Memory-Efficient 4-bit Preconditioned Stochastic Optimization

Jingyang Li[1] Kuangyu Ding[1] Kim-Chuan Toh[1] Pan Zhou[2]
[1] National University of Singapore [2] Singapore Management University
[1]{li_jingyang,kuangyud}@u.nus.edu mattohkc@nus.edu.sg [2]panzhou@smu.edu.sg

Abstract

Preconditioned stochastic optimization algorithms, exemplified by Shampoo, outperform first-order optimizers by offering theoretical convergence benefits and practical gains in large-scale neural network training. However, they incur substantial memory overhead due to the storage demands of non-diagonal preconditioning matrices. To address this, we introduce 4-bit quantization for Shampoo's preconditioners. We introduce two key methods: First, we apply Cholesky decomposition followed by quantization of the Cholesky factors, reducing memory usage by leveraging their lower triangular structure while better preserving spectral properties to minimize information loss. To our knowledge, this is the first quantization approach applied to Cholesky factors of preconditioners. Second, we incorporate error feedback in the quantization process, efficiently storing Cholesky factor and error state in the lower and upper triangular parts of the same matrix. Through extensive experiments, we demonstrate that combining Cholesky quantization with error feedback enhances memory efficiency and algorithm performance in large-scale deep-learning tasks. Theoretically, we also provide convergence proofs for quantized Shampoo under both smooth and non-smooth stochastic optimization settings.

1. Introduction

Deep learning has achieved significant advancements across numerous fields in recent years, including language modeling [7, 48], computer vision [16], and multi-modality [38]. These advancements are primarily driven by the scaling of model size, dataset volume, and computational power, as outlined in scaling laws that demonstrate the impact of increased resources on model performance [24, 25]. This trend of scaling has further extended into specialized domains such as finance [54], material science [56], and healthcare [30].

Along with the size growth of large-scale models, stochastic gradient descent (SGD) has become a widely adopted method for training thanks to its efficiency and simplicity [23, 41, 45]. However, adaptive gradient methods, e.g., Adagrad [17], Adam [26], and AdamW [35], apply a diagonal preconditioning to the gradient, which enables faster convergence than SGD [17, 63]. These adaptive methods have demonstrated empirical advantages in various applications [16, 58] and are now the standard optimizers for training large-scale neural networks.

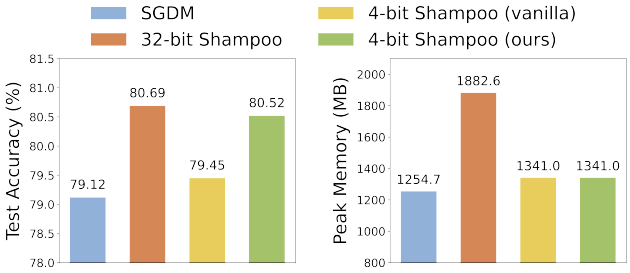


Figure 1. Comparison of test accuracy and peak memory usage for training ResNet-34 on CIFAR-100 dataset.

Building on adaptive gradient methods, full-matrix preconditioned gradient methods offer theoretically superior convergence by capturing richer correlations among parameters [17]. Despite these theoretical advantages, however, the memory overhead associated with non-diagonal matrices poses a significant challenge for large-scale neural networks, which can contain millions of parameters [16, 23, 34]. To address this, a range of efficient preconditioned gradient methods, such as K-FAC [36], Shampoo [22], K-BFGS [20], and AdaBK [60], aim to make full-matrix preconditioning computationally feasible by approximating the full-matrix preconditioner, e.g., block-diagonal precondition matrix. These algorithms have shown faster convergence rates in practice when compared to both SGD and adaptive gradient methods [3, 43, 60].

Nevertheless, these efficient preconditioned methods still impose substantial memory costs that restrict their scalability in practical and large-scale model applications. As shown in Fig. 1, the peak memory usage of methods like Shampoo remains significantly higher than SGD. While quantizing precondition matrices in these preconditioned methods from high-precision to low-precision, e.g., 32-bit to 4-bit, effectively reduces memory usage, it also inevitably introduces information loss, which, in turn, severely degrades the performance of these preconditioned

methods. This is validated by Fig. 1: compared with 32-bit Shampoo, 4-bit Shampoo enjoys much less memory cost but suffers from much worse performance. Therefore, for these efficient preconditioned methods, carefully designed strategies are essential to effectively compress precondition matrices without compromising optimization quality.

Contribution. We focus on Shampoo due to its simplicity, effectiveness, and popularity, aiming to enable efficient 4-bit quantization of preconditioners while maintaining the stability and efficiency of preconditioned gradient methods. Our main contributions are as follows:

- We introduce **Cholesky quantization** to improve memory efficiency and stability. Instead of directly quantizing preconditioners, we apply Cholesky decomposition and quantize the Cholesky factors. This reduces storage by half while better preserving spectral properties, mitigating quantization-induced information loss. To the best of our knowledge, this is the first quantization approach applied to Cholesky factors of preconditioners.
- We propose an **error feedback strategy** for Cholesky quantization to further reduce quantization error. Inspired by low-precision communication in distributed training [42, 46, 57], we maintain a 4-bit error state that exponentially-moving averages past quantization errors for stable error estimation. This state compensates the Cholesky factor at each iteration, reducing information loss. Moreover, the triangular structure of the Cholesky factor allows efficient joint storage with its error state.
- We establish **convergence guarantees** for quantized Shampoo in both smooth and nonsmooth stochastic nonconvex optimization. In the smooth case, our 4-bit Shampoo achieves the optimal $\mathcal{O}(\frac{1}{\sqrt{T}})$ convergence rate. In the nonsmooth case (e.g., ReLU-based networks [23]), we provide the first proof of global convergence for preconditioned gradient descent, showing convergence to stationary points under mild conditions.
- We develop **4-bit Shampoo** using these techniques and evaluate it on image classification with convolutional neural networks (CNNs) and vision transformers (ViTs). It outperforms vanilla 4-bit Shampoo, and significantly reduces memory usage compared to 32-bit Shampoo while maintaining comparable test performance, enabling larger models to be trained within existing resource constraints.

Discussion: Recent work [51] also studies preconditioner compression. Our approach improves it in two key aspects. First, we introduce error feedback (EF) to relieve the information loss from quantization, which [51] overlooks. EF helps retain critical curvature information during training. Second, our Cholesky Quantization (CQ) stores only lower-triangular factors, making it more memory-efficient than [51], which stores full orthogonal matrices. Moreover, Cholesky decomposition has lower computational cost ($\mathcal{O}(n_p^3/3)$) than the QR decomposition used in [51] ($\mathcal{O}(2n_p^3/3)$). These improvements lead to better efficiency and performance across models (see Tab. 7).

