Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 140-150
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Edge-Federated Self-Supervised Communication Optimization Framework Based on Sparsification and Quantization Compression

Yifei Ding

School of Science and Technology, Hunan University of Science and Technology, Xiangtan, China
Email: 648456042@qq.com

How to cite this paper: Ding, Y.F. (2024) Edge-Federated Self-Supervised Communication Optimization Framework Based on Sparsification and Quantization Compression. *Journal of Computer and Communications*, **12**, 140-150.
https://doi.org/10.4236/jcc.2024.125010

Received: April 26, 2024
Accepted: May 28, 2024
Published: May 31, 2024

Abstract

The federated self-supervised framework is a distributed machine learning method that combines federated learning and self-supervised learning, which can effectively solve the problem of traditional federated learning being difficult to process large-scale unlabeled data. The existing federated self-supervision framework has problems with low communication efficiency and high communication delay between clients and central servers. Therefore, we added edge servers to the federated self-supervision framework to reduce the pressure on the central server caused by frequent communication between both ends. A communication compression scheme using gradient quantization and sparsification was proposed to optimize the communication of the entire framework, and the algorithm of the sparse communication compression module was improved. Experiments have proved that the learning rate changes of the improved sparse communication compression module are smoother and more stable. Our communication compression scheme effectively reduced the overall communication overhead.

Keywords

Communication Optimization, Federated Self-Supervision, Sparsification, Gradient Compression, Edge Computing

1. Introduction

In traditional machine learning scenarios, the client needs to transfer local data to the server for model training. This presents a serious data privacy breach problem. In addition, a large amount of effective training data is often in the

hands of people in different fields and industries, and their role cannot be fully grasped, leading to the phenomenon of data islands. The concept of a federated learning framework was first proposed in 2015 [1], which can achieve distributed model training using different data scattered on multiple devices while protecting data privacy, thereby improving data security and privacy. In federated learning, because there is a large amount of original unlabeled data on the client side, manual labeling of data requires a huge workload and is very costly. Supervised learning not only relies on expensive annotations, but also suffers from problems such as generalization errors, spurious associations, and adversarial attacks [2]. Self-supervised learning (SSL) is an unsupervised learning method that aims to generate labels or tasks through the structure or characteristics of the data itself for model training. Self-supervised learning methods can effectively learn high-quality data representations from large amounts of unlabeled data. Combining the results of the two research fields of self-supervised learning and federated learning, a generalized federated self-supervised (FedSSL) framework is proposed, which includes existing SSL methods based on Siamese networks and provides future adaptation Method flexibility [3].

The use of the federated self-supervision framework not only solves the problem of sensitive data protection 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. However, using more available data adds significant communication solutions and communication costs. Client-side self-supervised learning can improve model performance, and the processing of unlabeled data is also the process of labeling by self-supervised learning itself. After many iterations, in the model parameter and gradient upload aggregation stage, the total data volume is larger than that of traditional federated learning, and the communication overhead is greater. The server needs to communicate with the client frequently to transmit updated model parameters and gradients, because in the server aggregation stage, you need to wait for all clients to complete uploading before aggregating. Due to the current computing and communication capabilities of the client, the learning performance under the training time budget is reduced [4]. Moreover, the communication distance between the client and the central server may be long, resulting in high communication speed delays and even the risk of communication interruption. Therefore, reducing communication overhead and improving communication efficiency are the main directions of federated self-supervision optimization.

In order to solve the problem of low communication efficiency between the client and the central server, researchers have proposed an edge computing model. The edge computing model can effectively reduce the communication pressure of the central server. In terms of data processing, because the edge does not need to process the data of all clients, it only needs to process the data of clients within its jurisdiction, so the processing is faster, more immediate, more accurate, and more intelligent. It can also reduce the energy consumption of the central server. The cloud-edge architecture that combines cloud servers, edge

servers, and terminal devices has been used in the field of artificial intelligence [5]. Existing research shows that deploying computing closer to the client can reduce computing load [6]. Researchers discussed the computing optimization issues caused by adding edge computing to the original federated learning system. It is necessary to optimize communication [7].

