CosSGD: Communication-Efficient Federated Learning with a Simple Cosine-Based Quantization

Yang He[1]*, **Hui-Po Wang**[1], **Maximilian Zenk**[2], **Mario Fritz**[1]
[1] CISPA Helmholtz Center for Information Security
[2] Division of Medical Image Computing, German Cancer Research Center (DKFZ)
yanhea@amazon.com, {hui.wang,fritz}@cispa.saarland,
m.zenk@dkfz-heidelberg.de

Abstract

Federated learning is a promising framework to mitigate data privacy and computation concerns. However, the communication cost between the server and clients has become the major bottleneck for successful deployment. Despite notable progress in gradient compression, the existing quantization methods require further improvement when low-bits compression is applied, especially the overall systems often degenerate a lot when quantization are applied in double directions to compress model weights and gradients. In this work, we propose a simple cosine-based nonlinear quantization and achieve impressive results in compressing round-trip communication costs. We are not only able to compress model weights and gradients at higher ratios than previous methods, but also achieve competing model performance at the same time. Further, our approach is highly suitable for federated learning problems since it has low computational complexity and requires only a little additional data to recover the compressed information. Extensive experiments have been conducted on image classification and brain tumor semantic segmentation using the CIFAR-10, and BraTS datasets where we show state-of-the-art effectiveness and impressive communication efficiency.

1 Introduction

Recently, Federated Learning (FL) [22] has emerged as a prominent approach to meet the heavy demands on data privacy and computation resources. It allows a vast number of edge devices, called workers, to train a model collaboratively without sharing private data. Due to the effectiveness of FL, it has been widely adopted in many practical applications, such as keyboard prediction [11, 40] and privacy-sensitive services [31, 30, 24].

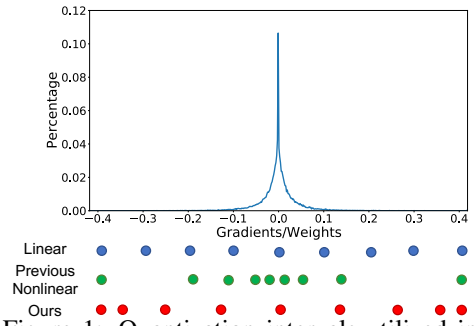


Figure 1: Quantization intervals utilized in the proposed method and prior works.

However, when training FL models, the model weights and gradients are transmitted many times between the server and workers. As the model complexity grows, the communication cost inevitably becomes the major bottleneck of FL systems. To alleviate the issue, many works have made efforts to apply quantization on the model weights [41, 35] or gradients [37, 2] to reduce the message size of model updates. Although the above works have achieved great progress, they often fail to train a model with the similar model performance to float based non-compressed training, when heavy

*Work done before joining Amazon.

1st NeurIPS Workshop on New Frontiers in Federated Learning (NFFL 2021), Virtual Meeting.

compression is deployed. Besides, we observe low-bits quantization may deteriorate other widely used techniques (e.g., gradient sparsification [1, 32]), which limits the further cost reduction.

In this work, we propose a novel quantization method based on cosine function, which is rather simply but effective to quantize model weights and gradients for round-trip compression. Inspired by the key observation that gradients and model weights with larger magnitudes are more informative than the others [10, 21, 1], we leverage the non-linearity of cosine functions for minimizing the quantization errors of extreme values, whereas most prior works [2, 10] aim for an opposite goal as shown in Fig. 1.

In addition, our method has low time/space complexity, requires *no* additional training, and introduces only little additional cost, which makes our approach further favorable to federated learning systems since the edge devices are often equipped with limited hardware resources.

