A Simple Federated Learning-Based Scheme for Security Enhancement Over Internet of Medical Things

Zhiang Xu, Yijia Guo, Chinmay Chakraborty, *Senior Member, IEEE*, Qiaozhi Hua, Shengbo Chen, and Keping Yu, *Member, IEEE*

***Abstract*—Nowadays, Federated Learning (FL) over Internet of Medical Things (IoMT) devices has become a current research hotspot. As a new architecture, FL can well protect the data privacy of IoMT devices, but the security of neural network model transmission can not be guaranteed. On the other hand, the sizes of current popular neural network models are usually relatively extensive, and how to deploy them on the IoMT devices has become a challenge. One promising approach to these problems is to reduce the network scale by quantizing the parameters of the neural networks, which can greatly improve the security of data transmission and reduce the transmission cost. In the previous literature, the fixed-point quantizer with stochastic rounding has been shown to have better performance than other quantization methods. However, how to design such quantizer to achieve the minimum square quantization error is still unknown. In addition, how to apply this quantizer in the FL framework also needs investigation. To address these questions, in this paper, we propose FEDMSQE – Federated Learning with Minimum Square Quantization Error, that achieves the smallest quantization error for each individual client in the FL setting. Through numerical experiments in both single-node and FL scenarios, we prove that our proposed algorithm can achieve higher accuracy and lower quantization error than other quantization methods.**

***Index Terms*—Internet of medical things, federated learning, quantization, security.**

Manuscript received 14 January 2022; revised 3 June 2022; accepted 26 June 2022. Date of publication 30 June 2022; date of current version 6 February 2023. This work was supported in part by the Japan Society for the Promotion of Science (JSPS) Grants-in-Aid for Scientific Research (KAKENHI) under Grants JP18K18044 and JP21K17736, in part by Hubei Natural Science Foundation under Grant 2021CFB156, in part by the National Natural Science Foundation of China under Grant 62102133, in part by Kaifeng Science and Technology Major Project under Grant 21ZD011, in part by Ji'An Finance and Science Foundation under Grant [2022]4. *(Corresponding authors: Shengbo Chen; Qiaozhi Hua.)*

Zhiang Xu and Shengbo Chen are with the Henan Key Laboratory of Big Data Analysis and Processing, the School of Computer and Information Engineering, Henan University, Kaifeng 475004, China (e-mail: 1193915738@qq.com; ccb02kingdom@gmail.com).

Yijia Guo is with the School of Automation Science and Electrical Engineering, Beihang University, China (e-mail: 3133512786@qq.com).

Chinmay Chakraborty is with the Birla Institute of Technology, Mesra, Ranchi 835215, India (e-mail: cchakrabarty@bitmesra.ac.in).

Qiaozhi Hua is with the Computer School, Hubei University of Arts and Science, Xiangyang 441000, China (e-mail: 11722@hbuas.edu.cn).

Keping Yu is with the Graduate School of Science and Engineering, Hosei University, Tokyo 184-8584, Japan, and also with the RIKEN Center for Advanced Intelligence Project, RIKEN, Tokyo 103-0027, Japan (e-mail: keping.yu@ieee.org).

Digital Object Identifier 10.1109/JBHI.2022.3187471

I. Introduction

FEDERATED Learning (FL) is a variant of ML, in which the neural network model itself is deployed in various devices for local data training instead of collecting the data to a central server. The models trained on all separate devices are aggregated at the central server to obtain a global model, which is then distributed to all the clients for the next training round. In this way, federated learning can help protect data privacy without exposing sensitive information to potential attackers.

While popular deep neural networks such as VGG, ResNet, etc., have shown great performance in FL, however, to deploy such neural networks on the Internet of Medical Things (IoMT) devices is very difficult. The reason is that the scales of these neural networks are usually extensive, and the corresponding computation always consumes much power and memory, which cannot be met by the IoMT devices. In addition, some IoMT devices, like those equipped on wearable devices, etc., have low transmission latency requirement, while the large-size parameters of neural networks are the obstacle to meet this low latency requirement. On the other hand, the traditional FL structure requires to upload the neural network model, which can protect the privacy of user data in some sense. However, some recent studies [1]–[9] have shown that there is still risk of privacy leakage when models are intercepted. For example, eavesdroppers can obtain some useful information of the data, or even recover the data through model inversion. This kind of privacy violation cannot be tolerated in the IoMT field [10], and the current FL privacy protection mechanism cannot deal with this problem effectively. Therefore, while maintaining the performance, reducing the size of the neural network and protecting data privacy has become challenges for FL over IoMT devices.

In order to solve this problem, we take a quantization method to protect the security of model transmission in FL. Quantization has been proven to be an effective method to enhance the security of data transmission, and is widely used in various fields such as signal processing, data compression, and signal conversion. For example, a 64-bit floating-point number can be quantized using 8 bits, which can reduce the storage size to 1/8. On the other hand, it can enhance the security during transmission, since

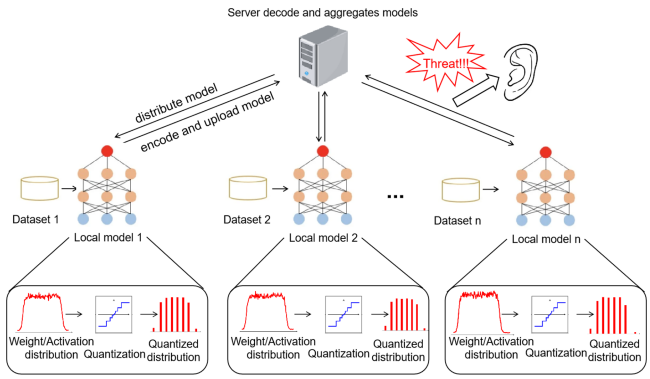


Fig. 1. Federated learning quantization process.

the quantized model deviates from the original one due to the "noises" introduced by the quantization. In particular, in the FL framework, we quantize and encode the model parameters before uploading to the server, and then decode on the server to protect the model transfer. In order to better illustrate our idea, we draw a diagram of the quantization process in FL, as shown in Fig. 1. It is of great interest to apply quantization for FL over IoMT devices.

