ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

Adaptive Lazily Aggregated Quantized Algorithm Based on Federated Self-Supervision

Yifei Ding[1,a,*]

[1]*School of Science and Technology, Hunan University of Science and Technology, Xiangtan, 411201, China*
[a]*648456042@qq.com*
[*]*Corresponding author*

Abstract:*The federated self-supervision framework can solve the problem of large amounts of unlabeled data in traditional federated learning and achieves good results in the entire learning process. However, self-supervised learning increases communication overhead. The limitation of communication capabilities increases the time cost of training and also affects the convergence and accuracy of the model. This paper proposes a method that combines quantization and threshold adaptive aggregation (Adaptive Lazily Aggregate Quantization, ALAQ) to reduce the communication overhead of the federated self-supervised framework. Experimental results prove that ALAQ can effectively reduce the number of communication bits and communication rounds between the client and the server in the federal self-supervision framework. Achieved the purpose of reducing communication overhead.*

Keywords: *Federated Learning, Communication Optimization, Federated Self-Supervision, Gradient Compression*

1. Introduction

In recent years, with the continuous development of machine learning, artificial intelligence and other technologies, the establishment of data models has become an effective data mining solution, which has greatly reduced the cost of manual data mining. Data mining technology based on machine learning has been widely used. Federated learning (FL) is an emergent collaborative framework. In 2017, Google implemented privacy-preserving collaborative model training in a decentralized learning method [1]. In federated learning, because there is a large amount of original unlabeled data on the client side, manual labeling of data requires a huge amount of work and is very costly. Self-supervised learning (SSL) has the ability to learn high-quality data representations from large amounts of unlabeled data, and can effectively utilize unlabeled data for model training, thereby reducing the cost and time of manually labeling data. Chen et al. combined the results of the two research fields of self-supervised learning and federated learning, and based on this, proposed a general federated self-supervised learning (FedSSL) framework, which includes existing SSL methods based on siam networks. And provides the flexibility to adapt to future methods [2] The use of the federated self-supervision framework not only solves the sensitive data protection issue in self-supervised learning, but also solves the problem that the client needs to process a large amount of original unlabeled data in federated learning. But the use of more available data increases a lot of communication overhead and communication costs. Due to the current computing and communication capabilities of the client, the learning performance under the training time budget is reduced [3]. Communication overhead has become an important problem that the overall training effect is too high. Reducing the communication expenditure of federal self-supervision and improving communication efficiency are the main directions of federal self-supervision optimization.

For the optimization of traditional federated learning communication overhead, researchers have also proposed a variety of solutions. These include local update and compression communication, model aggregation optimization, model and data selection, and more. Among them, the quantization scheme is the simplest and most effective compression scheme. The purpose of quantization is to compress gradients and reduce the number of bits in a single communication by limiting the number of bits representing floating point numbers during communication, and has been successfully applied to multiple projects using wireless sensor networks[4]. In the context of distributed machine learning, a 1-bit binary quantization method [5] and a multi-bit quantization scheme [6] have been applied. In terms of model aggregation optimization, Jun Sun et al[7]. proposed a novel aggregation gradient idea, which first

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

quantifies the calculated gradient, and then skips the less informative quantized gradient communication by reusing the outdated gradient. On this basis, Chen developed a new Lazily Aggregated Gradient (LAG) algorithm [8], which can adaptively calculate gradients and skip part of the gradient communication, thereby reducing communication bandwidth and relieving server pressure.

Aiming at the problems of high communication overhead and low communication efficiency for federal self-supervision. We propose Adaptive Lazily Aggregated Quantized algorithm (ALAG) based on federated self-supervision. ALAG uses a dynamic quantization method to process model parameters and gradients uploaded by the client, reducing the number of transmission bits. Then gradient filtering is performed through the Adaptive Lazily Aggregated solution to reduce communication rounds during uploading. The two are combined to optimize communication in the federated self-supervised learning process. This enables the existing federal self-supervision framework to reduce the total number of communication bits.

2. Methods and Model

2.1. Dynamic quantification

Dynamic quantification is similar to the encryption and decryption process in an asymmetric cryptosystem. It is encoded on the client and decoded when uploaded to the server. Because there is an additional decoding process, the error of dynamic quantization is much smaller than that of static quantization. The specific dynamic quantification scheme is as follows.

