity is clearly a data-dependent quantity, and can be shown to be bounded below by simple functions of the data in many cases, *e.g.,* for cases of generalized linear models and sparse conflict graphs [30]. We define the batch-size bound $B_D(\mathbf{w}) := n\Delta_D(\mathbf{w})$.

We are now ready to state our convergence results for particular classes of functions[2]. In all of these results, we assume that $B \leq \delta B_D(\mathbf{w})+1$ for all $\mathbf{w} \in \mathcal{W}$; the takeaway message is that we can guarantee convergence within a similar number of *gradient updates* as serial SGD provided that this condition is met. Note that this ensures linear speed-ups. In the following, we define $\mathbf{w}^* \in \arg\min_{\mathbf{w}\in\mathcal{W}} F(\mathbf{w})$, $F^* := \min_{\mathbf{w}\in\mathcal{W}} F(\mathbf{w})$, $D_0 := \|\mathbf{w}_0 - \mathbf{w}^*\|_2^2$, and assume $\frac{1}{n}\sum_{i=1}^n \|\nabla f_i(\mathbf{w})\|_2^2 \leq M^2$, $\forall \mathbf{w} \in \mathcal{W}$.

Theorem 1 (smooth functions)**.** *Suppose that $F(\mathbf{w})$ is β-smooth, $\mathcal{W} = \mathbb{R}^d$, and use step-size $\gamma = \frac{\epsilon}{\beta M^2}$. Then, after $T \geq \frac{2}{\epsilon^2}M^2\beta(F(\mathbf{w}_0) - F^*)$ gradient updates, $\min_{k=0,\ldots,T/B-1}\mathbb{E}[\|\nabla F(\mathbf{w}_{kB})\|_2^2] \leq (1+\frac{\delta}{2})\epsilon$.*

Theorem 2 (PL functions)**.** *Suppose that $F(\mathbf{w})$ is β-smooth, μ-PL, $\mathcal{W} = \mathbb{R}^d$, and use step-size $\gamma = \frac{2\epsilon\mu}{M^2\beta}$, and batch-size $B \leq \frac{1}{2\gamma\mu}$. Then, after $T \geq \frac{M^2\beta}{4\mu^2\epsilon}\log(\frac{2(F(\mathbf{w}_0)-F^*)}{\epsilon})$ gradient updates, we have $\mathbb{E}[F(\mathbf{w}_T) - F^*] \leq (1+\frac{\delta}{2})\epsilon$.*

For convex loss functions, we emphasize that there have been a lot of studies that establish similar rates, without explicitly using our notion of gradient diversity [13, 21, 34]. We emphasize the general form of our characterization that is essentially identical across convex and nonconvex objectives.

Theorem 3 (convex functions)**.** *Suppose that $F(\mathbf{w})$ is convex, and use step-size $\gamma = \frac{\epsilon}{M^2}$. Then, after $T \geq \frac{M^2 D_0}{\epsilon^2}$ gradient updates, we have $\mathbb{E}[F(\frac{B}{T}\sum_{k=0}^{\frac{T}{B}-1}\mathbf{w}_{kB}) - F^*] \leq (1+\frac{\delta}{2})\epsilon$.*

Theorem 4 (strongly convex functions)**.** *Suppose that $F(\mathbf{w})$ is λ-strongly convex, and use step-size $\gamma = \frac{\epsilon\lambda}{M^2}$ and batch-size $B \leq \frac{1}{2\lambda\gamma}$. Then, after $T \geq \frac{M^2}{2\lambda^2\epsilon}\log(\frac{2D_0}{\epsilon})$ gradient updates, we have $\mathbb{E}[\|\mathbf{w}_T - \mathbf{w}^*\|_2^2] \leq (1+\frac{\delta}{2})\epsilon$.*