Despite of much benefit brought about by quantization, it also has its downside. Due to the less bits used, the quantization process will introduce model deviation, known as quantization error. The occurrence of such errors in deep neural networks can lead to negative consequences, such as performance decrease. It is well known that quantizer with smaller square quantization error usually yields a better performance. For the currently popular quantizers, there are mainly classified into two categories: uniform quantizer and non-uniform quantizer. According to the different clipping range of the input vector, the uniform quantizer can be further divided to symmetric quantizer and asymmetric quantizer. The symmetric quantizer uses symmetric limiting range, which is easy to implement, but the performance of asymmetric quantizer using asymmetric limiting range is always better. For the non-uniform quantizer, although it can better capture the information of the input vector, it requires more computations. The uniform quantizer has been widely used in FL over IoMT, but its quantization error is usually large, resulting in an excessive decrease in system performance. Therefore, there is an urgent need for a quantizer design with small quantization error and high performance.

In the previous literature, the authors in [11] prove that the fixed-point quantizer with stochastic rounding has better performance than other quantization methods. However, how to design such quantizer with minimum square quantization error is still unknown. Furthermore, how to apply this quantizer in the FL framework also requires further investigation. In this paper, we consider the fixed-point quantizer with stochastic rounding, and propose FEDMSQE – Federated Learning with Minimum Square Quantization Error. In particular, in FEDMSQE, we design an MSQE quantizer that achieves the best square quantization error for each individual client, and the quantization is heterogeneous from client to client. In addition, the extra overhead occurred during the model uploading in FL is very limited.

Quantization can compress the size of the model, reduce storage space, improve the security of data transmission, and help better deploy neural networks on the devices. However, none of the existing quantizers can achieve the smallest quantization error, resulting in a poor performance. Compared with other quantizers, our designed MSQE quantizer has the smallest quantization error, and the simulations show that it has better performance than other quantizers, especially in the low-precision quantization scenarios. Therefore, the MSQE quantizer has significant potential in the IoMT field.

It is worth pointing out that our MSQE quantizer can be applied for any deep neural network on any single device. In other words, the quantizer we proposed is limited to FL. Throughout the paper, we assume quantization is only executed at the clients, and the server is assumed to run with full precision.

We summarize the main contributions of this paper as following.

1) To enhance the FL security, we design an MSQE quantizer for the smallest quantization error among all quantizers using the fixed-point quantization method with stochastic rounding.
2) We propose FEDMSQE, which applies the MSQE quantizer on each client, and orchestrates the local model upload with limited overhead increase.
3) Through extensive experiments on real-world classification tasks, we show that FEDMSQE outperforms other quantization methods.

The structure of the paper is as follows: The Section II mainly introduces related works. The new quantizer we proposed and its derivation are demonstrated in Section III. The FEDMSQE algorithm, is detailed in Section V. Numerical experiment results are reported in Section VI. Finally, Section VII concludes our paper.

II. Related Works

Recently, quantization of deep neural networks have become a hot research area. Various different quantizers have been proposed to reduce the scale of neural networks. In this section, we will discuss some related literature on quantizer design and quantization used in federated learning.

A. Security and Quantizer Design

The uniform quantizer [12]–[17] is one popular quantizer type that divides the input vector into equal intervals. Due to the difference of the vector limit range, it can be classified into two categories, i.e., symmetric quantization and asymmetric quantization. Symmetrical quantization uses a symmetrical clipping range, which is simple to calculate and easy to implement. While asymmetric quantization is more complicated, its performance is usually better than symmetric quantization. Although uniform quantization is widely used, the abnormal value of the vector will increase its limit range, resulting in a performance decrease. Therefore, the i-th maximum/minimum value [18] is proposed as the limit range. Another method is KL divergence quantization [19], [20]. This method is to select an appropriate clipping range, which can not only ensure that not too much data

is truncated, but also achieve a higher accuracy. On the other hand, non-uniform quantizers [21]–[26] adjust the quantization interval according to the input to better capture the information of the input vector, which requires more computation resources consequently.

Another way to classify quantizer is to divide to dynamic quantizer [27], [28] and static quantizer [29], [30], [31]. In dynamic quantization [32], the range is dynamically calculated for each activation map at runtime. The dynamic quantizer needs to count the range of the input vector in real time, so the overhead is quite large. For the static quantizer, the clipping range is pre-calculated and keeps static during the whole process. Compared with the dynamic quantizer, the static quantizer only generates a small computational overhead. Thus, the static quantizer is more popular.

There is also discussion on designing a binary weight quantizer [33]–[35]. That is, in the forward and backward propagation of neural network, binary weights are used instead of full-precision floating-point numbers, eliminating about 2/3 of the multiplication, so the speed increases by three times during training. [36] proposes a quantization method called BWNH, which only quantizes the weight. The main contribution of the algorithm is to solve the quantization problem from the perspective of hashing, and propose a layer-by-layer alternate update method, which has significant improvement in performance compared to the previous binary weight quantization method. The ternary weight quantizer [37], [38] is more expressive than the binary weight quantizer, so the performance of the ternary weight quantizer is preferable to that of the binary weight quantizer.

For other quantization research, [39] proposes a quantization method (QNN), a neural network with very low precision (such as 1 b) weight and activation at runtime. During the training period, the quantized weights and activation values are used to calculate parameter gradients. During the forward pass, QNN greatly reduces the memory size and access, and replaces most arithmetic operations with bitwise operations. Thus, the expected power consumption is greatly reduced. Meanwhile, in order to solve the problem that the lower the quantization level, the lower the model recognition rate, [40] introduces a new method to estimate and expand the task loss gradient of the quantizer step size of each weight and activation layer. And the experiment and analysis on ImageNet prove the effectiveness of the proposed method.

Compressed communication [41]–[44] is also a research hotspot. [41] proposed the SPARQ-SGD algorithm for decentralized training of large-scale machine learning models on graphs with n nodes, where the communication efficiency is achieved by the compressed exchange of local model parameters between adjacent nodes, triggered only when an event (local computable condition) is satisfied. Specifically, in SPARQ-SGD, each node takes a fixed number of local gradient steps, and then checks whether the model parameters have changed significantly from the last update; only when the change exceeds a certain threshold (by design criteria) specified, it uses quantization and sparsification to compress its local model parameters and communicate them to its neighbors. The algorithm demonstrates significant savings in communication bits over the state-of-the-art.

B. FL Quantization

There is plenty of literature on the use of quantization in FL, such as [45]–[48]. The authors of [49] propose that the clients of federated learning are heterogeneous and support different levels of quantification precision. This quantization heterogeneity raises a question: how to optimize and aggregate them on the server? In order to solve this problem, the authors propose the federated learning algorithm with heterogeneous quantization(FedHQ), which assigns different aggregation weights to different clients by minimizing the upper limit of the convergence rate as a function of the heterogeneous quantization error of all clients. Experiments show that the effect of FedHQ algorithm is better than ordinary FedAvg [50] with standard equal weights.