We uses INT8 to set X_{sf} as the scale factor, Z_q as the zero point in the quantized value, $W_f \in [W_f^{min}, W_f^{max}]$ and $W_i \in [Q_{min}, Q_{max}]$ represent the floating point number before quantization and the integer weight after quantization respectively. It can be known from the literature [9] that X_{sf} and Z_q can be expressed as :

$$X_{sf} = \frac{W_f^{max} - W_f^{min}}{Q_{max} - Q_{min}} \tag{1}$$

$$Z_q = \begin{cases} Q_{\max}, & \frac{W_t^{\max}}{X_{sf}} \in (-\infty, 0) \\ Q_{\max} - \frac{W_t^{\max}}{X_{sf}}, & \frac{W_t^{\max}}{X_{sf}} \in (0, Q_{\max} - Q_{\min}) \\ Q_{\min}, & \frac{W_t^{\max}}{X_{sf}} \in (Q_{\max} - Q_{\min}, +\infty) \end{cases} \tag{2}$$

The weight quantification formula from 32-bit floating point type (FP32) to 8-bit integer type (INT8) is as follows:

$$W_i = \left\lfloor Z_q + \frac{W_t}{X_{sf}} \right\rfloor \tag{3}$$

After receiving the quantized weight, the server can restore the 8-bit integer (INT8) to the 32-bit floating point (FP32) through the following formula:

$$W_f^{'} = X_{sf}(W_i - Z_q) \tag{4}$$

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

2.2. Lazily aggregated algorithm

He main idea of lazy aggregation is: before uploading this gradient, the local end first compares it with the gradient uploaded in the previous round, and calculates the difference between the two. If the difference is too small to meet the "lazy" condition. The client accumulates this gradient without uploading it, while the server uses the client's old gradient from the previous round.

During model training, it is assumed that the server communicates with the *m* client, and *M* represents the set of clients selected by the server. At the *k*-th loop iteration, the server will deliver the current global model $W_g^o = \left(W_g, W_g^p\right)$ to all clients. Then the client calculates the local parameters $\nabla F_m\ (\theta(k-1))$, and after quantification, it is judged through lazy aggregation and uploaded to the server. The sum of the model parameters is defined as ∇F_M^{k-1}.

η is the learning rate of the client. The average gradient parameter of the client after multiple iterations is defined as:

$$W_i^l(k) = W_i^l(k-1) - \eta_{K_1} \nabla F_k(W_i^l(k-1)) \tag{5}$$

Let M_{L} represent lazy nodes and M_{H} represent diligent nodes. The total node $M=M_{\mathrm{L}}+M_{\mathrm{H}}$, the sum of parameters is expressed as:

$$\nabla F_M^{k-1} = \nabla F_{M_{\mathrm{L}}}^{k-1} + \nabla F_{M_{\mathrm{H}}}^{k-1} \tag{6}$$

For lazy nodes, this article defines:

$$\frac{||\nabla F_{M_{\mathrm{L}}}^{k-1}||^2}{M_{\mathrm{L}}} \le \frac{||\nabla F_M^{k-1}||^2}{M} \tag{7}$$

Substituting the average gradient parameters of the client after multiple iterations into formula (7):

$$\left\|\nabla F_{M_{\mathrm{L}}}^{k-1}\right\|^2 \le \frac{M_{\mathrm{L}}}{\eta^2 M}\left\|\theta^k - \theta^{k-1}\right\|^2 \tag{8}$$

Among them, θ^k represents the k round parameter of the server, the model parameter $g\ (\theta_k)=\nabla F\ (\theta(k))$. Therefore, the sum of model parameters in round k–1 can be expressed as:

$$\left\|\nabla F_{M_{\mathrm{L}}}^{k-1}\right\|^2 = \sum_{m \in M_{\mathrm{L}}} ||\nabla F_m(\theta^{k-1})||^2 \tag{9}$$

From the mean inequality, we can get:

$$\left\|\nabla F_{M_{\mathrm{L}}}^{k-1}\right\|^2 \le M_{\mathrm{L}} \sum_{m \in M_{\mathrm{L}}} ||\nabla F_m(\theta^{k-1})||^2 \tag{10}$$

Therefore, if the device node is a Lazy node, that is, when m∈ML, if the node parameters satisfy Equation (11), then the update parameters satisfy the gradient descent algorithm $\theta^k = \theta^{k-1} - \eta \nabla F_M^{k-1}$

$$\left\|\nabla F_m(\theta^{k-1})\right\|^2 \le \frac{1}{\eta^2 M M_{\mathrm{L}}}\left\|\theta^k - \theta^{k-1}\right\|^2 \tag{11}$$

We set as the proportion coefficient of lazy and all nodes, indicating the proportion of lazy nodes:

$$M_{\mathrm{L}} = \mu M \tag{12}$$

Substituting equation (12) into equation (11)