We conduct extensive experiments under various scenarios on CIFAR-10 for image classification as well as BraTS 2019 dataset for 3D volume segmentation. We depict our contributions as follows: (a) As the key contribution, we propose a *simple-yet-effective* cosine-based quantization scheme. It better preserves model performance, has low time/space complexity and requires no extra training. It is thus applicable to a variety of distributed learning problems. (b) Our proposed quantization can be applied to compress model weights and gradients to reduce overall communication costs in federated learning. (c) Our method is flexible and compatible to many existing useful techniques such as gradient sparsification. By combining our quantization with other techniques, our system is able to obtain the similar model performance to the float based non-compressed training at very high compression ratios, while other quantization methods are hard to achieve.

2 Related work

2.1 Federated and distributed learning

Federated learning [39, 20, 12, 18] is a specialized distributed learning framework that allows a vast number of edge devices to jointly train a model while keeping private training data on devices. Federated Averaging (FedAvg) [22] is one of the most classical optimization methods. Recently, FL improves with adaptive optimization. Reddi et al. [25] applies an optimizer on the server with momentum terms to update the model with aggregated gradients, which shows faster convergence and higher performance in many tasks. Mime [13] uses server-level momentum in each client to mitigate the client-drift caused by the heterogeneous data across clients and reach higher performance. FedCD [17] dynamically groups the local workers with similar data distributions to improve the model performance on Non-IID data.

2.2 Gradient compression

We discuss gradient quantization [2, 9, 10, 16, 29] and sparsification [16, 1, 32, 3, 21] in this subsection. Quantization aims to compress the float gradients to low-bits representations. Due to biased approximation, systems may lose performance after quantization. To overcome this, two parallel works, TernGrad [37] and QSGD [2], utilize probabilistic unbiased quantization with correct expectations for quantized gradients. In addition, quantization errors can be reduced by rotations with random Hadamard matrices, which need extra computation [33]. Besides, signSGD [4] shows that the gradient signs – represented with only 1-bit – can be used for optimizing non-convex problems. Lastly, nonlinear quantization methods [10, 9] also study producing fewer quantization errors for the densely distributed data. Meanwhile, communication efficiency is vastly improved by combining with sparsification, which aims to return parts of gradients. The main strategies include random locations [16] and top-K locations [1, 32, 3, 21].

Different from prior work, we design a nonlinear quantization with larger quantization intervals for the less important values and more precise intervals for the values playing a more important role for training.

2.3 Reducing overall communication costs

So far, there are only few works on reducing overall communication costs. Existing works often approach the problem by simplifying the model architectures communicated between the server and workers. For example, Federated Dropout [6] and Adaptive Federated Dropout [5], motivated by

the idea of Dropout, stochastically construct slimmed sub-nets; Liang et al. [19] divide the overall network into local feature extractors and a global classifier. The feature extractors are kept on local devices while only the classifier part is communicated. In addition to federated learning, quantizing model weights and gradients have been explored in general distributed learning [35, 41] with linear quantization [2] to reduce the communication costs for both directions.

As a compensation, we propose a more effective quantization than previous widely used linear quantization [2, 16] and other methods [10, 9, 14, 38] for model weights and gradients, which is compatible with many existing frameworks.

3 Nonlinear quantization with *cosine* function

In this work, we apply the quantization on model weights (server-to-client) and gradients (client-to-server) to reduce the overall round-trip communication costs. To facilitate more effective compression, we propose a novel cosine-based quantization scheme, which achieves high compression ratios and is still manageable to train federated models. In this section, we first describe our cosine-based quantization method and then discuss the properties of the proposed quantization.

For weights and gradients, they have similar properties that the values around 0 are much more than larger values. Besides, the principles for compressing them are similar in that we observe the larger values play a more important role in training, as pointed out by [10].

In federated learning or data-parallel optimization, the workers share the same model $M = (w_1, ..., w_n)^T$ and update it by local data without explicit interactions between workers. After learning with local data, the gradients for the parameters ∇M are sent to the server. Accordingly, the model is updated by aggregating the gradients [22] from all the selected workers $\{\nabla M_i\}_{i=1}^m$ with the following formulation:

$$M^{t+1} = M^t - \eta_s \cdot \frac{\sum_{i=1}^m \nabla M_i \cdot N_i}{\sum_{i=1}^m N_i}, \tag{1}$$

where t is the time step, $\{N_i\}_{i=1}^m$ are the numbers of training data on individual workers, and η_s is the learning rate on the server.

