JointSQ: Joint Sparsification-Quantization for Distributed Learning

Weiying Xie [1,*†], Haowei Li [1,*], Jitao Ma [1], Yunsong Li [1], Jie Lei [2], Donglai Liu [1], Leyuan Fang [3]
[1] State Key Laboratory of Integrated Services Networks, Xidian University,
[2] University of Technology Sydney, [3] Hunan University
wyxie@xidian.edu.cn {23011210779,21011210271,dlliu_2}@stu.xidian.edu.cn
jie.lei@uts.edu.au ysli@mail.xidian.edu.cn fangleyuan@gmail.com

Abstract

Gradient sparsification and quantization offer a promising prospect to alleviate the communication overhead problem in distributed learning. However, direct combination of the two results in suboptimal solutions, due to the fact that sparsification and quantization haven't been learned together. In this paper, we propose Joint Sparsification-Quantization (JointSQ) inspired by the discovery that sparsification can be treated as 0-bit quantization, regardless of architectures. Specifically, we mathematically formulate JointSQ as a mixed-precision quantization problem, expanding the solution space. It can be solved by the designed MCKP-Greedy algorithm. Theoretical analysis demonstrates the minimal compression noise of JointSQ, and extensive experiments on various network architectures, including CNN, RNN, and Transformer, also validate this point. Under the introduction of computation overhead consistent with or even lower than previous methods, JointSQ achieves a compression ratio of 1000× on different models while maintaining near-lossless accuracy and brings 1.4× to 2.9× speedup over existing methods.

1. Introduction

The assistance of multi-clients distributed learning to gain deep insights onto the huge data closest to the data source has further fuelled embeded applications and edge devices [8, 9, 24, 40]. In such tiny artificial intelligence system, a large number of distributed devices are typically connected via wireless and long-distance connection, so bandwidth is severely limited [20, 36].

So far, gradient compression has indeed paved way for reducing the amount of communication exchanged between workers. Sparsification [12, 21, 29, 35] and quantization [4, 6, 13, 14, 26] are two of the most mainstream gradient compression techniques for efficient distributed learning and communication. Combining the two is often done in practice [5, 23, 37]. A direct way is to perform a two-stage sequential scheme which we called Co-compression: first sparsifying with a sparsification ratio set manually [5] or adaptively [23, 37] and then quantizing the retained gradients with a uniform bit-width. Such a two-stage sequential scheme is inadequate due to the limited solution space it brings, thus resulting in suboptimal solutions. On the one hand, the two stages (sparsification and quantization) have not been learned together to maximize their respective strengths through cooperation. On the other hand, the two-stage sequential scheme varies greatly across different models, making architecture specific compressors non-generalizable to new architectures. Our subsequent experimental results also validate this point.

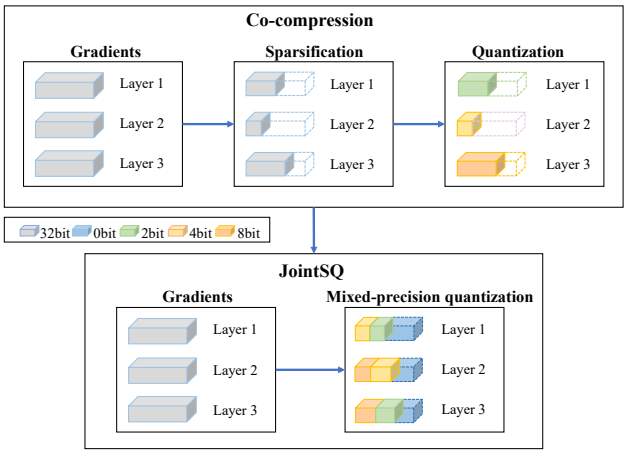


Figure 1. Existing Co-compression methods and our JointSQ framework. Existing Co-compression methods typically apply sparsification and quantization step by step. Our framework considers sparsification as 0-bit quantization and thus the two-stage process is transformed into a unified learning framework.

In this paper, we propose adaptive Joint Sparsification-Quantization (JointSQ) to address the suboptimal solution bottleneck for communication-efficient distributed learning.

*Equal contribution.
†Corresponding author.

The conceptual framework of our approach can be seen in Figure 1. The key idea is to treat sparsification as 0-bit quantization thus the sparsification can indeed be unified with quantization fundamentally. Specifically, JointSQ is first formulated as a mixed-bit precision quantization (*i.e.*, 0-bit, 2-bit, 4-bit and 8-bit) with sparsification to 0-bit for end-to-end Co-compression only once. Notably, to ensure adaptive bit-width assignment in such joint optimization space, a Multiple-Choice Knapsack Problem (MCKP) [3, 18, 38] is special designed per-layer and we have addressed this problem with the lowest computational cost. Theoretically, we prove that our JointSQ possesses the minimum compression noise and validate our theoretical analysis through extensive deep learning experiments, including image classification tasks on ImageNet [10], CIFAR-10 [19], and CIFAR-100 [19], as well as language modeling tasks on PTB [22]. We demonstrate that (i) JointSQ ahieves only a small loss of 1% in Top-1 accuracy under an extreme compression ratio as 1000×; (ii) JointSQ achieves no accuracy drop at 32× compression ratio; (iii) JointSQ achieves SOTA over various network architectures with 1.4× to 1.9× speedup compared to existing methods. Our contributions can be summarized as follows:

- We propose JointSQ, a novel adaptive joint sparsification-quantization framework to address the suboptimal solution bottleneck for communication-efficient distributed learning. To our best knowledge, our framework is the first work to treat sparsification as 0-bit quantization, and thus the two-stage indeed is transformed into a unified learning framework, *i.e.*, mixed-bit precision quantization.
- To adaptively find the optimal bit-width assignment, we heuristically model JointSQ as a MCKP that can be solved by the greedy search algorithm. We theoretically justify the superiority of JointSQ over the two-stage Co-compression.
- We empirically evaluate JointSQ on multiple deep learning tasks, including image classification tasks on ImageNet, CIFAR-10 and CIFAR-100, as well as language modeling tasks on PTB. JointSQ shows impressive improvement over previous methods while achieving the fastest training acceleration.