We can also show that our convergence result is optimal in the worst-case, for strongly convex objectives, but we omit this result due to space constraints. Another interesting consequence is that it validates many heuristics that exist in the literature as *diversity-inducing mechanisms*. These include DropConnect (DC) [36], stochastic gradient Langevin dynamics (SGLD) [39], and quantization (Quant) [2], which are used for improving large scale optimization. Using the abbreviation DIM for any such diversity-inducing mechanism, we note that mini-batch SGD is modified in the following way: when each data point i is sampled, instead of making gradient update $\nabla f_i(\mathbf{w})$, the algorithm updates with a random surrogate vector $\mathbf{g}_i^{\mathsf{DIM}}(\mathbf{w})$ by introducing some additional randomness, which is acquired i.i.d. across data points and iterations. The corresponding gradient diversity is defined analogously to Definition 1 as is the batch-size bound $B_D^{\mathsf{DIM}}(\mathbf{w})$; our key takeaway is that, for any $\mathsf{DIM} \in \{\mathsf{DC}, \mathsf{SGLD}, \mathsf{Quant}\}$, and any $\mathbf{w} \in \mathcal{W}$ with $B_D(\mathbf{w}) \leq n$, we have $B_D^{\mathsf{DIM}}(\mathbf{w}) \geq B_D(\mathbf{w})$.

Generalization We define the *generalization error* of the algorithm A as $\epsilon_{\text{gen}}(A) := \mathbb{E}_{\mathcal{S},A}[R_{\mathcal{S}}(A(\mathcal{S})) - R(A(\mathcal{S}))]$. In [3], Bousquet and Elisseef show the equivalence between the generalization error and algorithmic stability. The stability of mini-batch SGD is governed by the *differential gradient diversity*, defined as follows.

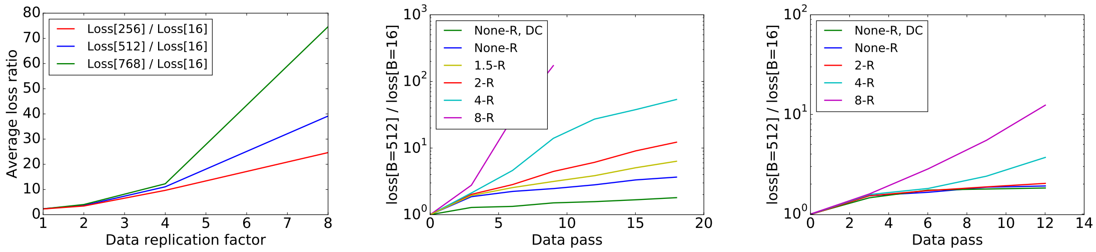
Definition 2 (differential gradient diversity, batch-size bound)**.** *For any $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$, $\mathbf{w} \neq \mathbf{w}'$, the differential gradient diversity and batch-size bound are given by*

$$\overline{\Delta}_D(\mathbf{w},\mathbf{w}') := \frac{\sum_{i=1}^n \|\nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}')\|_2^2}{\|\sum_{i=1}^n \nabla f_i(\mathbf{w}) - \nabla f_i(\mathbf{w}')\|_2^2}, \text{ and } \overline{B}_D(\mathbf{w},\mathbf{w}') := n\overline{\Delta}_D(\mathbf{w},\mathbf{w}').$$

Although it is a distinct measure, differential gradient diversity shares similar properties with gradient diversity; *e.g.,* it can be shown to be bounded below for generalized linear models and sparse conflicts; DropConnect and SGLD also induce differential gradient diversity.

We now analyze the stability (generalization) of mini-batch SGD via differential gradient diversity. We asume that, for each $\mathbf{z} \in \mathcal{Z}$, the loss function $f(\mathbf{w};\mathbf{z})$ is convex, L-Lipschitz and β-smooth in $\mathcal{W}$. Hardt et al. [17] analyzed such functions for serial SGD, and showed that stability is guaranteed up to a particular step-size $\bar{\gamma}$ (the bound takes multiple forms depending on the function class).

[2] The definitions of smooth, convex, and strongly convex functions are standard; we say a function F is μ-PL if $\frac{1}{2}\|\nabla F(\mathbf{w})\|_2^2 \geq \mu(F(\mathbf{w}) - F(\mathbf{w}^*))$ [31, 27].



(a) **(b)** **(c)**

Figure 3: Data replication. Here, 2-R, 4-R, etc represent 2-replication, 4-replication, etc, and DC stands for DropConnect. (a) Logistic regression with two classes of CIFAR-10 (b) Cuda convolutional neural network (c) Residual network. For (a), we plot the average loss ratio during all the iterations of the algorithm, and average over 10 experiments; for (b), (c), we plot the loss ratio as a function of the number of passes over the entire dataset, and average over 3 experiments. With larger replication factors, the convergence gap increases.