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

$$\left\|\nabla F_m(\theta^{k-1})\right\|^2 \le \frac{1}{\eta^2 \alpha M^2}\left\|\theta^k - \theta^{k-1}\right\|^2 \tag{13}$$

It is difficult to calculate the difference between the k round model parameters and the k-1 round model parameters separately $\theta^k - \theta^{k-1}$, because the iteration difference of the model itself does not change significantly, so we sum up the parameter differences of each round and take the average, which can be Approximately:

$$\theta^k - \theta^{k-1} \approx \sum_{d=1}^{D} \varepsilon_d (\theta^{k-d} - \theta^{k-1-d}) \tag{14}$$

ϵ_d and D are constant proportions, and the settings do not affect the formula results. Set $\varepsilon_d = 1/_D$,d=1 to simplify the calculation .Substituting equation (14) into equation (12), we can get the lazy node determination formula, that is:

$$\left\|\nabla F_m\left(\theta^{k-1}\right)\right\|^2 \le \frac{1}{\eta^2 \mu M^2}\left\|\sum_{d=1}^{D} \varepsilon_d (\theta^{k-d} - \theta^{k-1-d})\right\|^2 \tag{15}$$

Use (15) to determine the existing upload gradient. When the conditions are met, it is proved that the gradient change is not enough, and it is an inert node. It will no longer upload, and will continue to dormant and accumulate gradients. If the inertia determination is not met, it will be a normal node and participate in the server. End aggregation.

2.3. System Model

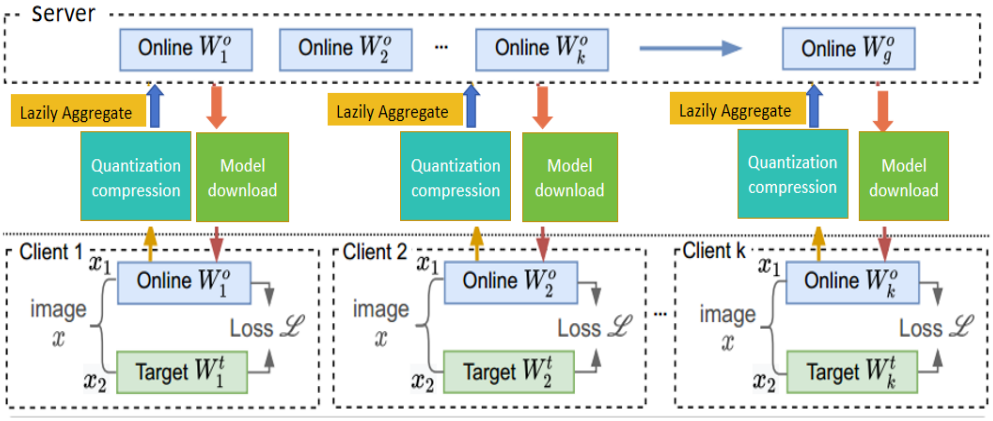

Figure 1: Lazily Aggregated Quantized algorithm based on federated self-supervision.

First, the server delivers the original model (the encoder and predictor in the server replace the encoder and predictor in the client respectively), secondly, the client's own local data performs self-supervised learning iterative calculation to obtain the weights of the model, and then Dynamic quantization and lazy aggregation schemes are used for node selection, and the processed local weights are uploaded to the server. Finally, the server restores and aggregates the received quantized weights, and updates the global model (encoder and predictor) and broadcasts it to each client(Figure 1).

3. Experimental design and Results

3.1. Basic Settings

Implementing FedSSL in Python using the popular deep learning framework PyTorch This article uses ResNet-18 as the default network for the encoder, using the first convolutional layer with a kernel size of 3 × 3, and a kernel size of 4 before the last linear layer. A × 4 average pooling layer replaces the

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

adaptive average pooling layer. The predictor is a two-layer multilayer perceptron (MLP) starting from a fully connected layer of 4096 neurons. This is followed by 1D batch normalization and ReLU activation function, and finally another fully connected layer with 2048 neurons. By default, this article trains R = 100 epochs, K = 5 clients, E = 5 local epochs, batch size B = 128, Initial learning rate $\eta = 0.032$, and cosine decay. Experimental settings GD, QGD, LAQ, LAG, ALAQ experimental comparison, this paper sets the gradient quantization b in the neural network to 8 bits.

3.2. Bvaluation Metrics

The experiments in this chapter focus on the two data sets of CIFAR-10 and CIFAR-100 to conduct related experiments to evaluate model accuracy, compression rate, comprehensive compression index, convergence speed, number of transmission bits, etc.