2. Related Work

Preconditioned Stochastic Optimization. Adaptive gradient methods are the most widely used preconditioned gradient methods in training neural networks, with Adagrad [17], RMSProp [47], and Adam [26] being notable examples. They use diagonal preconditioners to rescale the gradients, been shown to improve convergence in stochastic settings. Preconditioned gradient methods with non-diagonal preconditioners offer faster convergence in theory [17], and are widely explored recently due to faster convergence than adaptive gradient methods in practice [20, 22, 36, 60]. Among them, Shampoo [22] receives extensive concern for its simplicity and effectiveness [37, 50, 51], and it has been developed for large-scale distributed training [3, 43].

Quantization for Optimizers. Quantization has been widely used for gradient compression to enable efficient communication in large-scale optimization, particularly for distributed training [2, 49, 52]. Recent works have extended quantization to optimizer states—such as the momentum or second-moment estimates used by adaptive optimizers like Adam—to decrease peak memory usage during neural network training [15, 32]. Despite its computational efficiency, quantization incurs information loss, which can degrade algorithmic performance. To address this, ongoing research explores techniques such as error feedback compensation to mitigate these effects and improve robustness [40, 42].

3. Preliminaries

Here we introduce practical Shampoo from [3], and linear-square (linear-2) quantization [15] to compress the preconditioning matrices in our algorithm.

Notations. Let $\|A\|_F = \sqrt{\sum_{ij} A_{ij}^2}$ denote the Frobenius norm of a matrix A, and $\langle A, B\rangle = \sum_{ij} A_{ij}B_{ij}$ its inner product. The Kronecker product of A and B is denoted by $A \otimes B$. For a symmetric matrix H, $\lambda_{\max}(H)$ and $\lambda_{\min}(H)$ represent its maximum and minimum eigenvalues, respectively. For square symmetric matrices A and B, we write $A \preceq B$ if $B - A$ is positive semidefinite (PSD). Quantization and dequantization operations denoted by $\mathcal{Q}$ and $\mathcal{D}$.

3.1. Practical Shampoo

When minimizing a nonconvex stochastic objective:

$$F(W) \coloneqq \mathbb{E}_{\xi\sim\Xi}[F(W,\xi)], \tag{1}$$

where $W \in \mathbb{R}^{m\times n}$ is the parameter of the learning model, and data ξ is drawn from an unknown distribution Ξ. At each iteration, we sample a mini-batch of data points to compute the stochastic gradient $G \in \mathbb{R}^{m\times n}$, and use this stochastic gradient to update the model parameter W.

To accelerate convergence, Shampoo preconditions the stochastic gradient used in first-order optimizers. Specifically, at iteration k, it updates the preconditioning states L_k

and R_k with stochastic gradient G_k for preconditioning:

$$\begin{cases} L_k = \beta L_{k-1} + (1-\beta) G_k G_k^T, \\ R_k = \beta R_{k-1} + (1-\beta) G_k^T G_k, \\ \hat{G}_k = L_k^{-1/4} G_k R_k^{-1/4}, \end{cases} \tag{2}$$

where $\beta \in (0,1)$, and the $1/4$-th root inverse is computed efficiently using the Schur-Newton algorithm [21].

Next, first-order base optimizer $\mathcal{F}$ like SGD can use the preconditioned gradient $\tilde{G}_k$ in Eq. (2) to replace vanilla G_k for model update. For efficiency, Shampoo stores $(L_k, R_k, L_k^{-1/4}, R_k^{-1/4})$, and updates (L_k, R_k) for every T_1 iterations and $(L_k^{-1/4}, R_k^{-1/4})$ every T_2 iterations. See practical Shampoo algorithm in Algorithm 2 of Appendix A.

3.2. Linear Square Quantization for Compression

Quantization compresses tensors from high precision floating-point to low precision, reducing memory usage. Following [15, 32], we use block-wise quantization to mitigate outlier effects. Below, we introduce the quantization and dequantization processes, focusing on the two-dimensional tensor (matrix) case of Shampoo.

Quantization. For a floating-point matrix $X \in \mathbb{R}^{m\times n}$, we partition it into blocks of size $B \times B$, resulting in $P = \lceil m/B \rceil \times \lceil n/B \rceil$ blocks $\{X_p\}_{p=1}^P$. In each block X_p, a normalization factor $N_p = \max\{|X_p|\}$ scales elements to $[-1,1]$ via $\bar{X}_p = X_p / N_p$. Each element $\bar{x}_p$ in $\bar{X}_p$ is then quantized to a b-bit integer using a quantization mapping $\mathcal{M} : [0, 2^b - 1] \cap \mathbb{Z} \to [-1, 1]$, calculated by:

$$q_p = \underset{j \in [0, 2^b-1] \cap \mathbb{Z}}{\arg\min} |\bar{x}_p - \mathcal{M}(j)|. \tag{3}$$

Common quantization mappings include linear, dynamic, and quantile mappings [15, 32, 51]. Here we use a linear-2 mapping for simplicity and efficiency when $b = 4$:

$$\mathcal{M}(j) = \begin{cases} -(-1 + \frac{2j}{2^b-1})^2, & j < 2^{b-1} - 1, \\ 0, & j = 2^{b-1} - 1, \\ (-1 + \frac{2j}{2^b-1})^2, & j > 2^{b-1} - 1, \end{cases} \tag{4}$$

where $j \in \{0, 1, \ldots, 2^b - 1\}$. This block-wise quantization can be efficiently executed in parallel on GPUs [19, 59].

Dequantization. Dequantization $\mathcal{D}$ reverses the quantization process. For each quantized block Q_p, we map each element q_p back to $[-1,1]$ via $\bar{x}'_p = \mathcal{M}(q_p)$ to obtain $\bar{X}'_p$. We then restore the original scale using N_p, giving $X'_p = \mathcal{D}(Q_p) = N_p \bar{X}'_p$. Like quantization, dequantization is parallelizable on GPUs.

For block size $B \times B$, it balances accuracy and memory cost: smaller blocks improve accuracy but increase the number of normalization factors, raising memory overhead.

4. Memory-Efficient Shampoo Via Compensated Cholesky Quantization

We first present a direct quantization method to reduce the memory overhead of Shampoo's preconditioning matrices in Sec. 4.1. Then, in Sec. 4.2, we introduce a more memory-efficient Cholesky quantization approach that better preserves spectral properties to enhance vanilla quantization. Finally, in Sec. 4.3, we propose a compensation strategy to mitigate information loss from Cholesky quantization.

4.1. Quantization for Shampoo Compression

From Sec. 3.1, one knows that Shampoo requires storage of four preconditioning matrices $(L_k, R_k, L_k^{-1/4}, R_k^{-1/4})$, each sized $d \times d$, where d denotes the model parameter dimension. This brings much additional GPU memory cost, and becomes even more pronounced when training modern neural networks, which are often extremely high-dimensional. So reducing Shampoo's memory overhead is essential for efficient and scalable network training.