2. Related Work

Gradient Quantization. The main idea is to use lower precision of gradient, reducing the number of transmitted bits. Focusing on existing gradient quantization approaches, 1-bit SGD [26] and signSGD [6] quantized gradients to 1-bit, while TernGrad [36] quantized gradients to 2-bit. QSGD [4] proposed compression schemes with several bit-width levels (2, 4, and 8 bits). MemSGD [39] maintained accumulated errors in memory to achieve similar convergence rates as 32-bit SGD. Ef-signSGD [17] improved convergence by introducing error feedback in the next optimization step. Besides, some methods adaptively adjust the level of quantization during the training process. AQSGD [13] learned and adjusted the parameters for gradient compression in real-time to reduce the variance between gradients on individual worker nodes and accelerate training. Similarly, AdaQS [14] aimed to reduce the variance of quantized gradients by using a lower level of quantization in the early stages and a higher level of quantization in the later stages. However, gradient quantization only provides limited compression before accuracy degradation due to the requirement of transmitting at least one bit per entry. In contrast, sparsification can provide order-of-magnitude compression improvements.

Gradient Sparsification. The main idea is to communicate partial components of the local gradient. Pioneering work included Top-k [2, 5, 21] sparsifier which selected the largest k elements for communication. In the same period, Rand-k [30,35] sparsifier was introduced to randomly select k components. Most methods are based on the aforementioned two ideas to sparsify gradient. For example, Strom [31] and Aji [2] significantly compacted sub-gradients by considering only gradient elements whose absolute values exceed a threshold. DGC [21] employed several training tricks to achieve high compression ratio while maintaining performance. Song *et al.* [29] not only focused on gradients with larger absolute values but also gradually increased the sampling of small gradients as the sampling frequency increases. To maintain baseline's performance under orders of magnitude, sparsification usually leads to additional costs in terms of maintaining error correction and hyper-parameter tuning.

Combination of Sparsification and Quantization. The challenge in Co-compression methods lies in determining the optimal sparsification ratio and quantization levels for the model. Manual setting of these parameters often requires extensive empirical tuning based on prior experience [5, 16]. Therefore, adaptive methods have been developed to dynamically adjust the compression strategy [23, 37]. These approaches can effectively combine sparsification and quantization, however, lead to a suboptimal solution, because the sparsification and quantization are still two-stage and they have not been learned together to maximize their respective strengths through cooperation. Compared to the previous work, our framework achieves further performance improvements, while introducing minimal algorithmic complexity, by merging the two-stage operations into a single step. Moreover, our framework is compatible with all the aforementioned quantization schemes.

3. Joint Sparsification-Quantization

In this section, we firstly introduce a general formulation of Co-compressor and its unbiasedness and bounded variance. Secondly, we introduce the formula representa-

tion of JointSQ with its unbiasedness and bounded variance. We validate that JointSQ exhibits superior performance. Lastly, as our framework involves the allocation of mixed-precision quantization bit-width, we show that it can be formed as variant Knapsack problems and introduce efficient algorithms to solve them.

3.1. General Co-compression Formulation

Previous research has provided analysis for the Co-compressor with gradient amplification and we extend it to the general form of the Co-compressor.

Lemma 3.1 *(Unbiasness and Bounded Variance of Co-Compressor [7, 32, 37]): For gradient vector* $\mathbf{g} \in \mathbb{R}^d$ *, if the number of quantization bits is b, the sparsification size is k, then the compressed vector* $\hat{\mathbf{g}} = \mathcal{Q}_b\left[\mathcal{S}_k(\mathbf{g})\right]$ *satisfies:*

$$\mathbb{E}[\hat{\mathbf{g}}] = \mathbf{g}, \tag{1}$$

$$\mathbb{E}\left[\|\hat{\mathbf{g}}\|^2\right] \leq \|\mathbf{g}\|^2 + \frac{k}{4^b}\|\mathbf{g}\|^2, \tag{2}$$

where $\mathcal{Q}_b[\cdot]$ and $\mathcal{S}_k[\cdot]$ are the quantizer with b quantization bits and the sparsifier with k sparsification size, respectively [4, 30, 35]. Here d represents the length of the gradient tensor. Eq. (1) means that the compressed gradient $\hat{\mathbf{g}}$ is the unbiased estimate of $\mathbf{g}$. Eq. (2) provides the bounded variance of the Co-compressor. We focus on Eq. (2), that implies the noise added on the uncompressed gradient, defined as:

$$h \triangleq \frac{k}{4^b}. \tag{3}$$

3.2. JointSQ Formulation

JointSQ treats sparsification as 0-bit quantization and introduces mixed-precision quantization with n levels [33]. It also achieves unbiasedness and convergence, which can be reformulated as follows:

$$\mathbb{E}'[\hat{\mathbf{g}}] = \mathbf{g}, \tag{4}$$

$$\mathbb{E}'\left[\|\hat{\mathbf{g}}\|^2\right] \leq \|\mathbf{g}\|^2 + \sum_{i=1}^{n} \frac{k_i}{4^{b_i}} \|\mathbf{g}_i\|^2. \tag{5}$$