To reduce the round-trip communication costs, we utilize quantization on the model weights from the previous step M^t and gradients ∇M_i on each client. The vector to quantize can be depicted by a set of variables or the angles of the vector to the standard coordinate. We encode the angle information during quantization. Let $\mathbf{v}$ be a vector, where $\mathbf{v} = (v_1, ..., v_n)^T$. The angle between the vector and $i^{-\text{th}}$ axis $\mathbf{a}_i = (\underbrace{0, ..., 0}_{i-1}, 1, ..., 0)^T$ is computed by the inner product:

$$\cos(\theta_i) = \frac{\mathbf{v}^T \mathbf{a}_i}{\|\mathbf{v}\|_2 \cdot 1} = \frac{v_i}{\|\mathbf{v}\|_2}. \tag{2}$$

Accordingly, we have $\theta_i = \arccos(\frac{v_i}{\|\mathbf{v}\|_2})$, where $\theta_i \in [0, \pi]$.

Different from previous linear-based uniform quantization [2, 16], we quantize each value in the form of angles θ instead of the original values or the linearly transformed values [33] in the Euclidean space. Specifically, in a normalized vector, there are very few or even no values close to 1 or -1, especially when the dimension of the vector is high. Therefore, we do not quantize the angle vector $\mathbf{\Theta} = (\theta_1, ..., \theta_n)^T$ on $[0, \pi]$; instead, we compute a bound $b_\theta = \min(\min(\Theta), 1 - \max(\Theta))$ and quantize $\mathbf{\Theta}$ on $[b_\theta, \pi - b_\theta]$. The bound facilitates a more precise quantization in that it avoids the waste of quantization bins for the place where there is no value distributed.

In addition to setting the exact b_θ for a vector, we also clip the top dimensions alternatively. Sometimes, there is one dimension dominating the gradient or weight vector. It will lead to a very large b_θ, whereas most gradients (or weights) are small. Hence, the intensely dominating dimension prevents the others from making full use of the quantization space. To avoid this situation, we clip the top gradients or weights to get the bound b_θ. In the end, our quantization is a nonlinear operation that the space is partitioned unequally for the whole distribution.

We refer to the above vector quantization as Q_v, which is built on Q_θ, a biased linear quantization on the angle space. As pointed by [2, 37], the original quantization is biased because $\mathbb{E}[Q_\theta(\mathbf{\Theta})] \neq \mathbf{\Theta}$.

Accordingly, an unbiased quantization is achieved by a probabilistic procedure. Likewise, our cosine-based quantization can be extended with the probabilistic scheme, satisfying the unbiasedness. Formally, our unbiased s-bits quantization is defined as

$$Q_\theta(\mathbf{\Theta}; b, s) = \begin{cases} \lfloor v \rfloor & \text{with} \quad 1-p \\ \lfloor v \rfloor + 1 & \text{otherwise} \end{cases}, \tag{3}$$

where $v = \frac{\mathbf{\Theta}-b}{\pi-2b} \times (2^s - 1)$, $p = v - \lfloor v \rfloor$, and $\lfloor \cdot \rfloor$ is the round down operator. Finally, both our biased and unbiased methods return a quantized vector $Q_\theta(\mathbf{\Theta})$, the norm of the original gradient vector $\|\mathbf{v}\|_2$, and the bound b_θ. The gradients can be computed on the server by reversing the process.