The high complexity of communication is also a big problem for federated learning. In order to solve this problem, [51] applies quantization to model parameters and gradients, and proposes the following algorithms: 1) a low-precision algorithm AsyLPG with asynchronous parallelism; 2) integrating gradient sparsification with double quantization and develop Sparse-AsyLPG. These two algorithms can effectively save transmission bits without reducing performance.

Each user in FL needs to effectively transmit its learning model on an uplink channel with limited throughput. [52] proposes the use of quantitative theory tools to meet this challenge. Combining the general vector quantization method with FL produces a decentralized training system that is both effective and feasible. At the same time, the theoretical performance guarantee of the system is derived. Numerical results show that, compared with the previously proposed quantization method, this scheme obtains a significant performance gain on FL.

In order to solve the communication bottleneck and scalability issues of federated learning, [53] proposes FedPAQ, which is an efficient communication federation learning method with periodic averaging and quantification. FedPAQ relies on three key features: (1) periodic averaging, where the model is updated locally on the device and only periodically averaged on the server; (2) part of the equipment participates, and only a small part of the equipment participates in each round of training; (3) quantizing message passing, where edge nodes quantize their updates before uploading to the parameter server. These functions solve the communication and scalability challenges in federated learning.

Despite of a lot of prior work on quantization used in FL, all existing quantizers cannot achieve the smallest quantization error. The quantizer we designed is dedicated to reducing quantization errors, which enhancing the performance of deep neural networks.

III. MSQE Quantizer

In this section, we will introduce the MSQE quantizer. We assume that b quantization bits are used for each number, where each quantization bit can be "0" or "1". Therefore, there are a total of 2^b combinations, which are denoted as

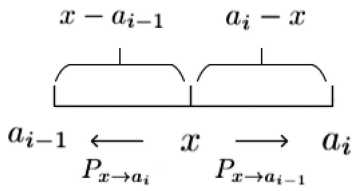


Fig. 2. Illustration of quantization.

$D_0, D_1, \ldots, D_{2^b-1}$, e.g., $D_0 =$ "$0 \cdots 0$". Each $D_i \in \{D_i\}_{i=0}^{2^b-1}$ can represent one real value, which we denote as $a_i \in \{a_i\}_{i=0}^{2^b-1}$. In other words, there is a 1-1 mapping between $\{D_i\}_{i=0}^{2^b-1}$ and $\{a_i\}_{i=0}^{2^b-1}$, which can be depicted by $D_i = f(a_i)$.

As stated before, we consider the fixed-point quantization with stochastic rounding, where any number x is rounded up or down at random such that $\mathbb{E}[Q(x)] = x$, where $Q(\cdot)$ is the quantization function. It is worth pointing out that given any vector $\vec{x}$ to be quantized, this assumption requires $a_0 \leq x \leq a_{2^b-1}$ for any $x \in \vec{x}$, that is, $a_0 = \min(\vec{x})$ and $a_{2^b-1} = \max(\vec{x})$. Otherwise, it will violate the assumption $\mathbb{E}[Q(x)] = x$.

Note that any specific design is to determine the quantization boundaries $\{a_i\}_{i=0}^{2^b-1}$. For example, the uniform quantizer is shown below.

Example: UniformQuant. An input vector $\vec{x}$ is quantized using b bits as:

$$a_0 = X_{min}, a_{2^b-1} = X_{max},$$
$$a_i = X_{min} + \frac{i\,(X_{max} - X_{min})}{2^b - 1},$$

where $[X_{min}, X_{max}]$ is defined as the minimum and maximum values of the input vector.

Now, we are ready to present the MSQE quantizer design. First, the probability of quantizing a number x to the left or right boundary is related to the distance between it and these two boundaries. Suppose x is between a_{i-1} and a_i, shown in Fig. 2.

We can derive the probability of quantizing x to the left boundary $P_{x \to a_{i-1}}$ and to the right $P_{x \to a_i}$ boundary, as shown in (1) and (2).

$$P_{x \to a_{i-1}} = \frac{a_i - x}{a_i - a_{i-1}}, \tag{1}$$

$$P_{x \to a_i} = \frac{x - a_{i-1}}{a_i - a_{i-1}}. \tag{2}$$

Assume all elements in the model from a vector $\vec{x}$, which has a probability distribution $f(x)$. Given a number x between a_{i-1} and a_i, the square quantization error for x is given by

$$SQE(x) = (x - a_{i-1})^2 \frac{a_i - x}{a_i - a_{i-1}} + (a_i - x)^2 \frac{x - a_{i-1}}{a_i - a_{i-1}}$$
$$= -x^2 + (a_{i-1} + a_i)\, x - a_{i-1} a_i. \tag{3}$$

Thus, the SQE between a_{i-1} and a_i can be written as

$$SQE\left([a_{i-1}, a_i)\right) = \int_{a_{i-1}}^{a_i} \left[-x^2 + (a_{i-1} + a_i)x - a_{i-1}a_i\right] f(x)dx, \tag{4}$$

where the integral is taken with respect to all values within the range $[a_{i-1}, a_i)$.

And the total SQE, which we denote as σ^2, for the model vector can be depicted as

$$\sigma^2 = \sum_{i=1}^{2^b-1} \int_{a_{i-1}}^{a_i} \left[-x^2 + (a_{i-1} + a_i)x - a_{i-1}a_i\right] f(x)dx. \tag{5}$$

Then the minimum square quantization error problem can be formulated as

$$\min_{\{a_i\}_{i=0}^{2^b-1}} \quad \sigma^2,$$
$$s.t. \quad a_0 = \min(\vec{x}), a_{2^b-1} = \max(\vec{x}). \tag{6}$$

To solve (6), we consider its partial derivative over a_i for $i \in [1, 2 \cdots, 2^b - 2]$, which we have

$$\begin{aligned}\frac{\delta\sigma^2}{\delta a_i} &= \int_{a_{i-1}}^{a_i} (x - a_{i-1}) f(x)dx + a_i^2 f(a_i) \\ &\quad + \int_{a_i}^{a_{i+1}} (x - a_{i+1}) f(x)dx - a_i^2 f(a_i) \\ &= \int_{a_{i-1}}^{a_{i+1}} x f(x)dx - a_{i-1} \int_{a_{i-1}}^{a_i} f(x)dx \\ &\quad - a_{i+1} \int_{a_i}^{a_{i+1}} f(x)dx. \end{aligned} \tag{7}$$

We denote $Prob_i \triangleq \int_{a_{i-1}}^{a_i} f(x)dx$. When $\frac{\delta\sigma^2}{\delta a_i} = 0$, we have

$$\int_{a_{i-1}}^{a_{i+1}} x f(x)dx = a_{i-1} Prob_i + a_{i+1} Prob_{i+1}. \tag{8}$$

In (8), the left hand side means the expectation of all numbers in the interval $[a_{i-1}, a_{i+1}]$, while $Prob_i$ and $Prob_{i+1}$ in the the right hand side implies the probability in the interval $[a_{i-1}, a_i)$, and $[a_i, a_{i+1})$, respectively. We also note that

$$Prob_i + Prob_{i+1} = \int_{a_{i-1}}^{a_{i+1}} f(x)dx, \tag{9}$$

which is known given a_{i-1} and a_{i+1}.