Our result is stated informally in Theorem 5, and holds for both convex and strongly convex functions. Here, $\overline{\gamma}$ is the step-size upper bound required to guarantee stability of serial SGD, and differently from the convergence results, we treat $\overline{B}_D(\mathbf{w}, \mathbf{w}')$ as a random variable defined by the sample $\mathcal{S}$.

Theorem 5 (informal stability result)**.** *Suppose that, with high probability, the batch-size* $B \lesssim \overline{B}_D(\mathbf{w}, \mathbf{w}')$ *for all* $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$*,* $\mathbf{w} \neq \mathbf{w}'$*. Then, after the same number of gradient updates, the generalization errors of mini-batch SGD and serial SGD satisfy* $\epsilon_{gen}(\text{minibatch SGD}) \lesssim \epsilon_{gen}(\text{serial SGD})$*, and such a guarantee holds for any step-size* $\gamma \lesssim \overline{\gamma}$*.*

Therefore, our main message is that, if with high probability, batch-size B is smaller than $\overline{B}_D(\mathbf{w}, \mathbf{w}')$ for all $\mathbf{w}, \mathbf{w}'$, mini-batch SGD and serial SGD can be both stable in roughly the *same range* of step-sizes, and the generalization error of mini-batch SGD and serial SGD are roughly the *same*. Thus, if batch-size B is not too large compared with the bound implied by differential gradient diversity, mini-batch SGD can achieve both speedup and good generalization ability.

4 Experiments

We conduct experiments to justify our theoretical results on convergence and stability. For lack of space, we present only the former here. Our neural network experiments are all implemented in Tensorflow and run on Amazon EC2 p2.xlarge instances.

Convergence We conduct the experiments on a logistic regression model and two deep neural networks (a cuda convolutional neural network [24] and a deep residual network [18]) with cross-entropy loss running on CIFAR-10 dataset. These results are presented in Figure 3. We use data replication to implicitly construct datasets with different gradient diversity. By replication with a factor r (or r-replication), we mean picking a random $1/r$ fraction of the data and replicating it r times. Across all configurations of batch-sizes, we tune our (constant) step-size to maximize convergence, *e.g.,* to minimize training time. The sample size does not change by data replication, but gradient diversity conceivably gets smaller while we increase r. We use the ratio of the loss function for large batch-size SGD (*e.g.,* $B = 512$) to the loss for small batch-size SGD (*e.g.,* $B = 16$) to measure the negative effect of large batch sizes on the convergence rate. When this ratio gets larger, the algorithm with the large batch-size is converging slower. We can see from the figures that while we increase r, the large batch size instances indeed perform worse, and the large batch instance performs the best when we have DropConnect, due to its diversity-inducing effect, as discussed in the previous sections. This experiment thus validates our theoretical findings.

Diversity-inducing Mechanisms We finally implement diversity-inducing mechanisms in a distributed setting with 2 workers and test the speedup gains. We use a convolutional neural network on MNIST and implement DropConnect with drop probability $p_{\text{drop}} = 0.4, 0.5$. We tune the step-size γ and batch-size B for vanilla mini-batch SGD and the diversity-induced setting, and find the (γ, B) pair that gives the fastest convergence for each setting. For instance, while comparing wall-clock times taken for 95% training accuracy, the two instantiations of DropConnect result in 31% and 25%- speedup gains, respectively. Indeed, the the batch-size gain afforded by DropConnect – the best batch-size for vanilla mini-batch SGD is 256, while with the diversity-inducing mechanism, it becomes 512 – is able to dwarf the noise in gradient computation. Reducing communication cost thus has the biggest effect on runtime, more so than introducing additional variance in stochastic gradient computations.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] D. Alistarh, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Randomized quantization for communication-optimal stochastic gradient descent. *arXiv preprint arXiv:1610.02132*, 2016.

[3] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2 (Mar):499–526, 2002.

[4] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[5] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv:1512.01274*.

[6] T. Chilimbi, Y. Suzue, J. Apacible, and K. Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX OSDI 14*, pages 571–582, 2014.