We demonstrate the proof in the appendix. Let $\mathbf{g}_i$ represent the subgradient quantized using the $i-th$ available bit-width b_i, and $\mathbf{g} = \sum_{i=1}^{n} \mathbf{g}_i$. For example, assuming the gradient vector is $\{0.1, 0.2, 0.3, 0.4\}$, it can be divided into subgradients as $\{0.1, 0.2\}$ and $\{0.3, 0.4\}$. For ease of analysis, we set the remaining positions of the subgradients to 0 to match the length of the original gradient vector. Here k_i denote the length of $\mathbf{g}_i$, and $\|\mathbf{g}\|$ is the l_2 norm of $\mathbf{g}$. Thus, the noise introduced on the uncompressed gradient can be defined as:

$$h' \triangleq \sum_{i=1}^{n} \frac{k_i}{4^{b_i}} \frac{\|\mathbf{g}_i\|^2}{\|\mathbf{g}\|^2}. \tag{6}$$

When $n = 1$, Eq. (5) is equivalent to Eq. (2), indicating that the general form of the Co-compressor is a special case of JointSQ. This implies that the solution space of the Co-compressor design is a subset of the solution space of JointSQ, and it holds that:

$$\min h' \leq \min h. \tag{7}$$

Assuming that the gradients are quantized by the Co-compressor to a uniform bit-width of b, we consider the scenario where there are n_1 gradient elements with respective bit width increments x_1, x_2, ..., x_{n_1}, and n_2 gradient elements with respective bit width decrements y_1, y_2, ..., y_{n_2}, such that $x_1+x_2+...+x_{n_1} = y_1+y_2+...+y_{n_2}$. According to Eq. (6), the variation of compressed noise is:

$$\Delta h' = \sum_{i=1}^{n_1} \left(\frac{1}{4^{b+x_i}} - \frac{1}{4^b}\right) \frac{|g_i|^2}{\|\mathbf{g}\|^2} + \sum_{j=1}^{n_2} \left(\frac{1}{4^{b-y_j}} - \frac{1}{4^b}\right) \frac{|g_j|^2}{\|\mathbf{g}\|^2}. \tag{8}$$

By solving the inequality $\Delta h' < 0$, we obtain the following result:

$$\sum_{i=1}^{n_1} \frac{4^{x_i} - 1}{4^{x_i}} |g_i|^2 > \sum_{j=1}^{n_2} \left(4^{y_j} - 1\right) |g_j|^2. \tag{9}$$

The detailed proof is provided in the appendix. This provides the fundamental condition for the faster convergence of JointSQ compared to the general Co-compressor. We observe that JointSQ expands the solution space of the Co-compressor and mitigates the issue of suboptimal solutions, primarily due to the introduction of lower compression noise by JointSQ.

The optimal design of JointSQ can be achieved by minimizing $h'(b_i, \mathbf{g}_i)$ while satisfying the bit constraint c.

$$\begin{aligned} &\min \ h' \\ &s.t. \sum_{i=1}^{n} k_i * b_i = c. \end{aligned} \tag{10}$$

The design of JointSQ relies on the mixed-precision quantization strategy. However, due to the vast solution space of this optimization problem, directly solving it incurs prohibitively high computational costs. In the next section, we will discuss how to obtain an approximate optimal solution with minimal computational overhead.

3.3. Mixed-precision Quantization

In the quantization strategy generation phase, our goal is to determine the optimal bidwidth assignment. A bitwidth

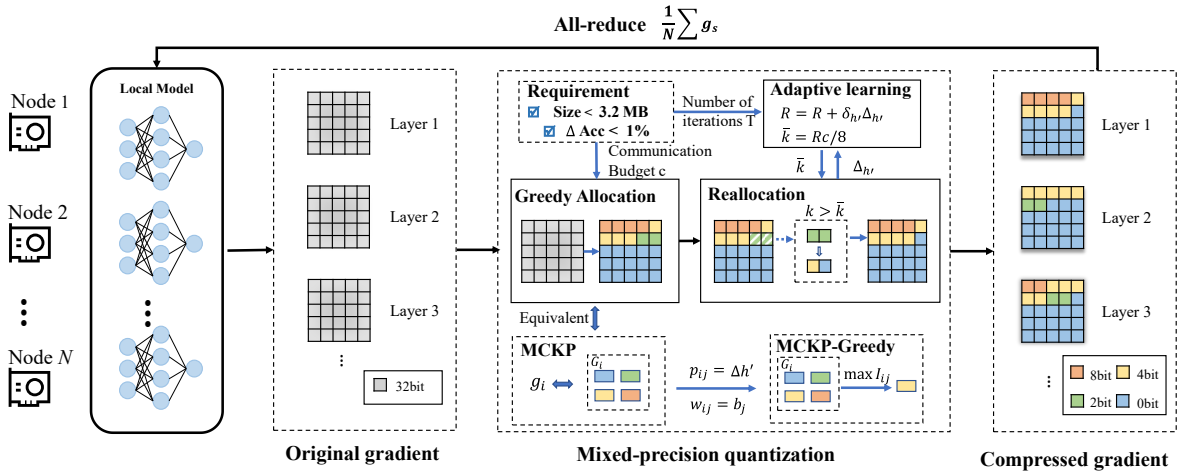


Figure 2. Overview of JointSQ. JointSQ aims to find a mixed precision quantization strategy for each layer, considering the communication budget requirements. Initially, Greedy Allocation approach is used, which can be viewed as solving the Multiple-Choice Knapsack Problem (MCKP). This approach provides an initial allocation strategy based on gradient magnitudes. Subsequently, Reallocation is performed to refine the strategy, with the number of Reallocation iterations determined by the adaptive learning parameter R. The final mixed-precision quantization strategy is obtained by combining the Greedy Allocation and Reallocation.