To address the problem of excessive communication overhead in traditional federated learning, researchers proposed to reduce the total amount of uploaded data, reduce the number of federated learning communication rounds, and use sparse methods or quantification methods to reduce communication overhead. Sparsification only transmits gradient coordinates that are large enough, and abandons transmitting other gradient coordinates that do not meet the filtering requirements. Although deleting large amounts of gradient data may intuitively affect model accuracy, empirically, even reducing gradients by 99% can achieve the desired accuracy [8]. For example, by extending the Sparse Ternary Compression (STC) framework of the existing top-k gradient dilution compression technology to specifically meet the needs of the federated learning environment [9], the sparse gradient can still bring good model accuracy, but deterministic sparsification. The solution still lacks performance analysis guarantees. Quantization aims 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 several engineering tasks using wireless sensor networks [10]. In the context of distributed machine learning, a 1-bit binary quantization method [11] and a multi-bit quantization scheme [12] have been applied.

Existing research directions on federated self-supervised learning mostly focus on optimizing data processing on the client [13] and improving model aggregation effects [14], while less on communication optimization of the federated self-supervised framework.

In conclusion, In order to reduce the communication overhead of the federated self-supervision framework, we combined the communication compression methods of edge computing and traditional federated learning to build a new federated self-supervision communication optimization framework.

The key contributions of this research's work are as follows:

- Introduce the concept of edge computing into the federal self-supervision framework, and solve the problem of low communication efficiency and high communication delay between clients and central servers by adding edge servers.
- Use the sparse gradient compression module to reduce the communication overhead when uploading client model parameters and gradients, and improve the adaptive learning rate optimization algorithm (Adam algorithm) in the sparse communication compression module to make the learning rate more stable.
- Use Quantization compression methods to reduce the communication overhead when downloading the model on the server side and further reduce the

total communication overhead.

2. Methodsand Model

2.1. System Model

The framework process is as follows:

- Initialize the central server-side model parameters W_g (encoder W_g^0 and predictor W_g^p), and send the initialized original parameters to each edge server.
- The edge server delivers the initial parameters to each subordinate client.
- Use Algorithm 2.3 - 2.5 to process the upload parameters. Model parameters and gradient compression during upload.
- The edge server aggregates and quantitatively sends it to the client, and accumulates gradients.
- After several rounds of intermediate aggregation, it is uploaded to the central server.
- Average gradient quantification obtained by adding the sparse tensors uploaded by the edge server to the central server.
- Download the model to the edge server.
- Loop iteration.

Figure 1 shows the entire edge-system component of the federal self-supervision framework.

2.2. Gradient Sparsification and Top-k Gradient Selection

In an edge computing environment, the client relies on local data for self-supervised learning to train the model. The model parameters and gradients of the local model are sparse and then uploaded to the edge server, which can alleviate communication bandwidth pressure [15]. During the gradient sparsification process, we set Top-k as the sparsification method filter and set a gradient threshold K. Only the gradients whose size reaches K can be aggregated. If not, gradient accumulation is performed locally iteratively, and iterations are repeated until The K value is reached in a certain round.

Let $F(w)$ be the total loss function, $f(x,w)$ represents the loss of sample x, N clients, the minimum batch processing b of nodes, during the gradient accumulation process

$$F(w)=\frac{1}{|\chi|}\sum_{x\in\chi} f(x,w) \tag{1}$$

$$w_{t+1}=w_t-\lambda\frac{1}{Nb}\sum_{k=1}^{N}\sum_{x\in\Phi\kappa,\tau+o}\nabla^i f(x,w_t) \tag{2}$$

λ represents the learning rate, χ is the data set, w represents the weight of the network, ΦK, τ+ o represents the sequence of total size Nb from the data set χ during training.