3.1 Quantization error bound analysis

As depicted above, we quantize the angle vector $\mathbf{\Theta}$ over $[b_\theta, \pi - b_\theta]$. Due to the symmetric property of the cosine function on $[b_\theta, \pi - b_\theta]$, we focus on analyzing our biased quantization in the range of $[b_\theta, \frac{\pi}{2})$, and the left holds the same rule on an opposite direction. The unbiased version can be estimated by taking the expectation form into considerations.

The angle bound b_θ corresponds to a value (weights or gradients) b_v (i.e., $v_i \in [-b_v, b_v]$), where $b_\theta = \arccos(\frac{b_v}{\|\mathbf{v}\|_2})$. Let $q = \frac{\pi - 2 \cdot b_\theta}{2^s - 1}$ be the width of angle quantization intervals, and then the quantization error of v_i is bounded by

$$\begin{aligned} |v_i - Q_v(v_i)| &\leq [\cos(q \cdot (k + \frac{1}{2})) - \cos(q \cdot (k+1))] \cdot \|\mathbf{v}\|_2 \\ &= 2 \cdot \sin(q \cdot (k + \frac{3}{4})) \cdot \sin(q \cdot \frac{1}{4}) \cdot \|\mathbf{v}\|_2, \end{aligned} \tag{4}$$

where $k = \lfloor (\arccos(\frac{v_i}{\|\mathbf{v}\|_2}) - b_\theta)/q \rfloor$, because $\cos(\theta - \epsilon) - \cos(\theta) < \cos(\theta) - \cos(\theta + \epsilon)$ on $[0, \frac{\pi}{2})$.

Observing the Eq. (4), we realize the quantization errors for different intervals are not fixed. Since $\sin(\cdot)$ is monotonically increasing on $[b, \frac{\pi}{2})$, the quantization errors $|v_1 - Q_v(v_1)| < |v_2 - Q_v(v_2)|$ if $|v_1| > |v_2|$. As a result, our method quantizes the gradients with larger magnitudes more precisely, which is important to training [27, 21].

Further, we then compare the quantization errors between the linear method and ours. The biased linear quantization has an error bound of $\frac{b_v}{2^s - 1} = \frac{\cos(b_\theta)}{2^s - 1} \cdot \|\mathbf{v}\|_2$ for all the gradients on $[-b_v, b_v]$. Therefore, for the k-th quantization interval, our method has smaller errors than the linear if

$$2 \cdot \sin(q \cdot (k + \frac{3}{4})) \cdot \sin(q \cdot \frac{1}{4}) < \frac{\cos(b_\theta)}{2^s - 1}. \tag{5}$$

In the Eq. (5), only variable k affects the result. As a result, top 66.6%, 40% and 42.35% quantization intervals from our 2-, 4- and 8-bits compression have smaller error bounds than the linear method. Interestingly, even though the quantization errors from our method are larger than the linear for most variables in a gradient vector, we are able to show more favorable results with higher model performance in section 4. This observation and analysis are inspiring to help us understand gradient quantization in that the success of recovering larger gradients precisely plays a critical role in training.

Besides, our quantization also has an opposite observation to many previous work [10, 42, 9], which learns finer quantization intervals for densely distributed ranges. However, the large gradients are rarely distributed, as an example shown in Fig. 1. In contrast, our solution shows the importance of gradients is also critical for preserving high performance, instead of their distribution only.

3.2 Complexity analysis

We analyze and compare the complexity of our quantization and previous methods [2, 16, 33, 10, 9]. Table 1 reports the computation complexity of quantization and dequantization. Let us consider a n-bits quantization for a m-d vector, which leads to a 2^n level compression. We show that the linear quantization [2] and our method have the lowest complexity at $\mathcal{O}(m)$ because both methods have

Table 1: Comparison of the computational complexity for m-d vectors using n-bits quantization, including linear (L) [2], linear with random Hadamard rotations (L+R) [16, 33], k-means (KM) [10], TinyScript (TS) [9] and ours.