Thus we have

$$Prob_i = \frac{a_{i+1} \int_{a_{i-1}}^{a_{i+1}} f(x)dx - \int_{a_{i-1}}^{a_{i+1}} x f(x)dx}{a_{i+1} - a_{i-1}}. \tag{10}$$

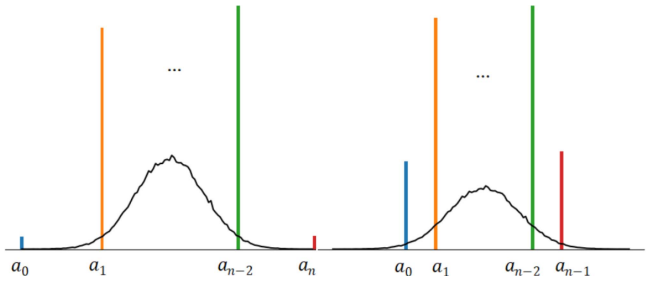
Then we can derive a_i from (10).

As a result, we can derive $\{a_i\}_{i=0}^{2^b-1}$ using iteration. Note that the mathematical deduction above considers a continuous case and uses the integral, while the actual vector elements are discrete. We provide the detailed algorithm given in Algorithm 1.

For the Algorithm 1, the quantization boundaries are initialized using lines 3-7. Line 12 is to find the vector x_c in the vector $\hat{x}$ that is greater or equal than a_{i-1}, and less or equal than a_{i+1}. Line 13 corresponds to the numerator in (10), that is, multiplying the number of x_c by a_{i+1}, and then subtracting the sum of x_c. By dividing by $(a_{i+1} - a_{i-1})$ and rounding down, we obtain idx, which corresponds to $Prob_i$ in (10). Finally,

Algorithm 1: MSQE Quantizer.

Input: parameter vector $\vec{x}$, quantization bits b;
Output: MSQE quantization boundaries $\{a_i\}_{i=0}^{2^b-1}$;
$a_0 = \min(\vec{x})$;
$a_{2^b-1} = \max(\vec{x})$;
for $i = 1$ ***to*** $2^b - 2$ **do**
  $a_i = a_0 + i * (a_{2^b-1} - a_0)/(2^b - 1)$; //initialization
end
$\hat{x} \leftarrow$ Sort $\vec{x}$ from smallest to largest;
while *True* **do**
  $\{\hat{a}_i\}_{i=0}^{2^b-1} = \{a_i\}_{i=0}^{2^b-1}$;
  for $i = 1$ ***to*** $2^b - 2$ **do**
    $x\_c \leftarrow find(a_{i-1} \le \hat{x} \le a_{i+1})$;
    $t = num(x\_c) * a_{i+1} - sum(x\_c)$;
    $idx = \lfloor t/(a_{i+1} - a_{i-1}) \rfloor$;
    $a_i \leftarrow$ The idx-th parameter of x_c;
  end
  if $\{\hat{a}_i\}_{i=0}^{2^b-1} = \{a_i\}_{i=0}^{2^b-1}$ **then**
    break;
  end
end
return $\{a_i\}_{i=0}^{2^b-1}$;



(a) Non-clipping (b) Clipping

Fig. 3. Clipping and non-clipping cases.

we get a_i using the index idx from the vector x_c. By iterating these steps, the MSQE quantization boundaries $\{a_i\}_{i=0}^{2^b-1}$ can be obtained.

IV. Parameter Clipping

It is difficult to avoid outliers for neural network parameters, and the outliers may significantly increase the quantization error and affect the performance of the system. By clipping the outliers, the quantization error can be greatly reduced. Therefore, it is of great interest to facilitate clipping during quantization. Based on the previous mathematical analysis, we further design an algorithm to find the optimal clipping values, as well as the quantization boundaries, so that the minimum mean square quantization error can be reached.

Note that the difference from the previous algorithm is that both a_0 and a_{2^b-1} are tunable, that is, they are not equal to the min/max values of the parameters respectively. An example is shown in Fig. 3(a) and (b). On contrast to the non-clipping case in Fig. 3(a), both a_0 and a_{2^b-1} have clipped some outliers in Fig. 3(b), so that the quantization error can be further reduced.

We denote l as the leftmost value of the parameter, r as the rightmost value of the parameter. Then, the mean square error γ^2 is shown in (11).

$$\gamma^2 = \int_l^{a_0} (a_0 - x)^2 f(x)dx + \int_{a_{2^b-1}}^{r} (a_{2^b-1} - x)^2 f(x)dx + \sum_{i=1}^{2^b-1} \int_{a_{i-1}}^{a_i} [-x^2 + (a_{i-1} + a_i)x - a_{i-1}a_i] f(x)dx. \tag{11}$$

Next, the minimum mean square error problem can be expressed as:

$$\min_{[l,r]} \quad \gamma^2, \qquad s.t. \quad l = \min(\vec{x}), r = \max(\vec{x}). \tag{12}$$

To find the partial derivative of (11) with respect to a_0, we have

$$\begin{aligned} \frac{\delta\gamma^2}{\delta a_0} &= \int_l^{a_0} 2(a_0 - x)f(x)dx + \int_{a_0}^{a_1} (x - a_1)f(x)dx \\ &= \int_l^{a_0} 2a_0 f(x)dx - \int_l^{a_0} 2xf(x)dx \\ &\quad + \int_{a_0}^{a_1} xf(x)dx - a_1 \int_{a_0}^{a1} f(x)dx. \end{aligned} \tag{13}$$

We can obtain the value of a_0 by letting (13) equal 0.