A straightforward approach is to use a quantizer $\mathcal{Q}$, e.g., the linear-2 quantization in Sec. 3.2, to compress the preconditioners in Shampoo for saving memory, and then adopt a dequantizer $\mathcal{D}$ to recover them for subsequent usage. Formally, at iteration k, we can compute two low-precision preconditioning states $(\bar{L}_k, \bar{R}_k)$ as

$$\begin{aligned} L_k &= \beta \mathcal{D}(\bar{L}_{k-1}) + (1-\beta) G_k G_k^T, \ \bar{L}_k = \mathcal{Q}(L_k), \\ R_k &= \beta \mathcal{D}(\bar{R}_{k-1}) + (1-\beta) G_k^T G_k, \ \bar{R}_k = \mathcal{Q}(R_k). \end{aligned} \tag{5}$$

In this work, we use 4-bit precision for efficient storage. For $\bar{L}_k^{-1/4}, \bar{R}_k^{-1/4}$, we update them as

$$\begin{aligned} L_k &= \mathcal{D}(\bar{L}_k), \ \bar{L}_k^{-1/4} = \mathcal{Q}((L_k + \lambda_{\max}^L \epsilon I_m)^{-1/4}), \\ R_k &= \mathcal{D}(\bar{R}_k), \ \bar{R}_k^{-1/4} = \mathcal{Q}((R_k + \lambda_{\max}^R \epsilon I_n)^{-1/4}), \end{aligned} \tag{6}$$

where, same as vanilla Shampoo, $\lambda_{\max}^L \epsilon I_m$ and $\lambda_{\max}^R \epsilon I_n$ provide numerical stability during the Schur-Newton iterations used to calculate the inverse $1/4$-th roots, in which $\lambda_{\max}^L$, $\lambda_{\max}^R$ are the maximal singular values of L_k, R_k, and ϵ is a small constant [60].

Accordingly, one can store 4-bit $(\bar{L}_k, \bar{R}_k, \bar{L}_k^{-1/4}, \bar{R}_k^{-1/4})$ instead of their original 32-bit versions, and dequantize them for usage, e.g., dequantizing $(\bar{L}_k^{-1/4}, \bar{R}_k^{-1/4})$ to compute preconditioned gradient in Eq. (2).

Despite its simplicity, direct quantization of preconditioners as in Eq. (5) and Eq. (6) can lead to performance degradation due to information loss, e.g., quantizing them from 32-bit to 4-bit precision. For instance, when training ViT-Small [16] on CIFAR-100 [27] with Shampoo using AdamW as the base optimizer, the 32-bit version Shampoo achieves 77.95% test accuracy, substantially outperforming the 4-bit quantized Shampoo, which reaches only 74.56%. Further experimental comparisons can be found in Sec. 6.

4.2. Efficient and Stable Cholesky Quantization

Here we introduce Cholesky quantization (CQ) to further improve memory efficiency and also stability of quantization in Sec. 4.1. Instead of quantizing L_k and R_k, we apply Cholesky decomposition on L_k and R_k, and quantize their corresponding Cholesky factors as $\bar{C}_k^L$ and $\bar{C}_k^R$ which

are lower triangular matrices and require much less storage. Formally, at iteration k, this process can be written as

$$\begin{aligned} &L_{k-1}=\mathcal{D}(\bar{C}^L_{k-1})\mathcal{D}(\bar{C}^L_{k-1})^T, R_{k-1}=\mathcal{D}(\bar{C}^R_{k-1})\mathcal{D}(\bar{C}^R_{k-1})^T,\\ &L_k=\beta L_{k-1}+(1-\beta)G_kG_k^T, R_k=\beta R_{k-1}+(1-\beta)G_k^TG_k,\\ &C^L_k=\texttt{Cholesky}(L_k+\epsilon I), C^R_k=\texttt{Cholesky}(R_k+\epsilon I), \end{aligned} \tag{7}$$

where $\texttt{Cholesky}(L_k+\epsilon I)$ computes a lower triangular matrix C^L_k such that $C^L_k {C^L_k}^T = L_k + \epsilon I$. The small term ϵI is added for numerical stability, with ϵ as small constant. Once C^L_k and C^R_k are computed, they are quantized as:

$$\bar{C}^L_k = \mathcal{Q}(C^L_k), \qquad \bar{C}^R_k = \mathcal{Q}(C^R_k). \tag{8}$$

Accordingly, we can only store two quantized lower triangular matrices $\bar{C}^L_k$ and $\bar{C}^R_k$. Here we quantize the off-diagonal part of $\bar{C}^L_k$ and $\bar{C}^R_k$ into 4-bit precision while retaining the diagonal elements for 32-bit. This approach is used because off-diagonal elements have less impact on numerical stability, allowing reduced precision with minimal accuracy loss. In contrast, diagonal elements are crucial for overall stability and accuracy, so keeping them in 32-bit helps prevent error accumulation in the factorization.

Now we discuss two advantages of CQ. Firstly, Cholesky factors are lower triangular matrices, requiring nearly half memory compared to storing full preconditioners in Sec. 4.1 or full orthogonal matrices in [51], reducing peak GPU memory. Secondly, the preconditioner L_k recovered from $L_k = \mathcal{D}(\bar{C}^L_k)\mathcal{D}(\bar{C}^L_k)^T$ remains symmetric and positive definite (PD), better preserving spectral properties. Consequently, its inverse $1/4$-th root more closely approximates the original 32-bit preconditioner. To quantify this preservation, we consider the Frobenius norm relative error (NRE) and angle error (AE) between matrices [51], given by

$$\begin{aligned} \text{NRE} &= \|A^{-1/4} - g(A)^{-1/4}\|_F / \|A^{-1/4}\|_F, \\ \text{AE} &= \arccos\left(\frac{\langle A^{-1/4}, g(A)^{-1/4}\rangle}{\|A^{-1/4}\|_F \|g(A)^{-1/4}\|_F}\right), \end{aligned} \tag{9}$$

where g represents the combined effect of quantization and dequantization. We evaluate these metrics using both synthetic PD matrices and preconditioners from 32-bit Shampoo training of VGG-19 on CIFAR-100. As shown in Tab. 1, CQ significantly reduces both NRE and AE, demonstrating its effectiveness in preserving spectral properties. See Appendix C.2 for further details.

Table 1. NRE and AE on synthetic and real preconditioners for vanilla quantization (VQ) and Cholesky quantization (CQ).

Preconditioner	VQ		CQ	
	NRE	AE	NRE	AE
Synthetic	46.141	27.187	9.188	9.204
Epoch 50	29.041	19.353	5.367	5.366
Epoch 100	25.712	18.505	4.852	4.853
Epoch 150	25.351	19.317	4.788	4.788
Epoch 200	34.908	20.795	6.152	6.154

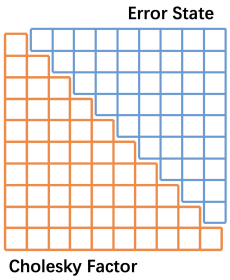


Figure 2. Efficient storage for Cholesky factor and error state.