w_i is the weight at the T-th position, then the weight after the T round is

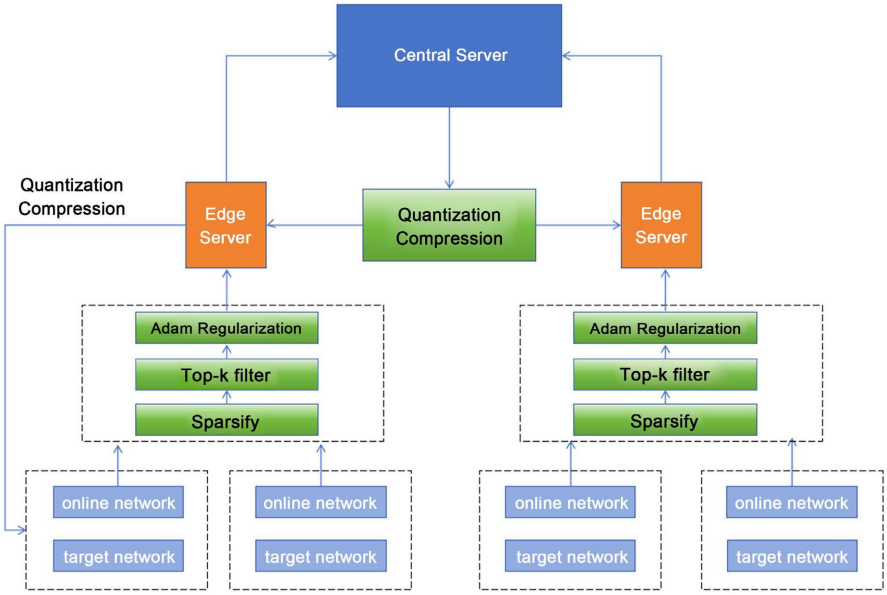

Figure 1. Federated self supervised communication compression model using edge computing.

$$w_{i,t+T} = w_{i,t} - \lambda T \times \frac{1}{NbT} \sum_{k=1}^{N} \left(\sum_{\tau=0}^{T-1} \sum_{x \in \Phi\kappa, \tau+o} \nabla^i f\left(x, x_{t+o}\right) \right) \tag{3}$$

From formula (3), we can see that the batch processing of the gradient accumulation process increases from Nb to NbT, where T is the gradient The length of the sparse update interval for *wi* iterations.

Sparse updates will slowly affect the convergence of the model [16]. In a gradient sparse scenario, the momentum method is used for momentum correction, and the gradient is updated as follows:

$$\begin{aligned} a_{k,t} &= a_{k,t-1} + \frac{1}{Nb} \sum_{x \in \Phi k,t} \nabla f\left(x, w_t\right) \\ r_t &= mr_{t-1} + \sum_{k=1}^{M} spa\left(a_{k,t}\right) \\ w_{t+1} &= w_t - \alpha r_t \end{aligned} \tag{4}$$

m represents momentum, $a_{k,t}$ represents the gradient accumulation of training edge node k. When the gradient value accumulation value reaches K, sparse uploading, the weight w_i after the sparse update interval T is

$$w_{i,t+T} = w_{i,t} - \lambda\left(\nabla^i_{k,t+T} \cdots + \nabla^i_{k,t+1} + \nabla^i_{k,t}\right) \tag{5}$$

Momentum-corrected gradients stabilize the size of the gradient sparse update interval T. The local accumulated gradient $a_{k,t}$ replaces the real gradient $\nabla_{k,t}$, and the accumulated gradient value $a_{k,t}$ after vector correction is subsequently used for gradient sparseness. The formula of (4) be:

$$\begin{aligned} a_{k,t} &= a_{k,t-1} + r_{k,t} \\ r_{k,t} &= r_{k,t-1} + \frac{1}{Nb} \sum_{x \in \Phi k,t} \nabla f\left(x, w_t\right) \end{aligned} \tag{6}$$

$$w_{t+1} = w_t + \lambda \sum_{k=1}^{N} spa\left(a_{k,t}\right)$$

2.3. Bias Correction for Adam-AvgS

The Adam (Adaptive Moment Estimation) [17] algorithm is a method with adaptive learning rate characteristics. This algorithm combines the AdaGrad algorithm and the RMSProp algorithm to solve the problem of sparse gradient optimization while providing a method to reduce noise. But there is still the problem of poor convergence effect. Small batch correction may not be a problem, but when optimizing a large number of gradients, large learning rate interference may occur. Sparsifying gradients results in a large number of gradient iterations.