L	L+R	KM	TS	Ours
$\mathcal{O}(m)$	$\mathcal{O}(m + m\log m)$	$\mathcal{O}(m(m+n))$	$\mathcal{O}(mn)$	$\mathcal{O}(m)$

Table 2: Comparison results (%) to previous quantization methods on CIFAR-10 ($B = 50, E = 5, C = 0.1$). Different bits are applied.

Method	Biased (n-bits)			Unbiased (n-bits)		
	8	4	2	8	4	2
float32	85.2					
linear [16]	85.18	85.15	10	85.19	85.16	73.11
k-means [10]	85.17	85.15	82.1	-	-	-
Ours	85.3	85.24	85.17	85.28	85.25	85.2

a closed-form quantization. Further, despite the improved performance from random Hadamard rotations [16, 33], it increases the complexity of compression due to the matrix-vector multiplication. Regarding nonlinear quantization, k-means based methods [10] have $\mathcal{O}(m^2)$ for clustering and $\mathcal{O}(mn)$ for searching. Last, TinyScript [9] also searches on irregular intervals for each dimension at $\mathcal{O}(\log 2^n) = \mathcal{O}(n)$. Consequently, we can clearly see the advantages of our method in terms of computation complexity since our approach needs less computation than other nonlinear approaches, making our method in particular suitable to deploy on edge devices with limited resources.

Additionally, we also discuss the memory costs generated for quantization and dequantization. First of all, we emphasize that our method only needs two float numbers (i.e., bound b_θ and norm $\|\mathbf{v}\|_2$) to perform dequantization. Therefore, the additional communication costs for our method is negligible compared to quantized gradients. Even though the linear quantization and its improved version with random Hadamard rotations also need only two float numbers for defining the quantization range, we point out their performances remain improvable in section 4. On the other hand, k-means needs additional costs of $\mathcal{O}(m)$ for the clustering centroids, leading to more data to communicate between a server and clients. In the end, TinyScript has to create quantization tables for each client, additionally introducing memory loading for edge devices. To conclude, our proposed method has a simple closed-form computation and does not need a lot of additional memory costs, which is suitable for federated learning.

3.3 Overall system

Despite the fruitful merits, we further reduce the costs with a few techniques. First, *Deflate* algorithm [8] is a lossless data-dependent compression method and widely used in applications such as gzip files. We apply the *Deflate* to compress the quantized messages in this work. Second, our method can employ parallel work for further cost reduction. In this work, we utilize random sparsification [16], a common technique for gradient sparsification, to send parts of the gradients and greatly save the communication cost while maintaining comparable performance. As a result, we reduce the server-to-client and client-to-server communication cost significantly.

4 Experiments

We conduct our experiments of federated learning using FedAvg [22] on image classification with CIFAR-10 and brain tumor semantic segmentation with BraTS2018 dataset. All the methods are implemented with PyTorch [23] and PySyft [28]. To test the broad optimization scenarios, we employ SGD [26, 34] or Adam [15] as the basic learners for the clients. Besides, we train the baseline models with float32 and FedAvg, followed by gradients quantization with 8-, 4- and 2-bits compression. For all the settings with quantization (i.e., the baselines and ours), we utilize layer-wise quantization on the neural networks. By default, we apply the biased quantization for ours and perform gradient clipping on top 1% gradients to obtain the bound b_θ as depicted in section 3.

We compare our nonlinear quantization with the standard linear-based method, linear-based probabilistic unbiased quantization (denoted by linear (U)) [2], as well as the improved version (denoted by linear (U, R)) [16] with random Hadamard rotations [33]. Further, we also compare our method with previous nonlinear methods k-means based approach [10] and recent proposed TinyScript [9]. Finally, we compare our method with 1-bit compression, including signSGD [4], and its improved version with error feedback (+EF) [14] and error assimilation (+CSEA) [38]. In our experiment, we first compare the gradient compression in section 4.2, and then discuss the round-trip cost reduction.