candidate set $B = \{0, 2, 4, 8\}$ is introduced for limiting the searching space based on two considerations: On the one hand, only power-of-two bitwidths are often efficiently implemented and supported in typical digital hardware [34]. On the other hand, as [4] mentioned, 4-bit or 8-bit gradient quantization is sufficient to recover or even slightly improve full accuracy.

In this section, building upon the theory presented in Section 3.1, we derive a conclusion: Larger gradient elements should be assigned larger bit-width. Based on this conclusion, we propose the following algorithm to solve the optimal bitwidth assignment problem: "Greedy Allocation" and "Reallocation", which can be observed in Figure 2.

3.3.1 Greedy Allocation

To further constrain the searching space, we discuss Eq. (6). For a single gradient element, when the quantization precision is increased from b bits to $b + x$ bits, the variation of the compressed noise is as follows:

$$\Delta h' = \frac{1}{4^b}\frac{g^2}{\|\mathbf{g}\|^2} - \frac{1}{4^{b+x}}\frac{g^2}{\|\mathbf{g}\|^2} = \frac{4^x - 1}{4^x 4^b}\frac{g^2}{\|\mathbf{g}\|^2}. \tag{11}$$

It is evident that when the quantization precision increases, larger components introduce more reduction in noise compared to smaller components. **Thus, the optimal solution to Eq. (10) must satisfy assigning higher quantization precision to the larger components.** This inference also aligns with the concept of Top-k compression methods [2,5,21]. In order to conduct a quantitative analysis, we introduce a special variant of the Knapsack problem called Multiple-Choice Knapsack Problem (MCKP) [18,38].

Definition 3.1 *Multiple-Choice Knapsack Problem (MCKP). There are L mutually disjoint groups $G_1, \ldots, G_L$ which contain $n_1, \ldots, n_L$ items respectively. The $j-th$ item from the $i-th$ group has a "profit" p_{ij}, and "weight" w_{ij}, $\forall i = 1, \ldots, L, j \in 1, \ldots, n_i$. MCKP formulates how to select exactly one item from each group to maximize the sum of profits and keep the sum of weights under a given budge β, i.e.,*

$$\max_{\mathbf{x} \text{ is binary}} \sum_{i=1}^{L}\sum_{j=1}^{n_i} p_{ij}\mathbf{x}_{ij},$$

$$s.t. \sum_{j=1}^{n_i} \mathbf{x}_{ij} = 1, \forall i = 1, \ldots, L; \sum_{i=1}^{L}\sum_{j=1}^{n_i} \omega_{ij}\mathbf{x}_{ij} \leq \beta. \tag{12}$$

Our quantization strategy can be obtained by solving the MCKP. Given a gradient vector $\mathbf{g}$, which consists of d gradient elements denoted as g_i, each group, denoted as G_i, is determined by each g_i and has a fixed size of $|B| = 4$. Thus, i represents the $i-th$ group, where $i \in \{1, 2, \ldots, d\}$. j denotes the $j-th$ item within each group, with $j \in \{1, 2, 3, 4\}$. Each choice of the quantization bit-width is considered as an MCKP item. Then, the Knapsack budget β is c, and x_{ij} indicates selecting which bit-width. We consider the bit-width as the weight of an item, denoted as w_{ij}. We take 0-bit quantization as the baseline, and consider the reduction in quantization noise for each bit-width option as the value of the item based on Eq. (11). It can be formulated as:

$$\begin{aligned} p_{ij} &= \frac{4^{w_{ij}} - 1}{4^{w_{ij}}}\frac{g_i^2}{\|\mathbf{g}\|^2}, \\ w_{ij} &= b_j, b_j \in \{0, 2, 4, 8\}. \end{aligned} \tag{13}$$

Based on the aforementioned definition, we have successfully reformulated the problem into MCKP as shown in the Figure 3.

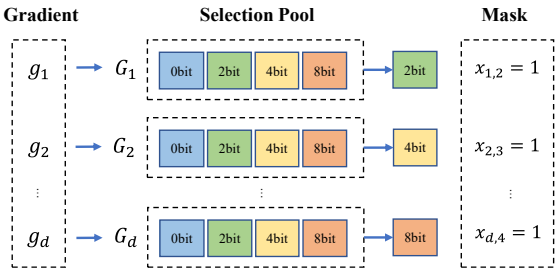


Figure 3. Illustration of the MCKP problem. Each gradient element corresponds to a group, and each choice of bit width represents an item. The selected items are represented by a mask, which is the quantization mask we aim to obtain.