The purpose is to observe the optimal ratio of model parameters and gradient compression under different data sets. Excessive compression will affect the accuracy, and a small amount of compression will not minimize the communication overhead. What is the overall effect of compression? Can model parameters and gradient compression be completed without affecting model accuracy to reduce communication overhead.The final sentence of a caption must end with a period.

3.2.1. Top-1 Accuracy (Acc)

Model effect judgment is the ratio of the number of correct classifications in the test sample to the total number of samples in the test set, that is, the accuracy of the model prediction. The higher the better.

3.2.2. Compression Ratio(CR)

The compression ratio is used to measure the gradient compression effect. The compression rate represents the degree of compression of the gradient. The smaller the compression rate, the higher the degree of compression. The definition of compression rate: the ratio of communication rounds after compression communication to before communication.

3.2.3. Composite Compression Index(CCI)

The comprehensive compression index is a calculation method that dynamically measures the balance between compression rate and accuracy. The higher the compression rate, the fewer communication rounds and the lower the accuracy. Reducing the compression rate can improve the accuracy to a certain extent, resulting in local contradictions, but it is of great help in comprehensively considering the effect of the entire model. CCI is defined as follows:

$$\mathrm{CCI} = \beta_1 \times \mathrm{Acc} + \beta_2 \times (1 - \mathrm{CR}) \tag{16}$$

$\beta 1$ and $\beta 2$ represent the proportional coefficients of Acc and CR respectively, $\beta 1 > 0$, $\beta 2 > 0$, and $\beta 1 + \beta 2 = 1$. The higher the CCI, the better the gradient compression effect of the model.

3.3. Gradient compression experiment

The compression ratio is used to measure the gradient compression effect. The compression rate represents the degree of compression of the gradient. The smaller the compression rate, the higher the degree of compression. The definition of compression rate: the ratio of communication rounds after compression communication to before communication.

Table 1: Model detection accuracy and compression rate under different values CIFAR-10.

μ	N_1	N_2	Accuracy(%)	CR(%)
0.1	500	42	83.11	8.40
0.2	500	220	84.23	44.00
0.3	500	242	84.11	48.40
0.4	500	260	84.29	52.00
0.5	500	312	84.04	62.40
0.6	500	325	83.71	65.00
0.7	500	357	83.79	71.40
0.8	500	418	83.14	83.60
0.9	500	456	82.99	91.20
1.0	500	500	83.25	100

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

In order to determine the value of the optimal coefficient in gradient compression, this article uses 80% of the CIFAR-10 and CIFAR-100 data sets for training and 20% for random testing, and uses the ALAQ quantified lazy aggregation method for model compression. Conduct multiple experiments, set the proportion parameter to different values according to intervals, observe the communication rounds and accuracy through experiments, and use experimental methods to obtain the optimal effect. μ represents compression factor, N1 represents the communication round before compression, N2 represents the communication round after compression, Accuracy is the model accuracy, and CR is the compression rate.

Table 1 shows that, In the CIFAR10 data set, after setting different compression coefficients, the accuracy and compression rate changes, and the best effect is around the compression coefficient of 0.1.

Table 2: Model detection accuracy and compression rate under different values CIFAR-100.

μ	N_1	N_2	Accuracy(%)	CR(%)
0.1	500	46	58.65	9.20
0.2	500	221	59.55	44.20
0.3	500	248	58.98	49.60
0.4	500	264	59.26	52.80
0.5	500	301	59.04	60.20
0.6	500	334	59.71	66.80
0.7	500	377	58.89	75.40
0.8	500	430	59.14	86.00
0.9	500	455	58.68	91.00
1.0	500	500	58.63	100

Table 2 shows that, In the CIFAR100 data set, after setting different compression coefficients, the accuracy and compression rate changes, and the best effect is around the compression coefficient of 0.1.

Experimental results show that maintaining the compression coefficient at 0.1 has the best compression effect for the overall model. Therefore, the subsequent experiments in this chapter will uniformly set the compression coefficient of the algorithm to 0.1 to ensure the best effect of the algorithm.

3.4. Accuracy evaluation experiment

There are two general experimental methods used to measure the performance of self-supervised models on image classification tasks, Linear Evaluation [10] and Semi-supervised Learning [11].

Table 3: The accuracy of different compression schemes on different datasets(Linear Evaluation).

Method	network	CIFAR-10	CIFAR-100
GD	Resnet-18	78.61	57.51
QGD	Resnet-18	76.32	56.95
LAQ	Resnet-18	83.16	58.54
LAG	Resnet-18	78.22	57.56
ALAQ	Resnet-18	84.11	58.45

Table 3 shows that,In linear evaluation method,ALAQ has higher accuracy when tested on different data sets.

