Supplementary Material of SGD_Tucker: A Novel Stochastic Optimization Strategy for Parallel Sparse Tucker Decomposition

Hao Li, *Student Member*, *IEEE*, Zixuan Li, Kenli Li, *Senior Member*, *IEEE*, Jan S. Rellermeyer, Lydia Chen, *Senior Member*, *IEEE*, Keqin Li, *Fellow*, *IEEE*.

1 Basic Optimization Models

The properties of $\mathbf{A}^{(n)}$, $n \in \{N\}$ accord to the task which determines the decomposition algorithm, i.e., in HOOI [1], each $a^{(n)}_{:,j_n}$, $j_n \in \{J_n\}$ where $n \in \{N\}$ is orthogonal and the L_2 norm constraints $\|a^{(n)}_{:,j_n}\|_2 = 1$, $j_n \in \{J_n\}$, $n \in \{N\}$. The orthogonality and unity of HOOI can track the optimal low-rank orthogonal subspace, as shown in Algorithm 1.

Algorithm 1 A Training Epoch of HOOI.

Input: Sparse tensor $\mathcal{X}$, Factor Matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, $n \in \{N\}$.
Output: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, $n \in \{N\}$;
Core Tensor $\mathcal{G}$ Nth order tensor $\in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$.

```
1: for n from 1 to N do
2:    Y ← X ×(1) A^(1)^T ×(2) ··· ×(n−1) A^(n−1)^T ×(n+1) A^(n+1)^T ··· ×(N) A^(N)^T;
3:    A^(n) ← J_n leading left singular vectors of Y_(n);
4: end for
5: G ← X ×(1) A^(1)^T ×(2) ··· ×(n) A^(n)^T ×(n+1) ··· ×(N) A^(N)^T.
```

The other methods are to infer the low-rank factor matrices $\mathbf{A}^{(n)}$, $n \in \{N\}$ and core tensor $\mathcal{G}$ with L_2 norm regularization [1–5]. L_2 norm regularization can keep the promise of the smoothness [6–9] for the low-rank factor matrices $\mathbf{A}^{(n)}$, $n \in \{N\}$ and the core tensor $\boldsymbol{\mathcal{G}}$. The optimization algorithms are divided into two classes, i.e., dense condition and sparse condition. The optimization problem for dense tensor is presented as:

$$\begin{aligned}\underset{\mathbf{A}^{(n)}, n\in\{N\}, \boldsymbol{\mathcal{G}}}{\arg\min} & f\left(\boldsymbol{\mathcal{X}}, \{\mathbf{A}^{(n)}\}, \boldsymbol{\mathcal{G}}\right) \\ &= \left\|\boldsymbol{\mathcal{X}} - \widehat{\boldsymbol{\mathcal{X}}}\right\|_2^2 + \lambda_{\boldsymbol{\mathcal{G}}}\left\|\boldsymbol{\mathcal{G}}\right\|_2^2 + \lambda_{\mathbf{A}}\left\|\mathbf{A}^{(n)}\right\|_2^2,\end{aligned} \tag{1}$$

where $\widehat{\boldsymbol{\mathcal{X}}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_n \mathbf{A}^{(n)} \times_{n+1} \cdots \times_N \mathbf{A}^{(N)}$ and $\lambda_{\boldsymbol{\mathcal{G}}}$ and $\lambda_{\mathbf{A}}$ are the regularization parameters for core tensor and low-rank factor matrices, respectively. The optimization objective (1) involves variables multiplication, which is non-convex. The non-convex problem can be tackled by convex solution via updating a variable and fixing the others. The optimization problem (1) can be split into updating core tensor $\boldsymbol{\mathcal{G}}$ (nth vectorization form $g^{(n)}$) and updating low-rank factor matrices $\mathbf{A}^{(n)}$, $n \in \{N\}$ as following [10–12]:

$$\begin{cases}\underset{g^{(n)}}{\arg\min} f\left(g^{(n)} \middle| x^{(n)}, \{\mathbf{A}^{(n)}\}, g^{(n)}\right) \\ \qquad = \left\|x^{(n)} - \widehat{x}^{(n)}\right\|_2^2 + \lambda_{g^{(n)}}\left\|g^{(n)}\right\|_2^2; \\ \underset{\mathbf{A}^{(n)}, n\in\{N\}}{\arg\min} f\left(\mathbf{A}^{(n)} \middle| \boldsymbol{X}^{(n)}, \{\mathbf{A}^{(n)}\}, \mathbf{G}^{(n)}\right) \\ \qquad = \left\|\boldsymbol{X}^{(n)} - \widehat{\boldsymbol{X}}^{(n)}\right\|_2^2 + \lambda_{\mathbf{A}}\left\|\mathbf{A}^{(n)}\right\|_2^2,\end{cases} \tag{2}$$

where $\widehat{x}^{(n)} = \mathbf{H}^{(n)} g^{(n)}$, $\widehat{\boldsymbol{X}}^{(n)} = \mathbf{A}^{(n)}\mathbf{G}^{(n)}\mathbf{S}^{(n)^T}$ and $\mathbf{S}^{(n)} = \mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}$. Equ. (2) discusses the basic optimization model for the two key parts, i.e., core tensor $\mathcal{G}$ and factor matrices $\mathbf{A}^{(n)}$, $n \in \{N\}$. The convex optimization objective (2) for core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{J_1 \times \cdots \times J_n \times \cdots \times J_N}$ cooperates with parameters $\{\boldsymbol{\mathcal{X}}, \mathbf{H}^{(n)}\}$ and the gradient of $g^{(n)}$ on (2) is:

$$\begin{aligned}&\frac{\partial f\left(g^{(n)} \middle| x^{(n)}, \{\mathbf{A}^{(n)}\}, g^{(n)}\right)}{\partial g^{(n)}} \\ &= -\mathbf{H}^{(n)^T} x^{(n)} + \mathbf{H}^{(n)^T}\mathbf{H}^{(n)} g^{(n)} + \lambda_{g^{(n)}} g^{(n)},\end{aligned} \tag{3}$$

- *Hao Li, Zixuan Li, Kenli Li and Keqin Li are with the College of Computer Science and Electronic Engineering, Hunan University, and National Supercomputing Center in Changsha, Hunan, China, 410082.*
- *Corresponding author: Kenli Li, Keqin Li.*
  E-mail: lihao123@hnu.edu.cn (H.Li-9@tudelft.nl), zixuanli@hnu.edu.cn, lkl@hnu.edu.cn, J.S.Rellermeyer@tudelft.nl, Y.Chen-10@tudelft.nl, lik@newpaltz.edu.
- *Hao Li is also with the TU Delft, Netherlands.*
- *J.S.Rellermeyer, Lydia Chen are with the TU Delft, Netherlands.*
- *Keqin Li is also with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.*

where:

$$\mathbf{H}^{(n)} =\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)} \otimes \mathbf{A}^{(n)}, \mathbf{H}^{(n)} \in R^{\prod_{n=1}^{N} I_n \times \prod_{n=1}^{N} J_n}, \quad (4)$$

and the element form is $h^{(n)}_{i,:} = a^{(N)}_{i_N,:} \odot \cdots \odot a^{(n+1)}_{i_{n+1},:} \odot a^{(n-1)}_{i_{n-1},:} \odot \cdots \odot a^{(1)}_{i_1,:} \odot a^{(n)}_{i_n,:}$, $h^{(n)}_{i,:} \in \mathbb{R}^{\prod_{n=1}^{N} J_n}$, $i \in \Omega^{(n)}_V$, $(i_1, \cdots, i_N) \in \Omega$, $\mathbf{H}^{(n)^T}\mathbf{H}^{(n)} = \mathbf{P}^{(N)} \otimes \cdots \otimes \mathbf{P}^{(n+1)} \otimes \mathbf{P}^{(n-1)} \otimes \cdots \otimes \mathbf{P}^{(1)} \otimes \mathbf{P}^{(n)}$, $\mathbf{P}^{(n)} = \mathbf{A}^{(n)^T}\mathbf{A}^{(n)}$, and $g^{(n)} \in \mathbb{R}^{\prod_{n=1}^{N} J_n}$. The overall space overheads for intermediate matrices $\{\mathbf{H}^{(n)}, \mathbf{H}^{(n)^T}\mathbf{H}^{(n)}\}$ are $\left\{O(\prod_{n=1}^{N} I_n J_n), O(\prod_{n=1}^{N} J_n^2)\right\}$, respectively, and the overall computational complexity for the gradient of core tensor (3) is $O\left(\prod_{n=1}^{N}(I_n J_n) + \prod_{n=1}^{N} J_n^2 + 3\prod_{n=1}^{N} J_n\right)$.