The MCKP is NP-hard [18]. However, prior studies [54] transform the MCKP to the fractional knapsack problem and use a **greedy algorithm** to solve it. Based on this idea, we can get a feasible MCKP solution. "Incremental Profit Density" is a metric that measures the additional profit generated by a unit increase. It can be calculated using the following approach:

$$I_{ij} = \frac{p_{ij} - p_{i,j-1}}{w_{ij} - w_{i,j-1}}. \tag{14}$$

Each time we make a selection, we prioritize the item with the highest Incremental Profit Density. After selecting an item, we remove all items in the current group from the selection pool. This process is repeated multiple times until the maximum capacity of the backpack is reached.

The aforementioned algorithm efficiently discovers a feasible solution for the MCKP and we designate it as "Greedy Allocation". The algorithm exhibits a time complexity of $O(dlog(k))$, where k is the sum of the lengths of the 2, 4, and 8-bit subgradients. We finally obtain a quantization mask, denoted as x_{ij}.

3.3.2 Reallocation

After performing Greedy Allocation, we obtain an approximately optimal solution for Eq. (10). In the later stages of training, we can further improve performance by incorporating some fine-tuning operations after Greedy Allocation. We first define the sum of the lengths 2, 4, and 8-bit subgradients as $k = k_2 + k_3 + k_4$. Then, we introduce a learnable parameter R to dynamically adjust k with the constraint $\bar{k} = Rc/8$, where $c/8$ represents the minimum k.

In each fine-tuning iteration, we attempt and select the direction that reduces compression noise. During fine-tuning, we still follow the principle of using larger bit-widths for larger gradients. For example, when $k < \bar{k}$, we first attempt to decrease the minimum 8-bit gradient element to 4-bit, while increasing the maximum 2-bit gradient element to 4-bit, and elevating the maximum 0-bit gradient element to 2-bit. If noise reduction is observed, we adopt this updating strategy. If noise increases, we attempt to decrease the minimum 4-bit gradient element to 2-bit while elevating the maximum 0-bit gradient element to 2-bit. These adjustments ensure that the communication constraint c is still satisfied and the sum of lengths $k = k + 1$. Therefore, we perform fine-tuning $\left|k - \bar{k}\right|$ times. Conversely, if $k > \bar{k}$, we perform the inverse process. By employing these guidelines and iteratively selecting noise-reducing directions, our compression approach gradually aligns with the optimal solution. This process is referred to as "Reallocation". By exclusively calculating the noise variation induced by the restricted number of modified gradient elements in each fine-tuning iteration, rather than evaluating the compression noise for the entire gradient tensor, the time complexity is reduced to $O(1)$.

3.3.3 Adaptive Strategy For Reallocation

Different layers in deep learning often have varying sensitivities to compression, requiring different compression strategies. We dynamically adjust the value of R for each layer based on the noise reduction achieved in each round of Reallocation. The initial value of R is determined by the results of Greedy Allocation. Given the number of update iterations T for R, our step size is the difference in compressed noise between the current Reallocation and the previous Reallocation, denoted as:

$$\Delta_{h'} = h'_i - h'_{i-1}, \tag{15}$$

where $i \in \{1, 2, ..., T\}$. We determine the learning rate $\delta_{h'}$ based on the magnitude of gradients for each layer. Therefore, our update rule is as follows:

$$R_i = R_{i-1} + \delta_{h'} \Delta_{h'}. \tag{16}$$

Through our proposed Greedy Allocation and Reallocation algorithm, we adaptively determine the compression strategy for each layer that minimizes compression noise. JointSQ provides a Sparsification-Quantization allocation strategy, offering flexibility for specific quantization encoding methods. In our experiments, we employ the widely-used random uniform quantization method. Due to the efficient utilization of parallel computing on GPUs, both the Greedy Allocation and Reallocation processes have low computational overhead. The details of our algorithm are provided in the appendix.

4. Experiments

Dataset and Network. The experiments are conducted on several benchmarks, including the CIFAR-10

Compression approach	ResNet-20		ResNet-110		SimpleViT		VGG-16	
	Acc(%).	Comp.	Acc.	Comp.	Acc.	Comp.	Acc.	Comp.
SGD(baseline)	91.75	1×	93.55	1	84.11	1×	91.02	1×
QSGD [4]	91.56	8×	93.54	8×	84.04	8×	91.41	8×
QLSGD [5]	91.26	20×	93.37	20×	76.82	20×	90.09	20×
AC-SGD [37]	89.88	20×	91.96	20×	78.73	20×	91.04	20×
JointSQ	**91.74**	20×	**93.62**	20×	**84.42**	20×	**91.42**	20×
DGC [21]	91.32	1000×	92.45	1000×	-	1000×	88.12	1000×
QLSGD	91.21	1000×	93.17	1000×	-	1000×	90.14	1000×
AC-SGD	89.61	1000×	91.88	1000×	-	1000×	90.33	1000×
JointSQ	**91.55**	1000×	**93.46**	1000×	-	1000×	**90.78**	1000×

Table 1. Comparison of multiple networks compression results on CIFAR-10.

dataset [19], CIFAR-100 dataset [19], ImageNet dataset [10] and PTB dataset [22]. We evaluate ResNet-18 [15] on the CIFAR-100 dataset, ResNet-101 on the ImageNet dataset and 2-LSTM [27] on the Penn Treebank (PTB) dataset. Specifically, we evaluate multiple networks, including ResNet, Vision Transformer (ViT) [11], and VGG [28], on the CIFAR-10 dataset to validate the generalizability of our framework across different network architectures.