In order to smoothly perform sparse bias correction, during the process of optimizing sparse model parameter compression, we improved the Adam algorithm to better adapt to the characteristics of sparse gradients. We subtract the gradient and momentum values to more accurately reflect the change in gradient. In the Adam algorithm, the first-order momentum m_t helps to smooth the fluctuations of the gradient, while the second-order momentum v_t is used to adaptively adjust the learning rate. In order to strengthen the connection between the two gradient parameters, we relocate the second-order momentum of this round so that the update of the second-order momentum is related to the parameters of the previous gradient. Such a design facilitates smoother convergence and reduces fluctuations during training.

In addition, in order to further improve the stability of the training process, we further limit the learning rate of adaptive learning. We calculated the sum of previous dynamic learning rates and found their average. By limiting the fluctuation range of the learning rate, we can better control the training process of the model and obtain more stable and reliable training results.

Adam-AvgS algorithm flow is as follows:

- Input: initial parameter θ_0, exponential decay rate β_1, $\beta_2 \in [0, 1]$, n is the number of learning rates participating in the mean, sum is the synthesis of previous learning rates, initial learning rate $\alpha = 0.001$. $\varepsilon = 10^{-8}$.
- Output: Update parameter θ_t.
- Initialization: Random objective function $f(\theta)$.
- $\theta_0, m_0 \leftarrow 0, v_0 \leftarrow 0, t \leftarrow 0, n \leftarrow 1$.
- while θ_t Not converged.
- $t \leftarrow t+1$.
- $g_t \leftarrow \nabla_\theta f_t\left(\theta_{t-1}\right)$ (Update gradient value).
- $m_t \leftarrow \beta_1 \cdot m_{t-1} + \left(1-\beta_1\right) \cdot g_t$ (Update first moment estimate term).
- $v_t \leftarrow \beta_2 v_{t-1} + \left(1-\beta_2\right)\left(g_t - m_t\right)\left(g_t - v_{t-1}\right) + \varepsilon$ (Update the second-order moment estimation term, replacing the second-order momentum with the difference between the gradient and the first-order momentum and the difference between the gradient and the previous round of second-order momentum).

- $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}, \hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}, \eta_t \leftarrow \alpha_t \Big/ \left(\sqrt{\hat{v}_t} + \varepsilon\right), \eta_t \leftarrow \alpha_t \Big/ \left(\sqrt{\hat{v}_t} + \varepsilon\right)$ (Correct the deviation of the first moment, correct the deviation of the second moment Calculate the current learning rate).
- $sum_t \leftarrow sum_{t-1} + \eta_t, S_t \leftarrow sum_t / n$ (Learning rate summation and averaging).
- Update parameters.

Figure 2 shows that: After using Adam-AvgS for learning rate smoothing, the learning rate change trend is more stable than before using the algorithm.

2.4. Quantitative Compression of Downloaded Model Parameters

Previous research has shown that there is too much repeated information in the gradient, and the model training can still be completed even if the gradient is sparsely reduced to one percent of the original, so a 99% sparsification rate will be used below. Although sparsification reduces a lot of communication overhead, the average gradient information uploaded is important information after compression. Compared with the one percent communication cost when uploading, there will be a communication cost several times higher when downloading than when uploading.

The gradient average aggregation with a sparsification rate of 99 will continue to reduce the sparsity rate as the number of nodes increases, which means more communication overhead is added (Table 1).