The gradient for the factor matrices $\mathbf{A}^{(n)}$, $n \in \{N\}$ on (2) is:

$$\frac{\partial f\left(\mathbf{A}^{(n)}\middle|\boldsymbol{X}^{(n)}, \{\mathbf{A}^{(n)}\}, \mathbf{G}^{(n)}\right)}{\partial \mathbf{A}^{(n)}} = -\boldsymbol{X}^{(n)}\mathbf{E}^{(n)^T} + \mathbf{A}^{(n)}\mathbf{E}^{(n)}\mathbf{E}^{(n)^T} + \lambda_{\mathbf{A}}\mathbf{A}^{(n)}, \quad (5)$$

where:

$$\begin{cases} \mathbf{S}^{(n)} = \mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)}; \\ \mathbf{S}^{(n)} \in \mathbb{R}^{\prod_{k=1,k\neq n}^{N} I_k \times \prod_{k=1,k\neq n}^{N} J_k}; \\ \mathbf{E}^{(n)} = \mathbf{G}^{(n)}\mathbf{S}^{(n)^T} \in \mathbb{R}^{J_n \times \prod_{k=1,k\neq n}^{N} I_k}, \end{cases} \quad (6)$$

and the element form is $s^{(n)}_{j,:} = a^{(N)}_{i_N,:} \odot \cdots \odot a^{(n+1)}_{i_{n+1},:} \odot a^{(n-1)}_{i_{n-1},:} \odot \cdots \odot a^{(1)}_{i_1,:}$, $s^{(n)}_{j,:} \in \mathbb{R}^{\prod_{k=1,k\neq n}^{N} J_k}$, $(i_n, j) \in \Omega^{(n)}_M$, and $(i_1, \cdots, i_N) \in \Omega$.

In common conditions, the original tensor is a HOHDST; thus, the space and time overheads for the intermediate matrices are huge which are not practical. There are some works that can construct the optimization objectives following the sparsity model, e.g., ALS [3, 4] and CD [5], etc. ALS should construct the Hessian matrices $\mathbf{C}^{(n)} \in \mathbb{R}^{J_n \times J_n}$, $n \in \{N\}$ and the element-wise form is presented as:

$$\begin{cases} g^{(n)} \leftarrow \left(\mathbf{C} + \lambda_{g^{(n)}}\mathbf{I}\right)^{-1} d, \mathbf{I} \in \mathbb{R}^{\prod_{n=1}^{N} J_n \times \prod_{n=1}^{N} J_n}; \\ a^{(n)}_{i_n,:} \leftarrow d^{(n)^T}_{i_n,:}\left(\mathbf{C}^{(n)} + \lambda_{\mathbf{A}}\mathbf{I}_n\right)^{-1}, \mathbf{I}_n \in \mathbb{R}^{J_n \times J_n}, \\ n \in \{N\}, \end{cases} \quad (7)$$

where

$$\begin{cases} d = h^{(n)^T}_{\Omega^{(n)}_V,:} x^{(n)}; \\ c_{j_1,j_2} = h^{(n)^T}_{\Omega^{(n)}_V,j_1} h^{(n)}_{\Omega^{(n)}_V,j_1}, j_1, j_2 \in \left\{\prod_{n=1}^{N} J_n\right\}; \\ d^{(n)}_{i_n,:} = x^{(n)}_{i_n,(\Omega^{(n)}_M)_{i_n}} \mathbf{E}^{(n)^T}_{i_n,(\Omega^{(n)}_M)_{i_n}}; \\ c^{(n)}_{j_1,j_2} = e^{(n)}_{j_1,(\Omega^{(n)}_M)_{i_n}} e^{(n)^T}_{j_1,(\Omega^{(n)}_M)_{i_n}}, j_1, j_2 \in \{J_n\}. \end{cases} \quad (8)$$