Likewise, for the right side, we take the partial derivative of a_{2^b-1} with respect to (11),

$$\begin{aligned} \frac{\delta\gamma^2}{\delta a_{2^b-1}} &= \int_{a_{2^b-1}}^{r} 2\left(a_{2^b-1} - x\right) f(x)dx \\ &\quad + \int_{a_{2^b-1}}^{a_{2^b-1}} \left(x - a_{2^b-2}\right) f(x)dx \\ &= \int_{a_{2^b-1}}^{r} 2a_{2^b-1} f(x)dx - \int_{a_{2^b-1}}^{r} 2xf(x)dx \\ &\quad + \int_{a_{2^b-2}}^{a_{2^b-1}} xf(x)dx - a_{2^b-2} \int_{a_{2^b-2}}^{a_{2^b-1}} f(x)dx. \end{aligned} \tag{14}$$

We can find the value of a_{2^b-1} when (14) equals 0.

So far, we have obtained the values of a_0 and a_{2^b-1} that can minimize the mean squared error at both ends of the parameter. Then we update the other quantization intervals with a_0 and a_{2^b-1} as the two ends of the quantization interval. By iterating these steps, the MSQE quantization boundaries $\{a_i\}_{i=0}^{2^b-1}$ can be obtained.

V. FedMSQE Algorithm

In this section, we describe the FedMSQE algorithm. As stated before, we consider the quantization of the model updates for each IoMT device, and we assume that the server is with full precision. This assumption is reasonable because IoMT devices only have limited resources, e.g., transmit power, memory, while the parameter server is much more powerful regarding the computation and communication capabilities.

A. Preliminaries

Assuming there are l clients in federated learning, we consider the standard FL problem

$$\min_{x\in\mathbb{R}^d} F(x) = \min_{x\in\mathbb{R}^d} \sum_{i=1}^{l} \frac{n_i}{n} F_i(x), \tag{15}$$

where $F_i(x) := \mathbb{E}_{\xi_i}[F_i(x,\xi_i)]$ is the differentiable loss function of client i for model x and data sample ξ_i, and n_i is the dataset size of client i with $\sum_i n_i = n$. With a slight abuse of notation, we denote $\nabla F_i(x) = \nabla F_i(x,\xi_i)$, which is a stochastic gradient of F_i.

To solve the problem (15), FEDAVG [50] was proposed. It works as follows: the server randomly chooses a client subset, and broadcasts them a global model. After receiving the global model, the clients use local data to train the model for several iterations. Clients then upload the models to the server, where the server updates the global model by aggregating the models. After T rounds, the server obtains the final global model.

B. FEDMSQE Algorithm

We present the FEDMSQE algorithm in Algorithm 2. The specific algorithm process is as follows: at the beginning of each round of $t \in [1, 2, \ldots, T]$, a subset of all clients S_t is select. For any client $i \in S_t$, it receives the server's global model x^{t-1}, and updates the model by running SGD K times locally to obtain a new model x_i^t. We then calculate the MSQE quantization boundaries $\{a_{i,j}^t\}_{j=0}^{2^b-1}$, and apply the stochastic rounding $Q(\cdot)$ for x_i^t, which yields a quantized model $\widetilde{x}_i^t$. Note that each element in the quantized model $\widetilde{x}_i^t$ is one of the quantization boundaries $\{a_{i,j}^t\}_{j=0}^{2^b-1}$. Recall that f_i is the 1-1 mapping between $\{D_j\}_{j=0}^{2^b-1}$ and $\{a_{i,j}\}_{j=0}^{2^b-1}$, i.e., $D_j = f_i(a_{i,j})$, which is heterogeneous between different clients. Therefore, we can encode each element of the quantized model $\widetilde{x}_i^t$ to $D_j \in \{D_j\}_{j=0}^{2^b-1}$, that is, applying f_i on $\widetilde{x}_i^t$. We then upload $f_i(\widetilde{x}_i^t)$ and $\{a_{i,j}^t\}_{j=0}^{2^b-1}$ to the server. After the server receives this information, it decodes $\widetilde{x}_i^t$ by applying the f_i^{-1} function. Finally, the server aggregates all the decoded models $\widetilde{x}_i^t, i \in S_t$ to obtain a new global model.

In Algorithm 2, line 12 implies that each client i calculates its own MSQE quantization boundaries $\{a_{i,j}^t\}_{j=0}^{2^b-1}$ using Algorithm 1. Note that the quantization boundaries for different clients are heterogeneous. Line 13 means that each client i generates a quantized model from its original model using its own MSQE quantizer. In line 14, upload the quantization boundaries $\{a_{i,j}^t\}_{j=0}^{2^b-1}$ and $f_i(\widetilde{x}_i^t)$ to the server. On the server side, the received models are averaged and then distributed to the selected clients.

Algorithm 2: The FEDMSQE Algorithm.

1 Input: Initial model x^0, quantization bits b;
2 Output: Final model x^t;
3 **for** $t = 1$ **to** T **do**
4 Randomly select a client subset S_t from all clients;
5 The server broadcasts parameter vector x^{t-1} to all clients in S_t;
6 **for** *each client* $i \in S_t$ **do**
7 $x_{i,0}^{t-1} \leftarrow x^{t-1}$;
8 **for** $\tau = 0$ **to** $K-1$ **do**
9 $x_{i,\tau+1}^{t-1} = x_{i,\tau}^{t-1} - \eta_t \nabla f_i(x_{i,\tau}^{t-1})$;
10 **end**
11 $x_i^t \leftarrow x_{i,K}^{t-1}$;
12 $\{a_{i,j}^t\}_{j=0}^{2^b-1} \leftarrow$ ***MSQE***(x_i^t, b) in Alg. 1;
13 $\widetilde{x}_i^t \leftarrow Q(x_i^t, \{a_{i,j}^t\}_{j=0}^{2^b-1})$;
14 Uploads $\{a_{i,j}^t\}_{j=0}^{2^b-1}$ and $f_i(\widetilde{x}_i^t)$ to server;
15 **end**
16 Server decodes $\widetilde{x}_i^t, i \in S_t$ from $f_i(\widetilde{x}_i^t), i \in S_t$;
17 Server updates $x^t \leftarrow \frac{1}{|S_t|}\sum_{i=1}^{|S_t|} \widetilde{x}_i^t$;
18 **end**
19 return x^t;

VI. EXPERIMENT

A. Experiment Setup

Models and Datasets: We use the MNIST dataset and CIFAR10 dataset to validate the performance of the MSQE quantizer and our FEDMSQE algorithm.