Baselines and Metric. We set the baseline experiment to full-precision SGD [25] and compare our framework with current state-of-the-art gradient compression methods related to ours, as well as some classical methods. These methods include: sparsification method DGC [21], quantization method QSGD [4], as well as Joint methods QLSGD [5], AC-SGD [37], and MCGQ [23]. For our framework, we employ a stochastic uniform quantization similar to QSGD to perform our mixed allocation strategy, and we utilize the gradient accumulation compensation method used in [21] and [23]. For the DGC method, to ensure fairness, we retained only the momentum correction and gradient accumulation. As for the QSGD method, we uniformly applied 4-bit quantization.

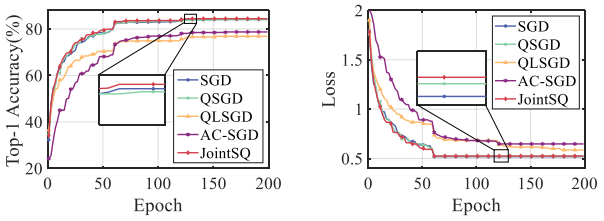
Experimental Setup. Our evaluation uses 4 NVIDIA A100 GPUs with 80GB VRAM on ImageNet and 4 NVIDIA RTX3090 GPUs with 24GB VRAM on other datasets. In addition to conducting comparative training for the aforementioned tasks, we also compared the training time of JointSQ with existing Co-compression methods through a multi-node setup. Furthermore, we tested the impact of different parameters of JointSQ on the experiments. All our experiments use the original training recipes, without any hyperparameter tuning to account for gradient-compressed training.

https : / / github . com / synxlin / deep - gradient - compression
https://github.com/YANGuangfeng/AC-SGD

4.1. Image Classification

4.1.1 CIFAR-10

We conducted comparative experiments on CIFAR-10 at compression ratio of 20× and 1000×, respectively to demonstrate the superiority of JointSQ under different compression strategies. Furthermore, we evaluated our framework across multiple network architectures, including the ViT architecture that has not been deployed in existing joint compression schemes. In our experiments, we utilized a SimpleViT network with 6 encoder layers, 256 hidden dimensions, and 8 attention heads. To ensure fairness, we employed the same experimental setup for all methods, including a learning rate of 0.1 and a batch size of 128. Table 1 presents results on the CIFAR-10 dataset and Figure 4 demonstrates the learning curve of SimpleViT.



(a) CIFAR-10 (b) CIFAR-10

Figure 4. Learning curve of SimpleViT on CIFAR-10. (a) Top-1 accuracy of SimpleViT on CIFAR-10. (b) Loss of SimpleViT on CIFAR-10.

Our experimental results demonstrate that JointSQ outperforms other methods in terms of accuracy preservation at both low and high compression ratios. For instance, in the case of the compression-sensitive ViT network, at a compression ratio of 20×, JointSQ achieves a complete recovery and even surpasses the full-precision SGD by approxi-

ResNet-18 on CIFAR-100				ResNet-101 on ImageNet			
Method	Acc.(%)	Δacc.(%)	Comp.	Method	Acc.(%)	Δacc.(%)	Comp.
SGD(baseline)	73.94	0.00	1×	SGD(baseline)	74.9	0.00	1×
signSGD [6]	71.18	-2.76	32×	DGC	74.46	-0.44	1000×
MCGQ [23]	74.25	0.31	32×	MCGQ	74.54	-0.36	1000×
JointSQ	**75.04**	**1.10**	32×	**JointSQ**	**74.72**	**-0.18**	1000×

Table 2. Performance comparison of JointSQ and existing methods on image classification tasks.

mately 0.3% in accuracy, while other joint schemes exhibit poor performance in handling highly sensitive networks. At a high compression ratio such as 1000×, as observed in ResNet-20, JointSQ exhibits only a 0.2% accuracy loss, outperforming DGC, which has a 0.43% loss. Additionally, the results highlight the strong generalization capability of JointSQ across multiple network architectures. In contrast, other methods show limited generalization. For example, the QLSGD method, which performs well on ResNet, shows subpar performance on ViT and VGG architectures.

4.1.2 CIFAR-100 and ImageNet

In our CIFAR-100 experiments, we used a batch size of 128 and an initial learning rate of 5.6e-2 [17, 23], which is the same as [6]. We compared our framework with full-precision SGD, signSGD, and MCGQ (with a sampling rate of $K = 0.5$) at a compression ratio of 32×. For our ImageNet experiments, we employed a batch size of 128 and an initial learning rate of 0.1. We compared our method with full-precision SGD, DGC, and MCGQ (with a sampling rate of $K = 0.5$) at a compression ratio of 1000×. The results of our experiments can be found in Table 2. Figure 5 illustrates Top-1 accuracy and the loss throughout training. As shown in Table 2, on the CIFAR-100 dataset, JointSQ achieves a 1.10% accuracy improvement compared to full-precision SGD, demonstrating a significant advantage over all compared compression methods. On the ImageNet dataset with an extremely high compression ratio of 1000×, JointSQ only incurs a minor accuracy loss of 0.18%, outperforming both MCGQ and DGC methods which are 0.36% and 0.44% respectively. From Figure 5, it can be observed that JointSQ exhibits the fastest convergence speed compared to other methods. These results validate the outstanding performance of JointSQ in image classification tasks, which can be attributed to its joint sparsification and quantization advantages.

4.2. Language Modeling

In our language modeling task on the PTB dataset, we followed a similar model and training setup as DGC. Specifically, we utilized a 2-layer LSTM with 1500 hidden neurons and trained the model using vanilla SGD with gradient clipping. The initial learning rate was set to 22, and we trained the model for 40 epochs with a batch size of 20.