Finally, we analyze the computational cost of CQ. In practice, parameter per layer is divided into 1200×1200 blocks before preconditioning (Appendix C.3), keeping the per-block CQ cost at $\mathcal{O}(n_p^3)$ with $n_p \leq 1200$. As shown in Tabs. 5 and 6, CQ introduces minimal overhead.

4.3. Compensated Cholesky Quantization

To mitigate the information loss from quantization, we introduce error feedback (EF) for Cholesky factors. Error feedback was original proposed to alleviate the information loss caused by gradient compression for communication in distributed training setting [40, 42]. The key idea is to compensate for compression errors by adding them back into the gradients before compression in the next step. Practical adaptations of EF has also been explored in [46, 57] to combine EF with adaptive gradient methods.

Different from previous work, our focus in this work is the compression of preconditioners of preconditioned gradient methods, and therefore our error feedback is conducted on the preconditioners. At each iteration, an additional low-precision (4-bit) error state, denoted as $\bar{E}^L_k$, is maintained to capture quantization error for the Cholesky factor $\bar{C}^L_k$. This error state is then used in the next iteration to enhance precision by compensating for potential quantization errors.

Specifically, at iteration k, we first compute the Cholesky factors C^L_k and C^R_k following the standard steps in Eq. (7). Before quantizing, we apply error compensation as follows:

$$\begin{aligned} E^L_{k-1} &= \mathcal{D}(\bar{E}^L_{k-1}), \qquad \bar{C}^L_k = \mathcal{Q}(C^L_k + E^L_{k-1}), \\ E^R_{k-1} &= \mathcal{D}(\bar{E}^R_{k-1}), \qquad \bar{C}^R_k = \mathcal{Q}(C^R_k + E^R_{k-1}). \end{aligned} \tag{10}$$

Next, we update the error states $\bar{E}^L_k$ and $\bar{E}^R_k$ using an exponential moving average to improve stability:

$$\begin{aligned} E^L_k &= \beta_e E^L_{k-1} + (1-\beta_e)(C^L_k + E^L_{k-1} - \mathcal{D}(\bar{C}^L_k)), \\ E^R_k &= \beta_e E^R_{k-1} + (1-\beta_e)(C^R_k + E^R_{k-1} - \mathcal{D}(\bar{C}^R_k)), \end{aligned} \tag{11}$$

where β_e is the momentum parameter. Since the Cholesky factors C^L_k and C^R_k are lower triangular and quantization excludes diagonal elements, the error states E^L_k and E^R_k are also triangular with zero diagonals. This enables efficient storage, as each error state can be stored as the upper triangular part as illustrated in Fig. 2, incurring no additional memory overhead compared to vanilla 4-bit Shampoo.

Finally, we can compute the inverse $1/4$-th root of the

Algorithm 1 4-bit Shampoo via Compensated Cholesky Quantization

Input: initial weight $W_0 \in \mathbb{R}^{m\times n}$, initial Cholesky factors $\bar{C}_0^L = \sqrt{\epsilon} I_m$, $\bar{C}_0^R = \sqrt{\epsilon} I_n$, quantization error states $\bar{E}_0^L = \mathbf{0}$, $\bar{E}_0^R = \mathbf{0}$, initial preconditioners $\hat{L}_0 = I_m$, $\hat{R}_0 = I_n$. Total training iterations T, interval of updating preconditioners T_1 and T_2, momentum parameter $\beta, \beta_e \in (0,1)$. First-order optimizer $\mathcal{F}$ with initial optimizer state s_0.
Output: final weight W_T.

for $k = 1, 2, \ldots, T$ **do**
  Compute gradient $G_k = \nabla \mathcal{L}_k(W_k)$
  if $k \% T_1 \equiv 0$ **then**
    Update Cholesky factors according to Eq. (7)
    Conduct error compensation following Eq. (10)
    Update quantization error states as Eq. (11)
  else
    $\bar{C}_k^L = \bar{C}_{k-1}^L$, $\bar{C}_k^R = \bar{C}_{k-1}^R$
  end if
  if $k \% T_2 \equiv 0$ **then**
    Compute inverse $1/4$-th root of the preconditioners following Eq. (12)
  else
    $\hat{L}_k = \hat{L}_{k-1}$, $\hat{R}_k = \hat{R}_{k-1}$
  end if
  $\hat{G}_k = \mathcal{D}(\hat{L}_k) G_k \mathcal{D}(\hat{R}_k)$
  $W_k, s_k = \mathcal{F}(W_{k-1}, s_{k-1}, \hat{G}_k)$
end for

preconditioners with stored Cholesky factors via

$$\begin{aligned} \hat{L}_k &= \mathcal{Q}((\mathcal{D}(\bar{C}_k^L)\mathcal{D}(\bar{C}_k^L)^T + \lambda_{\max}^L \epsilon I_m)^{-1/4}), \\ \hat{R}_k &= \mathcal{Q}((\mathcal{D}(\bar{C}_k^R)\mathcal{D}(\bar{C}_k^R)^T + \lambda_{\max}^R \epsilon I_n)^{-1/4}). \end{aligned} \tag{12}$$

Next, with SGD as the base optimizer, the model parameters are updated with the preconditioned gradient:

$$W_{k+1} = W_k - \eta_k \mathcal{D}(\hat{L}_k) G_k \mathcal{D}(\hat{R}_k), \tag{13}$$

where η_k is the learning rate for iteration k that is often scaled by $\|G_k\|_F / \|\hat{G}_k\|_F$ according to the grafting trick [1]. The preconditioned stochastic gradient $\mathcal{D}(\hat{L}_k) G_k \mathcal{D}(\hat{R}_k)$ can also be fed into another first-order optimizer $\mathcal{F}$, such as Adam, for model updates. Accordingly, we have arrived at our compensated Cholesky quantization based Shampoo summarized in Algorithm 1.

5. Theoretical Analysis

Here we provide theoretical analysis of Algorithm 1 with SGD base optimizer as an example. We first define

$$\begin{aligned} x_k &:= \mathrm{Vec}(W_k),\ g_k := \mathrm{Vec}(G_k), \\ H_k &:= \mathcal{D}(\hat{R}_k) \otimes \mathcal{D}(\hat{L}_k), \end{aligned} \tag{14}$$

where Vec reshapes the matrix into a vector by concatenating the columns of the matrix. Then, we rewrite Shampoo with SGD as base optimizer in Eq. (13) into an equivalent vectorization formulation:

$$x_{k+1} = x_k - \eta_k H_k g_k. \tag{15}$$

See this equivalent derivation in Appendix B. In the following, we analyze Shampoo with SGD as base optimizer in Eq. (15) under different situations.