For MNIST, there are 60,000 training samples and 10,000 testing samples. We first test the MSQE quantizer on the single-node case, where the neural network model is a simple multi-layer perceptron with two hidden layers. Each layer is with 200 units, activated by RELU. We then test FEDMSQE algorithm in FL. Assuming that there are $l = 10$ clients in FL, and each client is allocated 6000 training samples and 1000 testing samples. The neural network model we use is the same as the single-node case.

For CIFAR10, there are 50,000 training samples and 10,000 testing samples. For the federated learning case, we assume a total of 10 clients, and each client is allocated with 5000 training samples and 1000 testing samples. Both single-node and federated learning cases, ResNet18 is used, which is the same as the model of [54].

(Hyper)parameters: There are some notable (hyper)parameters in the experiment. We define T as the number of rounds of communication. K and B represent the number of training iteration and batch size in each round, respectively. η is defined as the learning rate(starting from the 11th round, the decay is 0.008% per round), and m is the momentum of the optimizer. λ is the weight attenuation of the optimizer. We set $K = 1$, $\eta = 0.02$, $B = 20$, $\lambda = 0.0005$ and $m = 0.5$ for MNIST and CIFAR10. For single-node, we test the accuracy of the quantized model and the quantization error in each round. For federated learning, we record the accuracy of the global model and average the quantization error of the clients in each round. We tested the quantization bits of $b = 3$ and $b = 5$ for the single-node and FL, which is a typical situation for IoMT devices.

State-of-art: In addition to the MSQE quantizer, we also implemented the uniform quantizer and the state-of-art APoT quantizer [55] as baselines for comparison. Then we compare

both with MSQE quantizer under the single-node and FL scenarios.

For the uniform quantizer, we take the maximum and minimum parameters as the two ends of the quantization boundary, and then uniformly take the quantization boundaries.

We have already explained this in Section III.

For the APoT quantizer, it is proposed to solve the rigid resolution problem of PoT. The core idea of PoT (as shown in (16)) is that the closer to parameter 0, the more quantization boundaries are used, which will cause too many quantization boundaries near parameter 0 and too few quantization boundaries at both ends of the parameter, i.e., rigid resolution problem.

$$\{a_i\}_{i=0}^{2^b-1} = \alpha \times \left\{0, \pm 2^{-2^{b-1}+1}, \pm 2^{-2^{b-1}+2}, \ldots, \pm 2^{-1}\right\}. \tag{16}$$

where $\{a_i\}_{i=0}^{2^b-1}$ is quantization boundary, b is quantization bits, α ensure that the quantization level and the parameter range are consistent.

APoT (as shown in (17)) proposes the method of adding two PoTs. This can reduce the quantization boundaries near parameter 0 and increase the quantization boundaries at both ends, i.e.,

$$\{a_i\}_{i=0}^{2^b-1} = \gamma \times \left\{\sum_{i=0}^{n-1} p_i\right\}, \tag{17}$$

where $p_i \in \{0, \frac{1}{2^i}, \frac{1}{2^{i+n}}, \ldots, \frac{1}{2^{i+(2^k-2)n}}\}$, and γ ensures that the quantization level and the parameter range are consistent. k is called the base bit-width, which is the bit-width for each additive term, and n is the number of additive terms. When the bit-width b and the base bit-width k are set, n can be calculated by $n = \frac{b}{k}$.

The MSQE quantizer we designed has the following advantages over other quantizers. First, based on the rigid mathematical derivation, it can be proved that the MSQE quantizer achieves the minimum quantization error. Second, for the uniform quantizer and APoT quantizer, their quantization boundaries are relatively fixed, i.e., do not change with parameter distribution. But the quantization boundaries of the MSQE quantizer are adaptive to the parameter distribution, which is more flexible and thus brings about performance advantage.

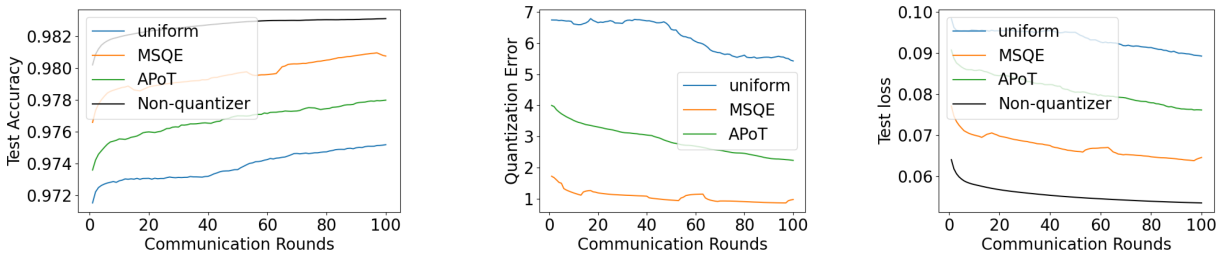
We test the computation overhead of the proposed MSQE quantizer relative to other methods. In terms of computational overhead, we test the runtime of each quantizer, where we use a PowerEdge R740 server. For the MNIST dataset, it on average takes 20.3 seconds to run a round without quantization, 21.5 seconds for the uniform quantizer, 23.32 seconds for the MSQE quantizer, and 22.44 seconds for the APoT quantizer in the case of 3-bit quantization. In the case of 5-bit quantization, the uniform quantizer on average takes 28.9 seconds, the MSQE quantizer takes 39.3 seconds, and the APoT quantizer takes 29.13 seconds. Therefore, the computational runtime of MSQE quantizer is the same as other quantizer for low precision case, and about 30% higher for the high precision case. On the other hand, we also calculate the communication overhead. For our scheme, each client needs to uploads the quantized neural network parameters and the quantization boundaries with full precision to the server. Thus, the overhead is the extra boundaries. We take the model on MNIST dataset as an example. The total number of parameters is 199210 within 6 layers, occupying 199210*32(bit)/8/1024=778.16 KB storage space. For the 3-bit quantization, the quantized model occupies a total of 199210*3(bit)/8/1024=72.95 KB storage space. For our scheme, a total of 6*8=48 boundaries with full precision need to be uploaded, which is 48*32(bit)/8/1024=0.19 KB. Thus, the total overhead for our scheme is 0.2%. Similarly, for the 5-bit quantization, the storage space for the model is 199210*5(bits)/8/1024=121.58 KB, and 0.75 KB for the boundaries, which results in a 0.6% overhead. Therefore, we conclude that the transmission overhead for this per-layer quantization is very low.