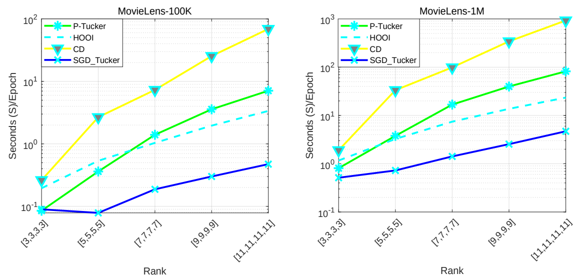
The computational complexity for $\left(\mathbf{C} + \lambda_{g^{(n)}}\mathbf{I}\right)^{-1}$ is $O\left((\prod_{n=1}^{N} J_n)^3\right)$ and the computational complexity for $\left(\mathbf{C}^{(n)} + \lambda_{\mathbf{A}}\mathbf{I}_n\right)^{-1}$ is $O\left(J_n^3\right)$. Thus, the total computational complexity is $O\left(\sum_{n=1}^{N} I_n J_n^3 + (\prod_{n=1}^{N} J_n)^3 + (\prod_{n=1}^{N} J_n)|\Omega| + (\prod_{n=1}^{N} J_n)^2|\Omega| + \sum_{n=1}^{N} J_n^2|\Omega| + \sum_{n=1}^{N} J_n|\Omega|\right)$ and the space overhead is $O\left(N|\Omega| + \sum_{n=1}^{N} I_n J_n + \prod_{n=1}^{N} J_n + \sum_{n=1}^{N} J_n^2 + (\prod_{n=1}^{N} J_n)^2\right)$. CD is a special version of ALS and CD updates each feature element in a feature vector discretely. In the sequel, CD neglects the successive reading and writing for a feature vector which will increase the data addressing overheads. The optimization objective for $\{g^{(n)}_\alpha, a^{(n)}_{i_n,j_n}\}$ is presented as:

$$\begin{cases} \underset{g^{(n)}_\alpha, \alpha \in \left\{\prod_{n=1}^{N} J_n\right\}}{\arg\min} f\left(g^{(n)}_\alpha \middle| x^{(n)}, \{\mathbf{A}^{(n)}\}, g^{(n)}\right) \\ \quad = \left\| x^{(n)} - \sum_{\beta\neq\alpha}^{\prod_{n=1}^{N} J_n} h^{(n)}_{\Omega^{(n)}_V,\beta} g^{(n)}_\beta - h^{(n)}_{\Omega^{(n)}_V,\alpha} g^{(n)}_\alpha \right\|_2^2 + \lambda_{g^{(n)}} \left\| g^{(n)}_\alpha \right\|_2^2; \\ \underset{a^{(n)}_{i_n,j_n}, n\in\{N\}}{\arg\min} f\left(a^{(n)}_{i_n,j_n} \middle| \mathbf{X}^{(n)}, \{\mathbf{A}^{(n)}\}, \mathbf{G}^{(n)}\right) \\ \quad = \left\| x^{(n)}_{i_n,(\Omega^{(n)}_M)_{i_n}} - \sum_{j\neq j_n}^{J_n} a^{(n)}_{i_n,j} e^{(n)}_{j,(\Omega^{(n)}_M)_{i_n}} - a^{(n)}_{i_n,j_n} e^{(n)}_{j_n,(\Omega^{(n)}_M)_{i_n}} \right\|_2^2 + \lambda_{\mathbf{A}} \left\| a^{(n)}_{i_n,j_n} \right\|_2^2. \end{cases} \quad (9)$$

The element-wise form for $\{g^{(n)}_\alpha, a^{(n)}_{i_n,j_n}\}$ is presented as:

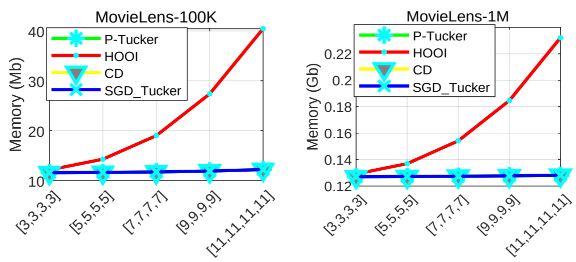
$$\begin{cases} g^{(n)}_\alpha \leftarrow \dfrac{h^{(n)^T}_{\Omega^{(n)}_V,\alpha}\left(x^{(n)} - \sum_{\beta\neq\alpha}^{\prod_{n=1}^{N} J_n} h^{(n)}_{\Omega^{(n)}_V,\beta} g^{(n)}_\beta\right)}{\lambda_{g^{(n)}} + h^{(n)^T}_{\Omega^{(n)}_V,\alpha} h^{(n)}_{\Omega^{(n)}_V,\alpha}}; \\ a^{(n)}_{i_n,j_n} \leftarrow \dfrac{\left(x^{(n)}_{i_n,(\Omega^{(n)}_M)_{i_n}} - \sum_{j\neq j_n}^{J_n} a^{(n)}_{i_n,j} e^{(n)}_{j,(\Omega^{(n)}_M)_{i_n}}\right) e^{(n)^T}_{j_n,(\Omega^{(n)}_M)_{i_n}}}{\lambda_{\mathbf{A}} + e^{(n)}_{j_n,(\Omega^{(n)}_M)_{i_n}} e^{(n)^T}_{j_n,(\Omega^{(n)}_M)_{i_n}}}. \end{cases} \quad (10)$$

HOOI, ALS and CD rely on the whole training set which results in high computation overhead and memory bottleneck especially in the situation of HOHDST.



(a) Movielen-100K (b) Movielen-1M

Fig. 1: Rank scalability for time overhead on full threads. The computational scalability for P−Tucker, HOOI, CD, and SGD_Tucker on the 2 datasets with successively increased number of total elements, i.e., Movielen-100K, Movielen-1M.

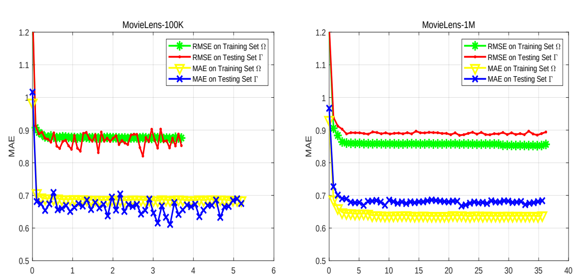


(a) Movielen-100K (b) Movielen-1M

Fig. 2: Rank scalability for memory overhead on a thread running. (On this work, GB refers to GigaBytes and MB refers to Megabytes). The space scalability for P−Tucker, CD, HOOI, and SGD_Tucker on the 2 small datasets with successively increased total elements, i.e., Movielen-100K, Movielen-1M.

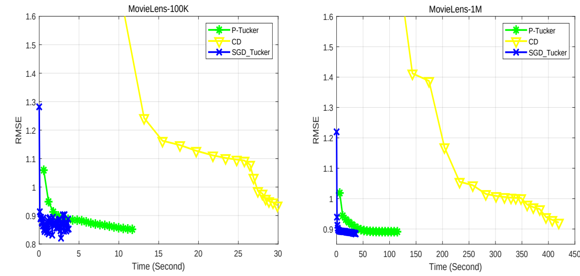
2 Experiments Results

Fig. 5 illustrates the computational time of the key parts, i.e., $TTMc$, *top-*N of $svds$, and the total computational time. The key and most time-consuming part of $TTMc$ is vectors Kronecker product which is computed by $kron$ of $Armadillo$ library and the index access code is borrowed from SPLATT of [1]. The SVD for the intermediate matrix of $TTMc$ is computed by *top-*N of $svds$ on $Armadillo$ library. As the Fig. 5 show, there are two