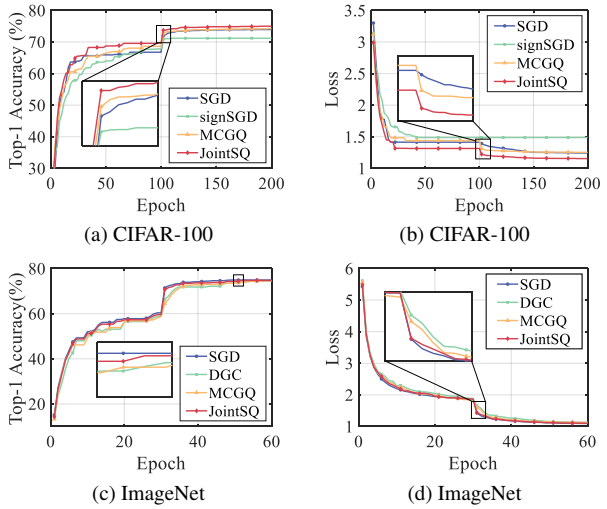


Figure 5. Learning curves of our framework and the comparison methods on image classification tasks. (a) Top-1 accuracy of ResNet-18 on CIFAR-100. (b) Loss of ResNet-18 on CIFAR-100. (c) Top-1 accuracy of ResNet-101 on ImageNet. (d) Loss of ResNet-101 on ImageNet.

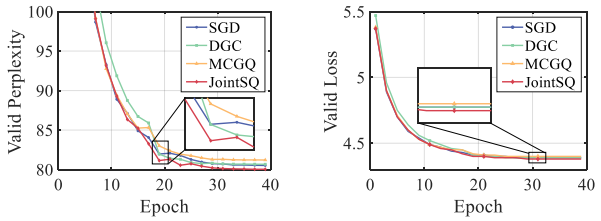
Task	Language Modeling on PTB		
Method	Perplexity	Δppl	Comp.
SGD(baseline)	77.84	0	1×
DGC	76.85	-0.99	1000×
MCGQ	77.67	-0.17	1000×
JointSQ	**76.38**	**-1.46**	1000×

Table 3. Compression results on PTB.

Previous research, including DGC and MCGQ, has already explored the application of models on the PTB dataset. To provide a comprehensive comparison, we evaluated our results against theirs at a compression ratio of

https://github.com/wabluy/ptb_lstm_pytorch

1000×, as presented in Table 3. Notably, our method outperforms full-precision SGD with a perplexity reduction of 1.46, surpassing DGC by 0.47 and MCGQ by 1.29. Figure 6 provide a visualization of the perplexity of JointSQ compared to the baseline methods.

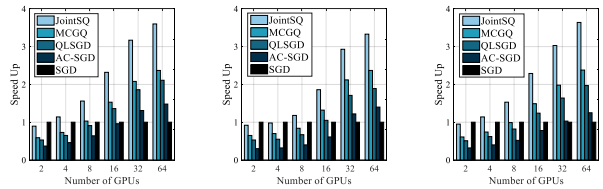


(a) PTB (b) PTB

Figure 6. Learning curves of our framework and the comparison methods on language modeling tasks. (a) Valid perplexity of 2-LSTM on PTB. (b) Loss of 2-LSTM on PTB.

4.3. Computational Overhead

In real-world scenarios, there is a high demand for gradient compression algorithms to have manageable computational overhead, meaning that they should provide tangible acceleration in multi-client distributed training within an acceptable range of computational costs [1, 21]. Our framework, under the same communication overhead, not only achieves performance improvements compared to existing joint methods but also does so without introducing any additional computational overhead. To assess the efficacy of our framework, we vary the number of nodes from 2 to 64 and compare the speedup of JointSQ (with SGD set as 1), as depicted in the Figure 7. As the number of nodes increases, indicating an increased communication demand, JointSQ demonstrates approximately 1.4× to 2.9× acceleration compared to existing Co-compression methods. This acceleration is consistently observed across different node settings, emphasizing the robustness of JointSQ in achieving significant speedups. The efficient computation demonstrated by JointSQ validates its feasibility in real distributed scenarios, where computational costs are a critical factor.



(a) ResNet-20 (b) SimpleViT (c) 2-LSTM

Figure 7. The speed up of JointSQ and existing Co-commpression methods: (a) ResNet-20 on CIFAR-10. (b) SimpleViT on CIFAR-10. (c) 2-LSTM on PTB.

4.4. Further Research

In order to achieve better performance, we attempted to dynamically vary the compression ratio during the training process. A comparative experiment was conducted on a classification task using ViT with a compression ratio set at 100×. In the early stages of training, we employed a lower compression ratio and linearly increased it as training progressed while ensuring the overall communication budget remained consistent with the unified compression ratio. The experimental results are shown in Figure 8.

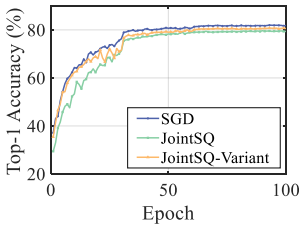


Figure 8. Extended experiments on ViT. We initially employed a lower compression ratio and linearly increased the compression ratio during training.

We observed that using a lower compression ratio in the early stages significantly improved the initial convergence rate. This provides us with insight into the potential combination of JointSQ with periodic variable compression ratios to further meet practical communication demands. It also serves as one of the focal points for our future research exploration.