B. Results of MSQE Quantizer

We first test the performance of MSQE quantizer, uniform quantizer and APoT quantizer for a single-node scenario. In order to better compare the performance of these quantizers, we test two cases where the number of quantization bits are $b = 3$ and $b = 5$, respectively.

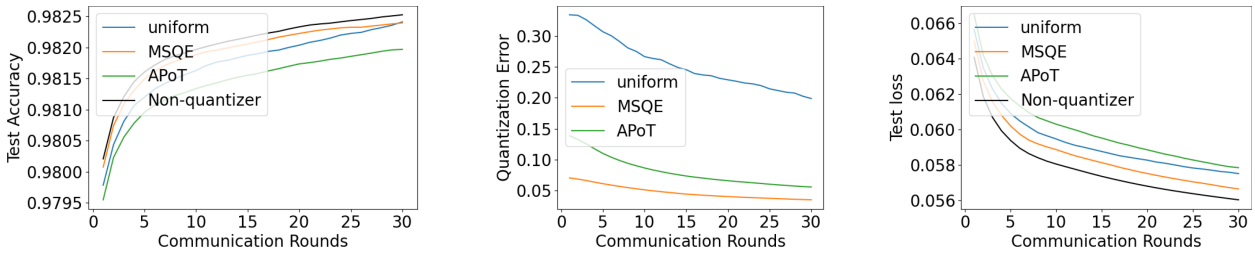
Figs. 4, 5, 7, and 8 show the single-node results under 3-bit and 5-bit quantization for both MNIST and CIFAR10 datasets, where in each figure, the subplots (a) (b) and (c) demonstrate the accuracy, quantization error, and test loss, respectively. In the Fig. 4(a), the accuracy of the MSQE quantizer is about 0.45% lower than the non-quantization case, 1.00% higher than that of the uniform quantizer, and 0.73% higher than that of APoT quantizer. Meanwhile, we can see from Fig. 4(b), the quantization error of MSQE is much lower than that of both the uniform quantizer and APoT quantizer. Finally, we can observe from Fig. 4(c) that the MSQE quantizer is the best among the three quantizers, which is also close to the non-quantization case. Similarly, it can be seen from Fig. 5(a), (b) and (c), among the three quantizers, the MSQE quantizer achieves the lowest quantization error, and its accuracy and test loss are the closest to the non-quantization case. In particular, the accuracy of the MSQE quantizer is about 0.35% lower than the non-quantization case, 0.39% higher than the uniform quantizer, and 0.41% higher than the APoT quantizer. Meanwhile, the quantization error of MSQE is about 81% lower than that of the uniform quantizer and 41% lower than that of the APoT quantizer. We can notice that the performance for 5-bit quantization is much greater than 3-bit quantization due to the finer quantization level. Another observation is that the performance gap among these quantizers is greater for 3-bit quantization case than 5-bit quantization case. The reason is that more quantization boundaries are used, which is closer to the non-quantization case. This shows that our MSQE quantizer has more potential in the coarse-quantized scenarios, such as IoMT area.

Fig. 7 illustrates the performance of 3-bit quantization for single-node case on CIFAR10. We can see that both APoT and uniform quantizer cannot converge, and their quantization errors keep increasing. The reason is that the model for CIFAR10 is much more complex than the model for MNIST. On the contrary, our MSQE quantizer can still achieve reasonable accuracy



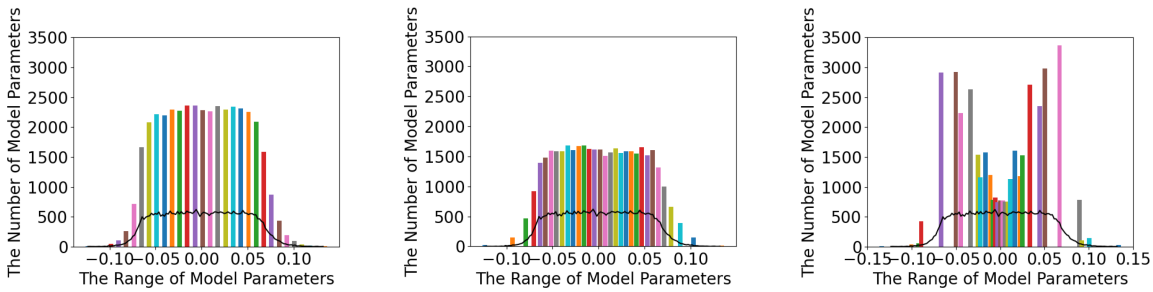
(a) Test accuracy of 3-bit MNIST (b) Quantization error of 3-bit MNIST (c) Test loss of 3-bit MNIST

Fig. 4. 3-bit quantization under single-node on MNIST dataset.



(a) Test accuracy of 5-bit MNIST (b) Quantization error of 5-bit MNIST (c) Test loss of 5-bit MNIST

Fig. 5. 5-bit quantization under single-node on MNIST dataset.



(a) Uniform of 5-bit on MNIST (b) MSQE of 5-bit on MNIST (c) APoT of 5-bit on MNIST

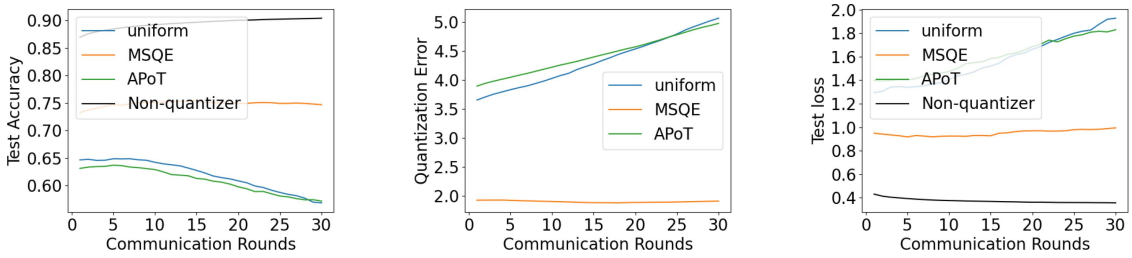
Fig. 6. Parameter distribution map under 5-bit quantization on MNIST dataset.

performance due to its excellent quantization error. Similarly, it can be seen from Fig. 8(a), (b) and (c) that in the case of 5-bit quantization on CIFAR10, the MSQE quantizer has the best performance in terms of test accuracy, quantization error and test loss. Therefore, the above experiment results validate that the MSQE quantizer has excellent performance, especially in the coarse-quantized scenarios.