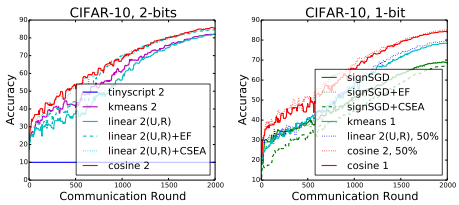


Figure 2: Low-bits comparison results (%) to various compression schemes on CIFAR-10 ($B = 50, E = 5, C = 0.1$).

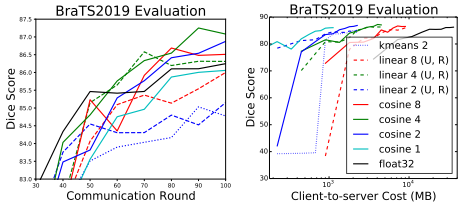


Figure 3: Comparison results (%) to other methods on BraTS2019 dataset ($B = 3, E = 3, C = 1$).

4.1 Datasets and setup

We conduct experiments using CIFAR-10 dataset for image classification, following [22]. We apply a CNN model [36] with 122,570 parameters. For FedAvg, we set $E = 5$ and $B = 50$, and $C = 0.1$.

Besides, we evaluate on 3D semantic segmentation using BraTS dataset, following [30]. We adopt a 3D-UNet [7] with 9,451,567 parameters for this task. For FedAvg, we set $C = 1$, $E = 3$, and $B = 3$. Each worker adopts Adam with betas equal to (0.9, 0.999). BraTS 2019 contains 335 training examples, which covers the BraTS 2018 training set. Therefore, we train a model on the BraTS2018 training set and compute the dice score on the 50 unseen examples.[2]

We train federated models with gradient compression only to compare the performance of different approach in quantizing gradients, as depicted in section 4.2. Further, we apply quantization on model weights and gradients to reduce the overall communication cost in section 4.2.

4.2 Experimental results

Full gradient compression We compare our quantization with linear quantization [2] and k-means based method [10] on CIFAR-10, as listed in Table 3. Although unbiased quantization (e.g. Eq. 3) indeed improves the performance, biased quantization is still helpful for understanding the effect of quantization intervals. Therefore, we conduct experiments using both biased and unbiased version for linear quantization [2] and our methods.

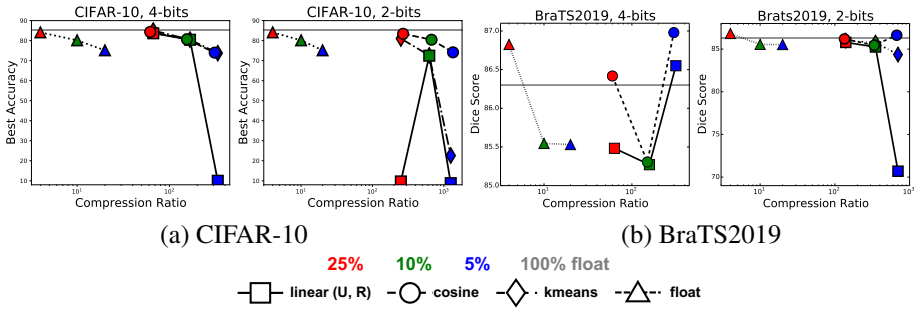
Table 3: Comparison results (%) to previous quantization methods on CIFAR-10 ($B = 50, E = 5, C = 0.1$). Different bits are applied.

Method	Biased (n-bits)			Unbiased (n-bits)		
	8	4	2	8	4	2
float32	85.2					
linear [16]	85.18	85.15	10	85.19	85.16	73.11
k-means [10]	85.17	85.15	82.1	-	-	-
Ours	85.3	85.24	85.2	85.28	85.25	85.2

We highlight the following observations: (1) Even though all the methods work well for high-bits compression, it remains challenging to maintain high model performance in the low-bits case, where only our 2-bits method can achieve comparable results to float32 based training. In particular, 2-bits linear quantization based training has a clear gap to the float based training, even unbiased version is applied. (2) Even though the k-means based method also adopts a non-linear interval, it fails to reach the performance of float32-based training (i.e., 82.1% vs. 85.2% in CIFAR-10), whereas our method resembles the performance of full precision. It clearly supports our hypothesis that larger gradients play a critical role in training.