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

$$W_i = \left\lfloor Z_q + \frac{W_t}{X_{sf}} \right\rfloor \tag{7}$$

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:

$$W_f' = X_{sf}\left(W_i - Z_q\right) \tag{8}$$

3. Experimental Design and Results

Implementing FedSSL in Python using the deep learning framework PyTorch We use ResNet-18 as the default network for the encoder. The predictor is a two-layer multilayer perceptron (MLP). By default, this article trains R = 100 epochs, K = 5 clients, E = 5 local epochs, batch size B = 128, and initial learning rate η = 0.032.

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.

3.1. Model Accuracy

The overall accuracy can be slightly improved under the edge computing framework, but the convergence and improvement are faster on CIFAR-100.

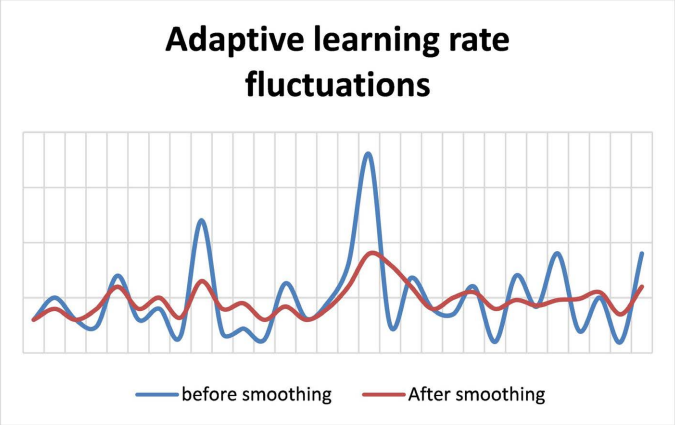

Figure 2. Adam-AvgS fluctuation of learning rate after smoothing.

Table 1. Additional communication overhead caused by gradient aggregation.

Sparsification Rate	Communication Overhead		
	Average Sparsification Rate	Node	Download Cost
99	98.01	2	2x
99	96.05	4	4x
99	92.27	8	7.7x
99	85.14	16	15x

This may be because the CIFAR-100 data set is more complex than CIFAR-10 and requires higher computing costs. And edge servers speed up convergence.

Figure 3 and Figure 4 shows that Compared with the existing federated self-supervision framework (FedBYOL), the new federated self-supervision framework (New FedBYOL), which adds edge servers and uses various communication compression modules, has higher model accuracy and smoother performance in different data sets (Figure 3 CIFAR10, Figure 4 CIFAR100).

3.2. Total Number of Bits

New federated self-supervision-edge computing framework can effectively reduce total communication bits.

Figure 5 shows compared with the existing federated self-supervision framework (FedBYOL), the new federated self-supervision framework (New FedBYOL) that adds edge servers and uses various communication compression modules can reduce the total number of communication bits and has been verified in different data sets.

4. Conclusion and Outlook

In the federated self-supervision framework, adding edge servers can improve the convergence speed of models on complex data sets and reduce the pressure on central servers. The sparse parameter upload compression method and

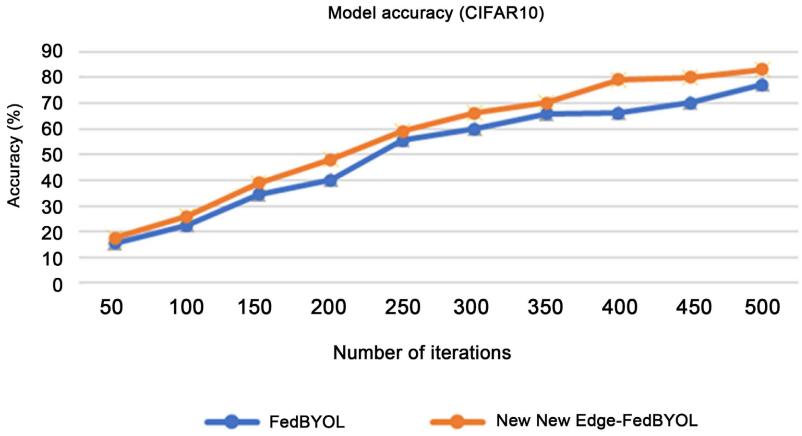

Figure 3. Accuracy of federal self supervised edge computing framework model CIFAR10.