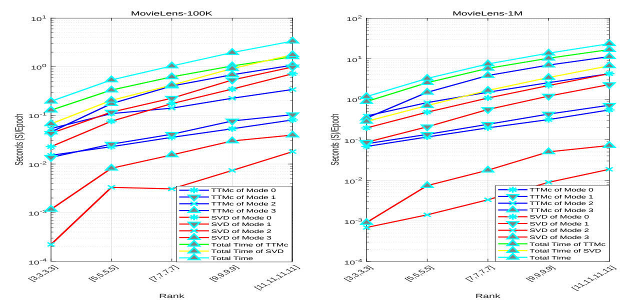
(a) Movielen-100K (b) Movielen-1M

Fig. 3: RMSE and MAE vs time for SGD_Tucker on training set Ω and testing set Γ

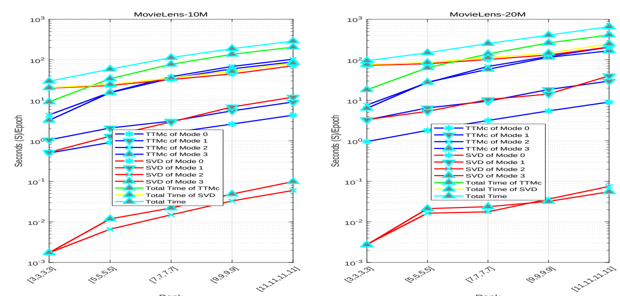


(a) Movielen-100K (b) Movielen-1M

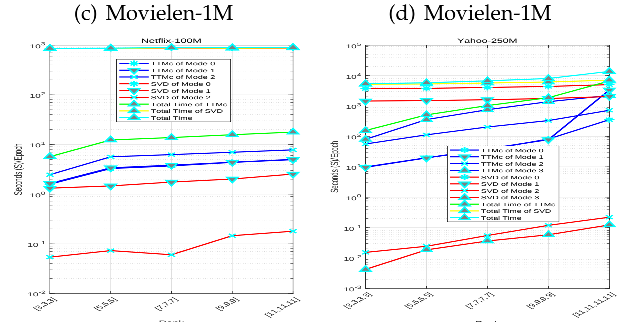
Fig. 4: RMSE comparison of SGD_Tucker, P−Tucker, and CD on 2 small datasets.



(a) Movielen-100K (b) Movielen-1M



(c) Movielen-1M (d) Movielen-1M



(e) Movielen-1M (f) Movielen-1M

Fig. 5: Computational time of $TTMc$ and *top-*N $svds$ based on the $Armadillo$ library on the 6 datasets. The experiment runs on full threads

regulations as: 1) the computational overhead for $TTMc$ is controlled by the degree of thread balance which is balance degree of $\sum_{i_n}^{belong\ to\ lth\ thread} |(\Omega_M^{(n)})_{i_n}|, n \in \{N\}$ for thread l. 2) the computational overhead for $svds$ is regulated by $\left\{ I_n \times \prod_{k \neq n}^{N} J_k \middle| n \in \{N\} \right\}$. From the results of Fig. 5 and Figs. 5 and 6 in main paper, the time scalability of HOOI is the same with SGD_Tucker. However, SGD_Tucker has lower computational overhead and less space overhead than HOOI.

References

[1] S. Smith and G. Karypis, "Accelerating the tucker decomposition with compressed sparse tensors," in *European Conference on Parallel Processing*. Springer, 2017, pp. 653–668.

[2] H. Ge, K. Zhang, M. Alfifi, X. Hu, and J. Caverlee, "Distenc: A distributed algorithm for scalable tensor completion on spark," in *2018 IEEE International Conference on Data Engineering*. IEEE, 2018, pp. 137–148.

[3] S. Oh, N. Park, S. Lee, and U. Kang, "Scalable tucker factorization for sparse tensors-algorithms and discoveries," in *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1120–1131.

[4] S. Oh, N. Park, J.-G. Jang, L. Sael, and U. Kang, "High-performance tucker factorization on heterogeneous platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2237–2248, 2019.

[5] M. Park, J.-G. Jang, and S. Lee, "Vest: Very sparse tucker factorization of large-scale tensors," *arXiv preprint arXiv:1904.02603*, 2019.

[6] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[7] H. Yu, R. Jin, and S. Yang, "On the linear speedup analysis of communication efficient momentum sgd for distributed non-convex optimization," in *International Conference on Machine Learning*, 2019, pp. 7184–7193.

[8] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1-2, pp. 83–112, 2017.

[9] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč, "Sarah: A novel method for machine learning problems using stochastic recursive gradient," in *Proceedings of the International Conference on Machine Learning*. JMLR. org, 2017, pp. 2613–2621.

[10] G. T. Minton and E. Price, "Improved concentration bounds for count-sketch," in *Proceedings of the annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 2014, pp. 669–686.

[11] A. Traoré, M. Berar, and A. Rakotomamonjy, "Singleshot: a scalable tucker tensor decomposition," in *Advances in Neural Information Processing Systems*, 2019, pp. 6301–6312.

[12] O. A. Malik and S. Becker, "Low-rank tucker decomposition of large tensors using tensorsketch," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 096–10 106.