[7] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pages 1647–1655, 2011.

[8] H. Daneshmand, A. Lucchi, and T. Hofmann. Starting small-learning with adaptive sample sizes. In *International conference on machine learning*, pages 1463–1471, 2016.

[9] S. De, A. Yadav, D. Jacobs, and T. Goldstein. Big batch sgd: Automated inference using adaptive batch sizes. *arXiv preprint arXiv:1610.05792*, 2016.

[10] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *NIPS*, pages 1223–1231, 2012.

[11] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

[12] J. Duchi, M. I. Jordan, and B. McMahan. Estimation, optimization, and parallelism when data is sparse. In *NIPS*, pages 2832–2840, 2013.

[13] M. P. Friedlander and M. Schmidt. Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*, 34(3):A1380–A1405, 2012.

[14] R. Frostig, R. Ge, S. M. Kakade, and A. Sidford. Competing with the empirical risk minimizer in a single pass. In *Conference on learning theory*, pages 728–763, 2015.

[15] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD*, pages 69–77. ACM, 2011.

[16] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[17] M. Hardt, B. Recht, and Y. Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[19] E. Hoffer, I. Hubara, and D. Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *arXiv preprint arXiv:1705.08741*, 2017.

[20] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan. Communication-efficient distributed dual coordinate ascent. In *NIPS*, pages 3068–3076, 2014.

[21] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford. Parallelizing stochastic approximation through mini-batching and tail-averaging. *arXiv preprint arXiv:1610.03774*, 2016.

[22] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[23] J. Konečnỳ, J. Liu, P. Richtárik, and M. Takáč. Mini-batch semi-stochastic gradient descent in the proximal setting. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):242–255, 2016.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[25] M. Li, T. Zhang, Y. Chen, and A. J. Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD*, pages 661–670. ACM, 2014.

[26] J. Liu, S. Wright, C. Re, V. Bittorf, and S. Sridhar. An asynchronous parallel stochastic coordinate descent algorithm. In *Proceedings of ICML 14*, pages 469–477, 2014.

[27] S. Lojasiewicz. A topological property of real analytic subsets. *Coll. du CNRS, Les ´equations aux d´eriv´ees partielles*, pages 87—-89, 1963.

[28] D. Needell and R. Ward. Batched stochastic gradient descent with weighted sampling. *arXiv preprint arXiv:1608.07641*, 2016.

[29] A. Nitanda. Stochastic proximal gradient descent with acceleration techniques. In *NIPS*, pages 1574–1582, 2014.

[30] F. Niu, B. Recht, C. Re, and S. Wright. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS*, pages 693–701, 2011.

[31] B. T. Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 3(4):643–653, 1963.

[32] H. Qi, E. R. Sparks, and A. Talwalkar. Paleo: A performance model for deep neural networks. 2016.

[33] S. Shalev-Shwartz and T. Zhang. Accelerated mini-batch stochastic dual coordinate ascent. In *NIPS*, pages 378–385, 2013.

[34] M. Takác, A. S. Bijral, P. Richtárik, and N. Srebro. Mini-batch primal and dual methods for svms. In *ICML (3)*, pages 1022–1030, 2013.

[35] M. Takáč, P. Richtárik, and N. Srebro. Distributed mini-batch sdca. *preprint arXiv:1507.08322*, 2015.

[36] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1058–1066, 2013.

[37] J. Wang, W. Wang, and N. Srebro. Memory and communication efficient distributed stochastic optimization with minibatch prox. *arXiv preprint arXiv:1702.06269*, 2017.

[38] W. Wang and N. Srebro. Stochastic nonconvex optimization with large minibatches. *arXiv preprint arXiv:1709.08728*, 2017.

[39] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of ICML*, pages 681–688, 2011.

[40] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *arXiv preprint arXiv:1705.07878*, 2017.

[41] H. Yun, H.-F. Yu, C.-J. Hsieh, S. Vishwanathan, and I. Dhillon. Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *arXiv:1312.0193*, 2013.

[42] C. Zhang, H. Kjellstrom, and S. Mandt. Stochastic learning on imbalanced data: Determinantal point processes for mini-batch diversification. *arXiv preprint arXiv:1705.00607*, 2017.