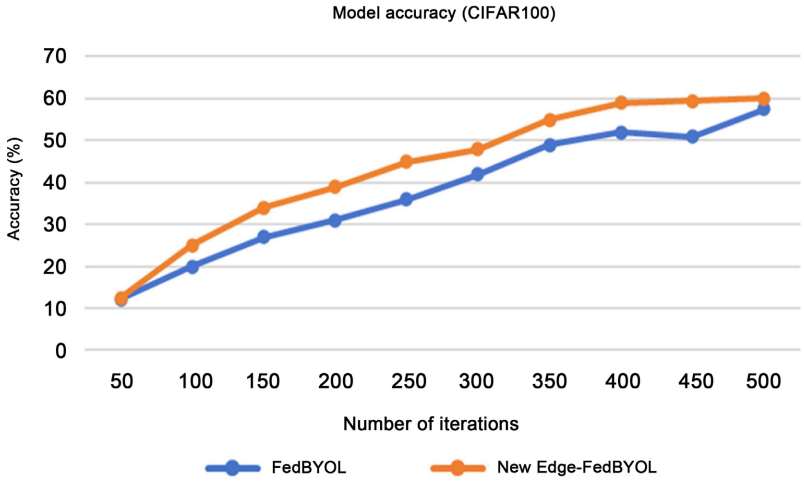

Figure 4. Accuracy of federal self supervised edge computing framework model CIFAR100.

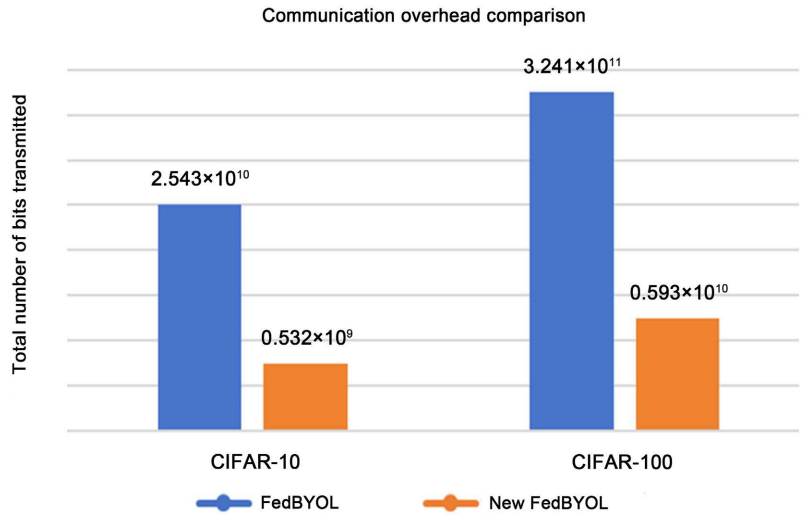

Figure 5. Comparison of communication overhead of federally self supervised edge computing framework under different data sets.

download dynamic quantization compression method reduce the number of bits in a single communication and effectively reduce the total communication overhead. It can be seen that the edge communication optimization of the Edge-federated self-supervised framework is feasible. As more edge computing frameworks are used, improvements to compression algorithms will be a better way to optimize the system.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] McMahan, B., Moore, E., Ramage, D., *et al*. (2017) Communication-Efficient Learning of Deep Networks from Decentralized Data. *Proceedings of the* 20*th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, USA, 20-22 April 2017, 1273-1282.