5.1. Smooth Nonconvex Training Loss

Here we analyze the smooth nonconvex f, which is defined according to loss function Eq. (1) as

$$f(x) := F(W), \tag{16}$$

where $x = \mathrm{Vec}(W)$ is the vectorized model parameter. To this end, we introduce the necessary assumptions.

Assumption 5.1. ***a)*** *Assume the training loss f is L-Lipschitz smooth, i.e., $\|\nabla f(x) - \nabla f(y)\|_2 \le L \|x - y\|_2$.* ***b)*** *Suppose the stochastic gradient g_k is unbiased and its variance can be bounded: $\mathbb{E}[g_k] = \nabla f(x_k)$ and $\mathbb{E}[\|g_k - \nabla f(x_k)\|_2^2] \le \sigma^2 (1 + \|\nabla f(x_k)\|_2^2)$.* ***c)*** *Assume the preconditioner H_k has bounded eigenvalues, i.e., $\sup_k \lambda_{\max}(H_k) \le \lambda_{H,\max} < \infty$ and $\inf_k \lambda_{\min}(H_k) \ge \lambda_{H,\min} > 0$.*

For Assumptions 5.1a) and b), these conditions are standard for stochastic first-order methods (in fact, Assumption 5.1b) is even milder than the commonly assumed condition $\mathbb{E}[\|g_k - \nabla f(x_k)\|_2^2] \le \sigma^2$). Assumption 5.1c) requires the preconditioner H_k to be upper bounded and positive definite, which is guaranteed by the implementation of the Schur–Newton method, the regularization step in Eq. (7), and Proposition 5.1. Specifically, (i) the upper bound follows from Eq. (7) where an ϵI regularization is added to ensure a lower bound on C_k^L; then, applying the $-\frac{1}{4}$ exponent yields an upper bound while the operator $\mathcal{DQ}$ keeps the quantization error bounded. (ii) The strict positive definiteness (i.e., the lower bound) is ensured by the Gershgorin Circle Theorem and the Schur–Newton method. In particular, if S_k denotes the inner matrix in Eq. (12) such that $\mathcal{D}(\hat{L}_k) = \mathcal{DQ}\big(S_k^{-1/4}\big)$, then the Schur–Newton method (applied for a limited number of steps) yields a diagonally dominant matrix Z_k approximating $S_k^{-1/4}$; writing $\mathcal{D}(\hat{L}_k) = Z_k + E_k^Z$ with E_k^Z denoting the quantization error, the Gershgorin Circle Theorem guarantees that if Z_k is strictly diagonally dominant and $\|E_k^Z\|$ is sufficiently small, then $Z_k + E_k^Z$ remains strictly positive definite, as further supported by Proposition 5.1. Empirical evidence in Fig. 3 also demonstrates that the eigenvalues of the dequantized preconditioners $\mathcal{D}(\hat{L}_k)$ and $\mathcal{D}(\hat{R}_k)$ remain positive throughout training, further validating Assumption 5.1c).

Proposition 5.1. *For the 4-bit Shampoo in Algorithm 1, let $M_k := (\mathcal{D}(\bar{C}_k^L)\mathcal{D}(\bar{C}_k^L)^T + \lambda_{\max}^L \epsilon I_m)^{-1/4}$, if $\|M_k\|_{\mathrm{off},\max} \le C_B$, then its preconditioners hold that*

$$\mathcal{D}(\hat{L}_k) \preceq M_k + C_B n_k 2^{-b} I,$$

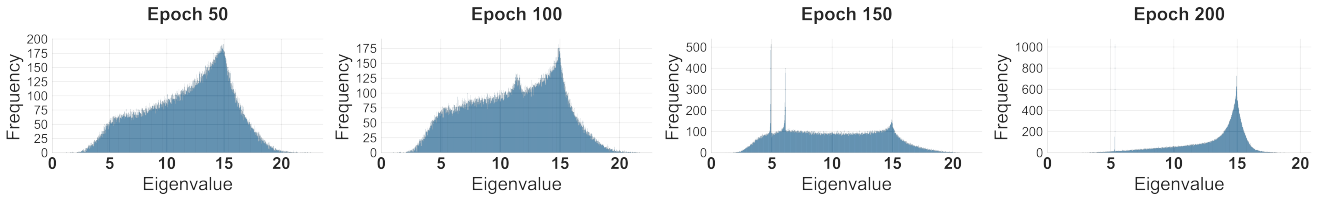


Figure 3. Eigenvalue frequency of the dequantized preconditioners $\mathcal{D}(\hat{L})$ and $\mathcal{D}(\hat{R})$ of VGG-19 on CIFAR-100 at 50, 100, 150, and 200 training epochs, all eigenvalues are greater than 0.

where $\|\cdot\|_{\text{off,max}}$ *is the maximal absolute value of all off-diagonal entries and* n_k *is the number of rows in* W_k*. Furthermore, if for every row index* i *it holds that* $|[M_k]_{ii}| > \left(1 + \frac{2}{2^b-1}\right) \sum_{j \neq i} |[M_k]_{ij}|$*, then* $\mathcal{D}(\hat{L}_k) \succ 0$*.*

This proposition shows that the sequence $\{\mathcal{D}(\hat{L}_k)\}$ can be bounded above and below. Now we are ready to derive the convergence, and state the main results below.

Theorem 5.1. *Suppose Assumption 5.1 holds. Let* $\eta_k = \frac{c}{\sqrt{T+1}}$ *with* $c \in \left(0, \frac{\lambda_{H,\min}}{2L(1+\sigma^2)\lambda^2_{H,\max}}\right)$*, then we have*

$$\mathbb{E}\left[\|\nabla f(\bar{x}_T)\|_2^2\right] \leq \frac{2(f(x_0) - \bar{f} + c^2 L\sigma^2 \lambda^2_{H,\max})}{c\lambda_{H,\min}\sqrt{T+1}},$$

where $\bar{x}_T$ *is randomly selected from* $\{x_0, x_1, ..., x_T\}$ *and* $\bar{f} := \min_{x \in \mathbb{R}^d} f(x)$*.*

See its proof in Appendix B. Theorem 5.1 shows that for smooth nonconvex training loss, our 4-bit Shampoo with SGD as the base optimizer can converge at the rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$. This convergence rate is optimal as shown in [8], indicating that our 4-bit Shampoo maintains the optimal convergence rate despite the potential information loss introduced by quantization.

5.2. Nonsmooth Nonconvex Training Loss

In this subsection, we analyze the nonsmooth nonconvex training loss function, particularly in cases where the activation function is nonsmooth, such as the ReLU in ResNet [23]. The iterative scheme can be written as:

$$x_{k+1} = x_k - \eta_k H_k(d_k + \xi_k),$$

where $d_k \in \partial f(x_k)$, ∂f denotes the subgradient of f, and $\{\xi_k\}$ is the sequence of the random noise in the subgradient. Relevant concepts are provided in Appendix B.2. Given a process $\{\xi_i\}_{i=0}^{\infty}$, let $\mathcal{F}_k$ denote the history up to iteration k. To this end, we introduce the necessary assumptions.