From the above results, the low-bits cases are the key to achieve breakthroughs. Hence, we focus on more challenging 2-bits and 1-bit compression and compare our methods to recent advanced methods including [16, 14, 4, 38, 9]. Fig. 2 plots the performance curves. From our previous results, we show 2-bits unbiased linear quantization is not enough to achieve decent results, therefore, we boost it with random Hadamard rotations [33, 16], namely linear 2(U,R). We also apply the error feedback (EF) [14] and error assimilation (CSEA) [38] on linear 2(U,R) and signSGD. Last, we also combine 50% random sparsification [16] for linear 2(U,R) and cosine 2, resulting in comparable communication costs for gradients. From this plot, we emphasize several aspects: (1) Our quantization performs quite well in all the cases. Our 1-bit compression only decreases the performance of float32 a little in CIFAR-10 (i.e., 84.2% vs. 85.2%). (2) Even though the EF and CSEA improves 2-bits linear quantization, it is still worse than our method in performance and it needs to store the residual gradients on each client. Further, the improvement of those two methods is not significant

[2]BraTS datasets do not provide GT for the validation set.



(a) CIFAR-10 (b) BraTS2019

Figure 4: Comparison results (%) on the combination of random sparsification [16]. The x-axis with the logarithmic scale indicates the compression ratios on the gradients.

in 1-bit compression due to less frequent interactions in federated learning. (3) We achieve better results than k-means [10, 9], which obviously demonstrates the effectiveness of our quantization.

In addition to image classification, we show the results for brain tumor semantic segmentation in Fig. 3 using BraTS2019 dataset. Because of the effectiveness of unbiasedness and Hadamard rotations [16] from our previous comparison, we only apply the unbiased linear quantization with Hadamard rotations for BraTS2019, which is the strongest setup for the linear. Besides, k-means based method is very expensive for high-bits quantization in the 3D-UNet for segmentation, therefore, we only apply 2-bits k-means for comparison. From Fig. 3, we also achieve better results (Dice score is applied) in this task. Besides, we measure the client-to-server cost over dice scores, which further validates the effectiveness of compression schemes, as drawn in the right plot of Fig. 3. Clearly, we observe that our method achieves better performance while requiring much fewer costs.

Combination with gradient sparsification We report the results by combining gradient sparsification, *Deflate* compression and different quantization methods (2- & 4-bits are considered), as shown in Fig. 4. We return 25%, 10% and 5% gradients for each client to aggregate the global model. In Fig. 4, we first observe the unsuitability of linear, leading to fluctuating performance. Second, we release k-means fails to maintain the performance when 2-bits and 5% sparsification are employed. In contrast, our method can still train a model with good performance and reach very high compression ratio, such as more than $1000\times$ when the extreme compression is applied.

Reducing overall costs In the end, we employ and compare different quantization methods for compressing model weights and gradients. First, we compress the model weights using different methods with various precision. Second, we apply different combincation of various quantization for jointly compressing server-to-client (weights) and client-to-server (gradients) costs. All the results are drawn in Fig. 5. In this experiment, only unbiased linear quantization with Hadamard rotations is tested, because it gives the best performance compared to other versions of linear quantization. We clearly observe the advances of our method. In terms of compressing model weights, our quantization still works well, and achieves better performance over the same level of communication costs. Besides, we can see that only our method is able to compress model weights with 4-bits quantization.