[2] Liu, X., Zhang, F., Hou, Z., *et al*. (2021) Self-Supervised Learning: Generative or Contrastive. *IEEE Transactions on Knowledge and Data Engineering*, **35**, 857-876. https://doi.org/10.1109/TKDE.2021.3090866

[3] Chen, X. and He, K. (2021) Exploring Simple Siamese Representation Learning. 2021 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Nashville, 20-25 June 2021, 15750-15758. https://doi.org/10.1109/CVPR46437.2021.01549

[4] Fang, P.F., Li, X., Yan, Y., *et al*. (2022) Connecting the Dots in Self-Supervised Learning: A Brief Survey for Beginners. *Journal of Computer Science and Technology*, **37**, 507-526. https://doi.org/10.1007/s11390-022-2158-x

[5] Sun, C., Li, X., Wen, J., Wang, X., Han, Z. and Leung, V.C.M. (2023) Federated Deep Reinforcement Learning for Recommendation-Enabled Edge Caching in Mobile Edge-Cloud Computing Networks. *IEEE Journal on Selected Areas in Communications*, **41**, 690-705. https://doi.org/10.1109/JSAC.2023.3235443

[6] Shi, W.S., Cao, J., Zhang, Q., *et al*. (2016) Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, **3**, 637-646. https://doi.org/10.1109/JIOT.2016.2579198

[7] Dong, Y.M., Zhang, J., Xie, C.Z. and Li, Z.Y. (2024) A Survey of Key Issues in Edge Intelligent Computing under Cloud-Edge-Terminal Architecture: Computing Optimization and Computing Offloading. *Journal of Electronics & Information Technology*, **46**, 765-776.

[8] Alistarh, D., Grubic, D., Li, J., *et al*. (2017) QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *Advances in Neural Information Processing Systems*, **30**, 1707-1718.

[9] Sattler, F., Wiedemann, S., Müller, K.R., *et al*. (2019) Robust and Communication-Efficient Federated Learning from Non-Iid Data. *IEEE Transactions on Neural Networks and Learning Systems*, **31**, 3400-3413. https://doi.org/10.1109/TNNLS.2019.2944481

[10] Msechu, E.J. and Giannakis, G.B. (2011) Sensor-Centric Data Reduction for Estimation with WSNs via Censoring and Quantization. *IEEE Transactions on Signal Processing*, **60**, 400-414. https://doi.org/10.1109/TSP.2011.2171686

[11] Bernstein, J., Wang, Y.X., Azizzadenesheli, K., *et al.* (2018) signSGD: Compressed Optimisation for Non-Convex Problems. *International Conference on Machine Learning*, 560-569.

[12] Qu, Z., Zhou, Z., Cheng, Y., *et al.* (2020) Adaptive Loss-Aware Quantization for Multi-Bit Networks. 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (*CVPR*), Seattle, 13-19 June 2020, 7988-7997. https://doi.org/10.1109/CVPR42600.2020.00801

[13] Li, J., Lyu, L., Iso, D., *et al.* (2022) Mocosfl: Enabling Cross-Client Collaborative Self-Supervised Learning. *The Eleventh International Conference on Learning Representations*, New Orleans, 19-24 June 2022.

[14] Wang, R., Hu, Y., Chen, Z., *et al.* (2024) TabFedSL: A Self-Supervised Approach to Labeling Tabular Data in Federated Learning Environments. *Mathematics*, **12**, Article No. 1158. https://doi.org/10.3390/math12081158

[15] Shi, S., Wang, Q., Zhao, K., *et al.* (2019) A Distributed Synchronous SGD Algorithm with Global Top-k Sparsification for Low Bandwidth Networks. 2019 *IEEE* 39*th International Conference on Distributed Computing Systems* (*ICDCS*), Dallas, 07-10 July 2019, 2238-2247. https://doi.org/10.1109/ICDCS.2019.00220

[16] Chen, C.Y., Choi, J., Brand, D., *et al.* (2018) Adacomp: Adaptive Residual Gradient Compression for Data-Parallel Distributed Training. *Proceedings of the AAAI Conference on Artificial Intelligence*. https://doi.org/10.1609/aaai.v32i1.11728

[17] Diederik, P.K. (2014) Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (*ICLR*), San Diego, 7-9 May 2015.