Assumption 5.2. ***a)*** *The function* f *is* ℓ*-Lipschitz continuous. Additionally,* f *is a Whitney stratifiable function.*
b) *The noise in the subgradient is unbiased and has bounded variance*

$$\mathbb{E}[\xi_k | \mathcal{F}_{k-1}] = 0, \quad \mathbb{E}[\|\xi_k\|_2^2 | \mathcal{F}_{k-1}] \leq \sigma^2,$$

c) *For any convergent subsequence* $x_{k_j} \to \bar{x}$*, we have* $\lim_{N\to\infty} \frac{1}{N} \sum_{j=1}^{N} H_{k_j} = \bar{H}$ *for some positive definite matrix* $\bar{H}$*. Additionally,* $\sup_{k \geq 0} \lambda_{\max}(H_k) \leq M$*.*

The class of Lipschitz continuous functions is broad and includes pathological cases where subgradient flows fail to converge to stationary points [12]. To address this, we focus on Whitney stratifiable functions, which generalize most practical cases, including loss functions in neural networks with nonsmooth activations like ReLU [5, 13]. Assumption 5.2c) requires only Cesàro summability of $\{H_k\}$, a mild condition crucial for handling non-time-homogeneity.

Theorem 5.2. *Suppose that Assumption 5.2 holds and the sequence* $\{x_k\}$ *remains within a compact set. If the learning rate satisfies* $\sum_{k=1}^{\infty} \eta_k = \infty$ *and* $\sum_{k=1}^{\infty} \eta_k^2 < \infty$*, then*

$$\lim_{k\to\infty} \operatorname{dist}(x_k, \Omega) = 0,$$

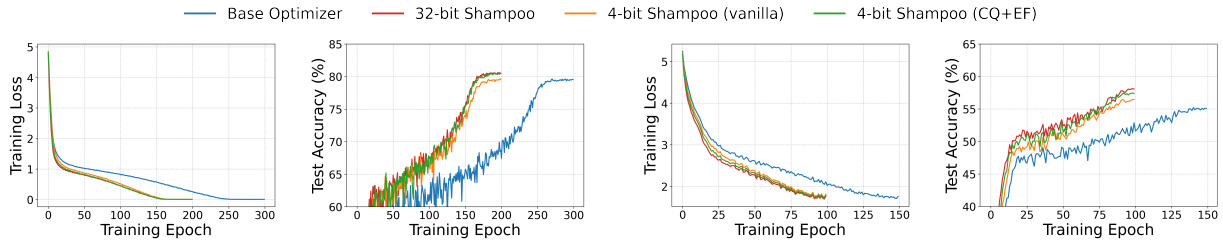
where $\Omega := \{x : 0 \in \partial f(x)\}$ *is the set of stationary points.*

For a stratifiable function, the result of convergence to the stationary point set is tight. There are no complexity results due to the challenges posed by its complex nonconvexity and nonsmoothness [5, 13]. This result ensures the convergence of our proposed algorithm on nonsmooth training losses, including those arising in deep neural networks such as ReLU-based architectures.

6. Experiments

In this section, we evaluate our 4-bit Shampoo Algorithm 1 on classical image classification and large language model (LLM) pre-training. We compare its performance against vanilla 4-bit and 32-bit Shampoo when using SGD with momentum (SGDM) [45] or AdamW [35] as base optimizer, and the base optimizer itself. For all experiments, we report test accuracy and peak memory usage to assess both algorithmic performance and GPU memory overhead.

Training Setting. Following standard benchmarks for image classification [23, 31, 53], we train VGG-19 [44], ResNet-34 [23], Swin Transformer Tiny (Swin-Tiny) [34], and Vision Transformer Small (ViT-Small) [16] on CIFAR-100 [27] and Tiny-ImageNet [29], as well as ResNet-50 and ViT-Base on ImageNet [14]. For LLM pre-training, we follow [33, 64] to train LLaMA [48] on the C4 dataset [39] with varying model sizes. Training hyperparameters for Shampoo match those of the base optimizer, except that the base optimizer is trained for additional epochs in image classification to achieve comparable performance. All experiments are conducted on a single NVIDIA A100 GPU (80GB). Further details are provided in Appendix C.3.



(a) SGDM as base optimizer.

(b) AdamW as base optimizer.

Figure 4. Comparison of training loss and test accuracy (%) for training ResNet-34 on CIFAR-100 and ViT-Small on Tiny-ImageNet. The left figure shows ResNet-34 results, and the right figure shows ViT-Small results.

Table 2. Test accuracy (%) and peak memory (MB) of vanilla 4-bit Shampoo with off-diagonal and original block-wise quantization for VGG-19 on CIFAR-100 and Swin-Tiny on Tiny-ImageNet.

Method	VGG-19		Swin-Tiny	
	Accuracy	Memory	Accuracy	Memory
Original	74.20	661.7	60.83	1126.2
Off-Diagonal	74.36	662.2	61.28	1126.9

Table 3. Test accuracy (%) and peak memory (MB) on CIFAR-100. Here VQ denotes vanilla quantization, CQ denotes Cholesky quantization, and EF denotes error feedback.

Model	Optimizer	Accuracy	Memory
VGG-19	SGDM	74.43	597.3
	SGDM + 32-bit Shampoo	75.02	1065.2
	SGDM + 4-bit Shampoo (VQ)	74.36	662.2
	SGDM + 4-bit Shampoo (CQ)	74.99	646.0
	SGDM + 4-bit Shampoo (CQ+EF)	75.21	662.2
ResNet-34	SGDM	79.12	1254.7
	SGDM + 32-bit Shampoo	80.69	1882.6
	SGDM + 4-bit Shampoo (VQ)	79.45	1341.0
	SGDM + 4-bit Shampoo (CQ)	80.27	1319.5
	SGDM + 4-bit Shampoo (CQ+EF)	80.52	1341.0
Swin-Tiny	AdamW	78.28	1095.3
	AdamW + 32-bit Shampoo	79.84	1248.6
	AdamW + 4-bit Shampoo (VQ)	78.33	1116.8
	AdamW + 4-bit Shampoo (CQ)	79.29	1111.5
	AdamW + 4-bit Shampoo (CQ+EF)	79.45	1116.8
ViT-Small	AdamW	73.00	2930.0
	AdamW + 32-bit Shampoo	77.95	3448.9
	AdamW + 4-bit Shampoo (VQ)	74.56	3001.7
	AdamW + 4-bit Shampoo (CQ)	77.34	2983.7
	AdamW + 4-bit Shampoo (CQ+EF)	77.74	3001.7

6.1. Test Performance

To ensure a fair comparison between vanilla 4-bit Shampoo and our method, we apply off-diagonal 4-bit block-wise quantization to Shampoo's preconditioners while retaining diagonal elements in 32-bit, defining this as vanilla 4-bit Shampoo. As shown in Tab. 2, off-diagonal quantization only slightly increases peak memory but improves test performance. Thus, we adopt off-diagonal quantization for vanilla 4-bit Shampoo in all experiments.