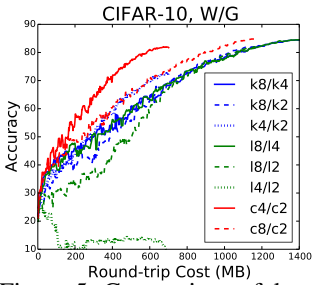


Figure 5: Comparison of double quantization on CIFAR-10. k: k-means; l: linear (U,R); c: cosine. W: weights; G: gradients.

5 Conclusion

We propose a simple quantization scheme, which is in principle different from previous methods - we consider the effects of values to be quantized instead of their distribution. Inspired by the observation that gradients/weights with large magnitudes are more informative, we quantize values according to a non-uniform interval induced by cosine functions, in which larger values have finer levels while the others have coarser levels. Due to the simplicity of cosine functions, our method is a favorable choice for federated learning.We successfully apply the proposed method on model weights and gradients to reduce the round-trip communication cost in federated learning. Furthermore, by incorporating gradient sparsification and lossless data compression, extensive results show that our method is more effective than previous widely used linear and non-linear quantization to reduce communication costs and train a model with simply computation steps at decent performance.

Acknowledgement

This work is partially funded by the Helmholtz Association within the project "Trustworthy Federated Data Analytics (TFDA)" (ZT-I-OO1 4).

References

[1] A. F. Aji and K. Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[2] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *NeurIPS*, 2017.

[3] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli. The convergence of sparsified gradient methods. In *NeurIPS*, pages 5973–5983, 2018.

[4] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *ICML*, 2018.

[5] N. Bouacida, J. Hou, H. Zang, and X. Liu. Adaptive federated dropout: Improving communication efficiency and generalization for federated learning. *arXiv preprint arXiv:2011.04050*, 2020.

[6] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[7] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *MICCAI*, 2016.

[8] P. Deutsch. Deflate compressed data format specification version 1.3. *IETF RFC 1951*, 1996.

[9] F. Fu, Y. Hu, Y. He, J. Jiang, Y. Shao, C. Zhang, and B. Cui. Don't waste your bits! squeeze activations and gradients for deep neural networks via tinyscript. In *ICML*, 2020.

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016.

[11] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[12] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[13] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. Mime: Mimicking centralized stochastic algorithms in federated learning. *arXiv preprint arXiv:2008.03606*, 2020.

[14] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. *NeurIPS*, 2019.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[16] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[17] K. Kopparapu, E. Lin, and J. Zhao. Fedcd: Improving performance in non-iid federated learning. In *SIGKDD*, 2020.

[18] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 2020.

[19] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523*, 2020.

[20] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao. Federated learning in mobile edge networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2020.

[21] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *ICLR*, 2018.

[22] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[24] S. R. Pokhrel and J. Choi. Federated learning with blockchain for autonomous vehicles: Analysis and design challenges. *IEEE Transactions on Communications*, 2020.

[25] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečnỳ, S. Kumar, and H. B. McMahan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[26] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

[27] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[28] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach. A generic framework for privacy preserving deep learning. *arXiv preprint arXiv:1811.04017*, 2018.

[29] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[30] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, et al. Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific reports*, 2020.

[31] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas. Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pages 92–104. Springer, 2018.

[32] N. Strom. Scalable distributed dnn training using commodity gpu cloud computing. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[33] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan. Distributed mean estimation with limited communication. In *ICML*, 2017.

[34] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

[35] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu. Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. In *ICML*, 2019.

[36] T. team. *Tensorflow convolutional neural networks tutorial*, 2016. `http://www.tensorflow.org/tutorials/deep_cnn`.

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

[38] C. Xie, S. Zheng, O. Koyejo, I. Gupta, M. Li, and H. Lin. Cser: Communication-efficient sgd with error reset. In *NeurIPS*, 2020.

[39] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2019.

[40] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[41] Y. Yu, J. Wu, and L. Huang. Double quantization for communication-efficient distributed optimization. In *NeurIPS*, 2019.

[42] D. Zhang, J. Yang, D. Ye, and G. Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, 2018.