Table 4: The accuracy of different compression schemes on different datasets(Semi-supervised Learning).

	CIFAR-10		CIFAR-100	
	1%	10%	1%	10%
GD	70.48	76.23	30.63	47.25
QGD	70.36	76.12	30.55	46.32
LAQ	74.22	80.42	31.66	46.77
LAG	70.39	76.66	30.68	47.05
ALAQ	74.78	77.13	31.87	47.44

Table 4 shows that,In the Semi-supervised Learning experiment, whether in the CIFAR-10 data set or the CIFAR-100 data set. The ALAQ algorithm has better accuracy for compressing less labeled data, and the accuracy of the model is improved compared to the best existing LAQ method.

ISSN 2616-5775 Vol. 7, Issue 5: 72-78, DOI: 10.25236/AJCIS.2024.070509

Table 5: The accuracy of different compression schemes on different datasets LE CIFAR10.

CIFAR-10	iterations	round	Bit	accuracy
GD	500	5000	$2.543x10^{10}$	78.61
QGD	500	5000	$6.375x10^{9}$	76.32
LAQ	500	1125	$2.531x10^{9}$	83.16
LAG	500	1016	$9.501x10^{9}$	78.22
ALAQ	500	420	$0.864x10^{9}$	84.11

Table 6: The accuracy of different compression schemes on different datasets LE CIFAR100.

CIFAR-100	iterations	round	Bit	accuracy
GD	500	5000	$3.241x10^{11}$	57.51
QGD	500	5000	$8.126x10^{10}$	56.95
LAQ	500	1125	$3.189x10^{10}$	58.54
LAG	500	1016	$1.198x10^{11}$	57.56
ALAQ	500	450	$1.088x10^{10}$	58.45

Table 5 and Table 6 shows that, ALAQ algorithm has lower communication rounds and fewer communication bits, Linear evaluation on different data sets(CIFAR10 and CIFAR100) proved this.

4. Conclusion

In the experiments on the two data sets CIFAR-10 and CIFAR-100,GD and QGD only compressed the number of transmission bits, but the number of communication rounds was not reduced. Compared with LAQ, LAG had fewer communication rounds. But the total number of transmitted bits is not as good as LAQ. Compared with the GD method, ALAQ compresses 90% of the communication rounds and nearly an order of magnitude more communication bits without affecting the accuracy of the model. The ALAQ method performs well on the federated self-supervision framework FedBYOL. It has lower communication overhead than LAQ and higher model accuracy than the original framework.

References

[1] McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data[C]//Artificial intelligence and statistics. PMLR, 2017: 1273-1282.
[2] Chen X, He K. Exploring simple siamese representation learning[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 15750-15758.
[3] Fang P F, Li X, Yan Y, et al. Connecting the Dots in Self-Supervised Learning: A Brief Survey for Beginners [J]. Journal of Computer Science and Technology, 2022, 37(3): 507-526.
[4] Msechu E J, Giannakis G B. Sensor-centric data reduction for estimation with WSNs via censoring and quantization[J]. IEEE Transactions on Signal Processing, 2011, 60(1): 400-414.
[5] Bernstein J, Wang Y X, Azizzadenesheli K, et al. signSGD: Compressed optimisation for non-convex problems[C]//International Conference on Machine Learning. PMLR, 2018: 560-569.
[6] Qu Z, Zhou Z, Cheng Y, et al. Adaptive loss-aware quantization for multi-bit networks [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 7988-7997.
[7] Sun J, Chen T, Giannakis G B,et al.Communication-Efficient Distributed Learning via Lazily Aggregated Quantized Gradients.2019[2024-03-25].DOI:10.48550/arXiv.1909.07588.
[8] Chen T, Giannakis G B, Sun T,et al.LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning[J]. 2018.DOI:10.48550/arXiv.1805.09965.
[9] Xu W, Fang W, Ding Y,et al.Accelerating Federated Learning for IoT in Big Data Analytics With Pruning, Quantization and Selective Updating[J].IEEE Access, 2021, PP(99):1-1.DOI:10.1109/ ACCESS. 2021.3063291.
[10] Kolesnikov A, Zhai X, Beyer L. Revisiting self-supervised visual representation learning[C] //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 1 920-1929.
[11] Beyer L, Zhai X, Oliver A,et al.S4L: Self-Supervised Semi-Supervised Learning[C]//International Conference on Computer Vision.0[2024-03-25].DOI:10.1109/ICCV.2019.00156.