As shown in Tab. 3, 4-bit Shampoo with Cholesky quantization consistently outperforms vanilla 4-bit Shampoo. For instance, with SGDM as the base optimizer for ResNet-34 on CIFAR-100, it achieves 80.27% test accuracy versus 79.45% for vanilla 4-bit Shampoo. Similarly, with AdamW for ViT-Small on CIFAR-100, it reaches 77.34% compared to 74.56%. This improvement stems from Cholesky quantization's ability to recover preconditioners from Cholesky factors, better preserving the spectral properties of 32-bit Shampoo preconditioners (Sec. 4.2).

Table 4. Test accuracy (%) and peak memory (MB) on Tiny-ImageNet. Here VQ denotes vanilla quantization, CQ denotes Cholesky quantization, and EF denotes error feedback.

Model	Optimizer	Accuracy	Memory
VGG-19	SGDM	62.19	1632.8
	SGDM + 32-bit Shampoo	63.36	2102.5
	SGDM + 4-bit Shampoo (VQ)	62.34	1697.8
	SGDM + 4-bit Shampoo (CQ+EF)	63.51	1697.8
ResNet-34	SGDM	68.27	4221.3
	SGDM + 32-bit Shampoo	69.11	4846.0
	SGDM + 4-bit Shampoo (VQ)	68.43	4307.7
	SGDM + 4-bit Shampoo (CQ+EF)	68.88	4307.7
Swin-Tiny	AdamW	60.74	1105.5
	AdamW + 32-bit Shampoo	62.73	1256.8
	AdamW + 4-bit Shampoo (VQ)	61.28	1126.9
	AdamW + 4-bit Shampoo (CQ+EF)	62.81	1126.9
ViT-Small	AdamW	55.21	2944.2
	AdamW + 32-bit Shampoo	58.11	3468.1
	AdamW + 4-bit Shampoo (VQ)	56.47	3016.0
	AdamW + 4-bit Shampoo (CQ+EF)	57.51	3016.0

Moreover, experimental results in Tab. 3 validate the effectiveness of the error compensation strategy for Cholesky factors introduced in Sec. 4.3. With SGDM as the base optimizer for ResNet-34 on CIFAR-100, 4-bit Shampoo with compensated Cholesky decomposition improves test accuracy by 0.25% over 4-bit Cholesky quantization. Similarly, with AdamW for ViT-Small on Tiny-ImageNet, it achieves a 0.4% improvement. This consistent gain stems from EF, which retains and integrates quantization errors from previous steps into the updated Cholesky factors before each quantization, iteratively minimizing quantization errors.

Experimental results on larger image classification datasets (Tabs. 4 and 5) further validate the superiority of our 4-bit Shampoo with compensated Cholesky quantization. On Tiny-ImageNet, it consistently improves test accuracy by over 0.45% compared to vanilla 4-bit Shampoo, whether using SGDM or AdamW as the base optimizer. For ResNet-50 and ViT-Base on ImageNet, it increases test accuracy by 0.27% and 0.73%, respectively, achieving performance close to the original 32-bit Shampoo.

For LLM pre-training experiments (Tab. 6), our 4-bit

Table 5. Comparison of test accuracy (%), wall-clock time (min), and peak memory (MB) on the ImageNet dataset.

Model	Optimizer	Accuracy	Time	Memory
ResNet-50	Base	77.56	2106.4	11356.2
	32-bit	78.06	1841.1	11986.4
	4-bit (VQ)	77.73	1882.8	11445.2
	4-bit (ours)	78.00	1899.4	11445.2
ViT-Base	Base	72.59	1741.6	11839.7
	32-bit	75.47	1392.4	13319.1
	4-bit (VQ)	72.28	1406.2	12052.3
	4-bit (ours)	75.01	1409.6	12052.3

Table 6. Comparison of test perplexity (PPL, lower is better), update time (min), and peak memory (GB) on the C4 dataset.

Model	Optimizer	PPL	Time	Memory
LLaMA-130M	Base	27.32	162.9	45.9
	32-bit	26.93	169.1	48.2
	4-bit (VQ)	28.08	170.5	46.2
	4-bit (ours)	26.98	178.9	46.2
LLaMA-350M	Base	24.29	431.7	52.9
	32-bit	24.07	443.8	58.8
	4-bit (VQ)	25.14	445.3	53.7
	4-bit (ours)	23.99	456.2	53.7
LLaMA-1B	Base	48.39	403.7	59.0
	32-bit	*Out of GPU Memory*		
	4-bit (VQ)	48.53	411.4	61.9
	4-bit (ours)	46.31	425.0	61.9

Shampoo with compensated Cholesky quantization consistently achieves lower test perplexity than vanilla 4-bit Shampoo and the base optimizer. Its performance closely matches 32-bit Shampoo, provided the 32-bit version fits within GPU memory. These results demonstrate the effectiveness of our quantization strategy in preserving test performance for large-scale neural network training.

6.2. Memory and Computational Efficiency

For GPU memory usage, Tabs. 3 to 6 show that 4-bit quantization significantly reduces the peak memory consumption of 32-bit Shampoo. For instance, with SGDM as the base optimizer for ResNet-34 on CIFAR-100, 4-bit Shampoo lowers peak memory by over 540MB, reducing usage by more than 28%. Similarly, with AdamW for LLaMA-350M on C4, it reduces peak memory by 5.1GB. Moreover, when training LLaMA-1B, 32-bit Shampoo exceeds GPU memory limits on an A100 (80GB), whereas our 4-bit Shampoo runs efficiently with strong test performance.

Additionally, as shown in Tab. 3, 4-bit Cholesky quantization further reduces peak GPU memory usage compared to vanilla quantization. For example, when training ResNet-34 on CIFAR-100 with SGDM, it reduces peak memory by 21.5MB, accounting for 25% of the 86.3MB overhead introduced by vanilla 4-bit Shampoo's preconditioners. This efficiency arises from storing only the lower triangular Cholesky factors $\bar{C}_k^L, \bar{C}_k^R$, which require half the memory of full matrices L_k, R_k (Sec. 4.2). Thus, 4-bit Shampoo with Cholesky quantization achieves additional memory savings over vanilla 4-bit Shampoo. See Appendix C.4 for further details.

For computational efficiency, Tabs. 5 and 6 show that the overhead introduced by compensated Cholesky quantization over vanilla 4-bit quantization is minimal. When training ResNet-50 and ViT-Base on ImageNet, the additional computation time is under 20 minutes, accounting for less than 1% of the total training time. For LLaMA training on the C4 dataset, it adds less than 15 minutes, contributing to under 5% of the total training time.