5. Conclusion

In this paper, we propose an adaptive Joint Sparsification-Quantization (JointSQ) framework that thoroughly unifies sparsification and quantization by treating sparsity as 0-bit. We introduce Greedy Allocation and Reallocation based on the MCKP problem to solve JointSQ. The framework addresses the issue of existing Co-compression approaches falling into suboptimal solutions, thereby further improving the performance of distributed learning. We conduct extensive experiments on various networks and datasets. The experimental results demonstrate that JointSQ achieves higher compression ratios compared to conventional Co-compression methods while maintaining SOTA performance with minimal computational overhead. Specifically, at a compression ratio of 32×, JointSQ achieves faster convergence speed and 1.1% accuracy improvement compared to existing optimal methods while achieving a compression ratio of 1000× on CIFAR-10 and ImageNet with near-lossless performance.

References

[1] Saurabh Agarwal, Hongyi Wang, Shivaram Venkataraman, and Dimitris Papailiopoulos. On the utility of gradient compression in distributed training systems. In *Machine Learning and Systems*, pages 652–672, 2022. 8

[2] Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. In *2017 Conference on Empirical Methods in Natural Language Processing*, 2017. 2, 4

[3] Mohammadreza Alimohammadi, Ilia Markov, Elias Frantar, and Dan Alistarh. L-GreCo: Layerwise-daptive gradient compression for efficient and accurate deep learning. *arXiv preprint arXiv:2210.17357*, 2022. 2

[4] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017. 1, 2, 3, 4, 6

[5] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems*, 2019. 1, 2, 4, 6

[6] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569, 2018. 1, 2, 7

[7] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018. 3

[8] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation*, pages 571–582, 2014. 1

[9] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012. 1

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2, 6

[11] 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. *arXiv preprint arXiv:2010.11929*, 2020. 6

[12] Melih Elibol, Lihua Lei, and Michael I Jordan. Variance reduction with sparse gradients. *arXiv preprint arXiv:2001.09623*, 2020. 1

[13] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel M Roy, and Ali Ramezani-Kebrya. Adaptive gradient quantization for data-parallel SGD. In *Advances in Neural Information Processing Systems*, pages 3174–3185, 2020. 1, 2

[14] Jinrong Guo, Wantao Liu, Wang Wang, Jizhong Han, Ruixuan Li, Yijun Lu, and Songlin Hu. Accelerating distributed deep learning by adaptive gradient quantization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1603–1607, 2020. 1, 2

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

[16] Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. In *Advances in Neural Information Processing Systems*, 2018. 2

[17] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes signSGD and other gradient compression schemes. In *International Conference on Machine Learning*, pages 3252–3261, 2019. 2, 7

[18] Hans Kellerer, Ulrich Pferschy, David Pisinger, Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Multidimensional knapsack problems*. Springer, 2004. 2, 4, 5

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Technical report*, 2009. 2, 6

[20] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, 2014. 1

[21] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017. 1, 2, 4, 6, 8

[22] Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *COMPUTATIONAL LINGUISTICS*, 19(2):313–330, 1993. 2, 6

[23] Gonçalo Mordido, Matthijs Van Keirsbilck, and Alexander Keller. Monte carlo gradient quantization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 718–719, 2020. 1, 2, 6, 7

[24] Philipp Moritz, Robert Nishihara, Ion Stoica, and Michael I Jordan. Sparknet: Training deep networks in spark. *arXiv preprint arXiv:1511.06051*, 2015. 1

[25] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407, 1951. 6

[26] 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 *15th Annual Conference of the International Speech Communication Association*, 2014. 1, 2

[27] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation

nowcasting. In *Advances in Neural Information Processing Systems*, 2015. 6

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Computer Science*, 2014. 6

[29] Liuyihan Song, Kang Zhao, Pan Pan, Yu Liu, Yingya Zhang, Yinghui Xu, and Rong Jin. Communication efficient SGD via gradient sampling with bayes prior. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 12065–12074, 2021. 1, 2

[30] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with memory. In *Advances in Neural Information Processing Systems*, 2018. 2, 3

[31] Nikko Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Interspeech*, 2015. 2

[32] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *International Conference on Machine Learning*, pages 6155–6165, 2019. 3

[33] Wenting Tang, Xingxing Wei, and Bo Li. Automated model compression by jointly applied pruning and quantization. *arXiv preprint arXiv:2011.06231*, 2020. 3

[34] Ying Wang, Yadong Lu, and Tijmen Blankevoort. Differentiable joint pruning and quantization for hardware efficiency. In *European Conference on Computer Vision*, pages 259–277, 2020. 4

[35] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1306–1316, 2018. 1, 2, 3

[36] 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. In *Advances in Neural Information Processing Systems*, 2017. 1, 2

[37] Guangfeng Yan, Tan Li, Shao-Lun Huang, Tian Lan, and Linqi Song. AC-SGD: Adaptively compressed SGD for communication-efficient distributed learning. *IEEE Journal on Selected Areas in Communications*, 40(9):2678–2693, 2022. 1, 2, 3, 6

[38] Haichuan Yang, Shupeng Gui, Yuhao Zhu, and Ji Liu. Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2178–2188, 2020. 2, 4

[39] Shuai Zheng, Ziyue Huang, and James Kwok. Communication-efficient distributed blockwise momentum SGD with error-feedback. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2

[40] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, 2010. 1