6.3. Ablation Study

Comparison with [51]. We compare our 4-bit Shampoo with [51] across a variety of tasks. As shown in Tab. 7, our method consistently outperforms [51].

Table 7. Test accuracy (%) and PPL of [51] and our algorithm.

Model Metric	VGG-19 Acc. (↑)	Swin-Tiny Acc. (↑)	LLaMA-350M PPL (↓)	LLaMA-1B PPL (↓)
4-bit ([51])	74.71	79.04	24.39	48.32
4-bit (ours)	75.21	79.45	23.99	46.31

Effects of β and β_e. Following modern Shampoo algorithms [3, 43], we maintain an exponential moving average of Cholesky factors and error states. Tab. 8 shows the robustness of our method to momentum coefficients β, β_e.

Table 8. Test accuracy (%) for ResNet-34 on CIFAR-100.

β, β_e	0.6	0.7	0.8	0.9	0.95	0.98
Accuracy	80.40	80.36	80.44	80.47	80.52	80.30

More Optimizers. We further evaluate RMSprop as the base optimizer. As shown in Tab. 9, our 4-bit Shampoo consistently outperforms vanilla 4-bit Shampoo while reducing memory usage compared to the 32-bit version.

Table 9. Test accuracy (%) and peak memory (MB) for Swin-Tiny on CIFAR-100 with RMSprop as the base optimizer.

Optimizer	Accuracy	Memory
RMSprop	74.35	1066.1
RMSprop+32-bit Shampoo	75.67	1219.5
RMSprop+4-bit Shampoo (VQ)	74.82	1087.5
RMSprop+4-bit Shampoo (ours)	75.31	1087.5

7. Conclusion

We propose 4-bit Shampoo, a memory-efficient preconditioned gradient method that significantly reduces GPU usage while maintaining performance close to 32-bit Shampoo. By quantizing only the lower-triangular Cholesky factors to 4 bits, our approach halves memory cost while preserving spectral properties. An error feedback mechanism further corrects quantization errors at each step. We prove convergence in nonconvex settings and demonstrate strong results on image classification and LLM pre-training.

Limitations. (a) Our CQ and EF are currently tested only with Shampoo, though they are applicable to other preconditioned methods, which we leave for future work. (b) Due to limited compute, we focus on image classification and LLM pre-training, leaving domains like detection, video generation, and large-scale fine-tuning for future study.

Acknowledgments

This research is supported by the Ministry of Education, Singapore, under its AcRF Tier 2 Funding (Proposal ID: T2EP20224-0048) and its AcRF Tier 1 grant (project ID: 23-SIS-SMU-063). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

[1] Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020. 5

[2] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017. 2

[3] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020. 1, 2, 8

[4] Michel Benaïm, Josef Hofbauer, and Sylvain Sorin. Stochastic approximations and differential inclusions. *SIAM J. Control and Optimization*, 44(1):328–348, 2005. 3

[5] Jérôme Bolte and Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, 188:19–51, 2021. 6, 3

[6] Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009. 3

[7] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020. 1

[8] Yair Carmon, John C Duchi, Oliver Hinder, and Aaron Sidford. Lower bounds for finding stationary points ii: first-order methods. *Mathematical Programming*, 185(1):315–355, 2021. 6

[9] Frank H Clarke. *Optimization and nonsmooth analysis*. SIAM, 1990. 3

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 5

[11] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 5

[12] Aris Daniilidis and Dmitriy Drusvyatskiy. Pathological subgradient dynamics. *SIAM Journal on Optimization*, 30(2): 1327–1338, 2020. 6, 3

[13] Damek Davis, Dmitriy Drusvyatskiy, Sham Kakade, and Jason D Lee. Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, 20 (1):119–154, 2020. 6, 3, 4

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[15] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021. 2, 3

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 3, 6

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011. 1, 2

[18] John C Duchi and Feng Ruan. Stochastic methods for composite and weakly convex optimization problems. *SIAM J. Optimization*, 28(4):3229–3259, 2018. 3

[19] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022. 3

[20] Donald Goldfarb, Yi Ren, and Achraf Bahamou. Practical quasi-newton methods for training deep neural networks. *Advances in Neural Information Processing Systems*, 33: 2386–2396, 2020. 1, 2

[21] Chun-Hua Guo and Nicholas J Higham. A schur–newton method for the matrix p th root and its inverse. *SIAM Journal on Matrix Analysis and Applications*, 28(3):788–804, 2006. 3

[22] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018. 1, 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 2, 6, 5

[24] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 1

[25] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[26] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1, 2

[27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3, 6

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural net-

works. *Advances in neural information processing systems*, 25, 2012. 5

[29] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 6

[30] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020. 1

[31] Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. Vision transformer for small-size datasets. *arXiv preprint arXiv:2112.13492*, 2021. 6, 5

[32] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 3

[33] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates. *arXiv preprint arXiv:2307.05695*, 2023. 6, 5

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 6, 5

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 1, 6

[36] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International Conference on Machine Learning*, pages 2408–2417. PMLR, 2015. 1, 2

[37] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. *arXiv preprint arXiv:2406.17748*, 2024. 2

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 1

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. 6

[40] Peter Richtárik, Igor Sokolov, and Ilyas Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems*, 34:4384–4396, 2021. 2, 4

[41] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951. 1

[42] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech*, pages 1058–1062. Singapore, 2014. 2, 4

[43] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023. 1, 2, 8

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[45] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147. PMLR, 2013. 1, 6

[46] Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pages 10118–10129. PMLR, 2021. 2, 4

[47] Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012. 2

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1, 6

[49] Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[50] Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024. 2

[51] Sike Wang, Jia Li, Pan Zhou, and Hua Huang. 4-bit shampoo for memory-efficient network training. *arXiv preprint arXiv:2405.18144*, 2024. 2, 3, 4, 8

[52] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017. 2

[53] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 6

[54] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023. 1

[55] Nachuan Xiao, Xiaoyin Hu, Xin Liu, and Kim-Chuan Toh. Adam-family methods for nonsmooth optimization with convergence guarantees. *Journal of Machine Learning Research*, 25(48):1–53, 2024. 3

[56] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120 (14):145301, 2018. 1

[57] Xingyu Xie, Zhijie Lin, Kim-Chuan Toh, and Pan Zhou. Loco: Low-bit communication adaptor for large-scale model training. *arXiv preprint arXiv:2407.04480*, 2024. 2, 4

[58] Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1, 5

[59] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183, 2022. 3

[60] Hongwei Yong, Ying Sun, and Lei Zhang. A general regret bound of preconditioned gradient method for dnn training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7866–7875, 2023. 1, 2, 3

[61] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 5

[62] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 5

[63] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020. 1

[64] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024. 6, 5

[65] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008, 2020. 5