To further analyze the quantization difference, Fig. 6 depicts the distribution map of model parameters and the quantization boundaries for these three quantizers on MNIST dataset, where the y-axis is the number of model parameters, and the x-axis is the parameter range. The solid black curve represents the distribution of all the model parameters. Each colored bar denotes one quantization boundary, and its height represents the number of parameters that are quantized to it. It is worth pointing out that the denser the intervals, the shorter the bars. In the figure, there are a total of 32 boundaries which divide the whole parameter range into 31 intervals. For the uniform quantizer, the interval range is the same no matter how the parameter distribution looks like. For the APoT quantizer, we can clearly see that the interval density is much higher near parameter 0, and much looser in the other areas. Since the interval range is so small near 0, the number of parameters quantized to these boundaries is very limited. The MSQE quantizer achieves a good tradeoff between these previous two. In particular, it smartly adjusts the interval ranges so that it guarantees the lowest quantization error.

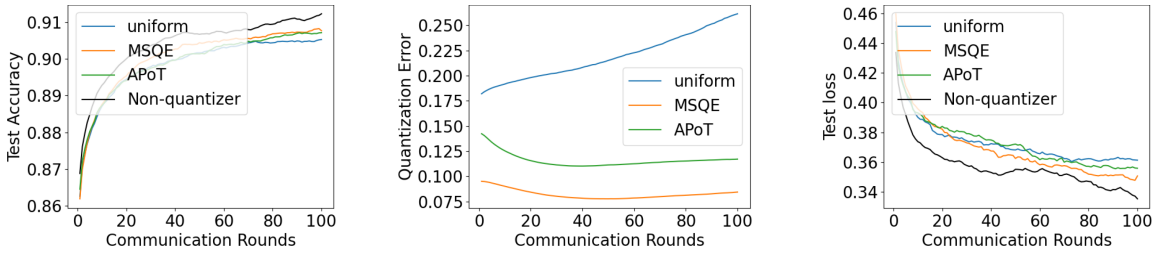
C. Results of FEDMSQE

We further test the performance of the FEDMSQE algorithm for FL scenario. We also implement two other schemes for comparison. The first one is FEDAVG with uniform quantizer, which we call FEDUNIFORM. And the second one is FEDAVG with APoT quantizer, which we call FEDAPOT. In the experiment,



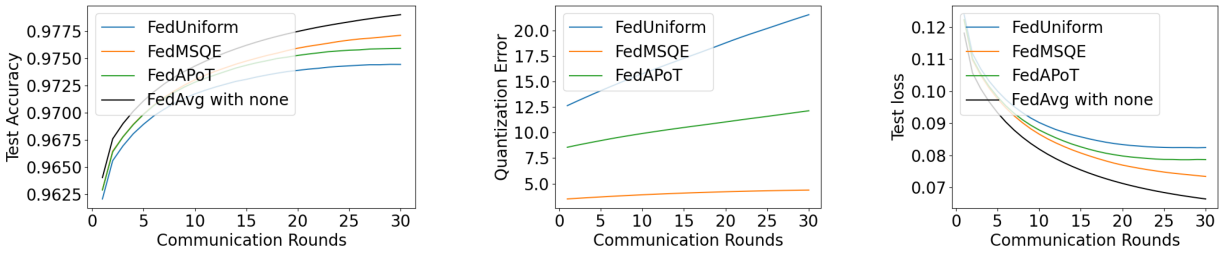
(a) Test accuracy of 3-bit CIFAR10 (b) Quantization error of 3-bit CIFAR10 (c) Test loss of 3-bit CIFAR10

Fig. 7. 3-bit quantization under single-node on CIFAR10 dataset.



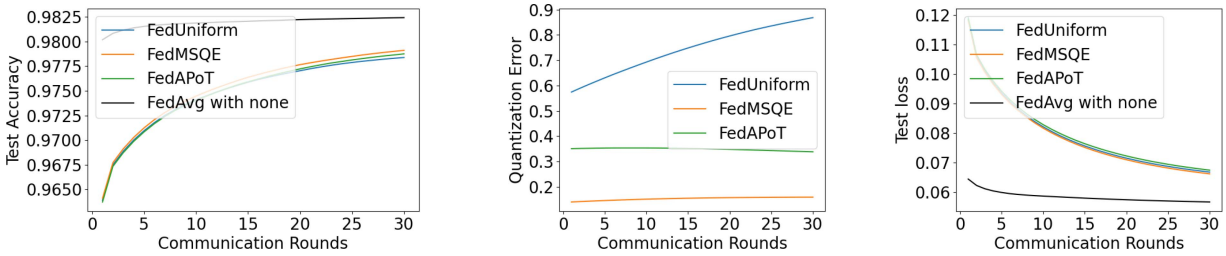
(a) Test accuracy of 5-bit CIFAR10 (b) Quantization error of 5-bit CIFAR10 (c) Test loss of 5-bit CIFAR10

Fig. 8. 5-bit quantization under single-note on CIFAR10 dataset.



(a) Test accuracy of 3-bit MNIST (b) Quantization error of 3-bit MNIST (c) Test loss of 3-bit MNIST

Fig. 9. 3-bit quantization under FL on MNIST dataset.

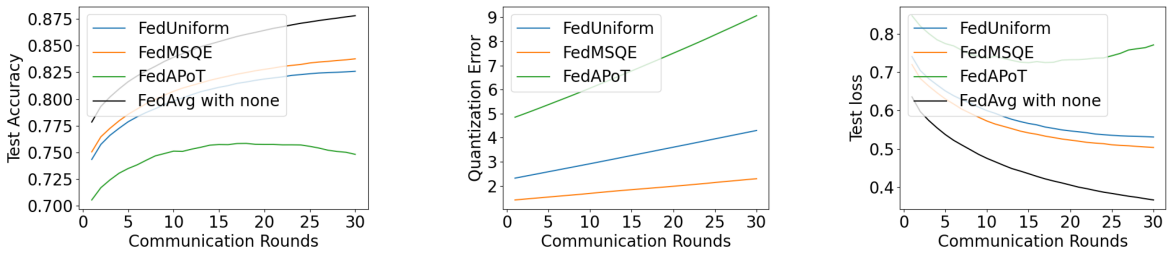


(a) Test accuracy of 5-bit MNIST (b) Quantization error of 5-bit MNIST (c) Test loss of 5-bit MNIST

Fig. 10. 5-bit quantization under FL on MNIST dataset.

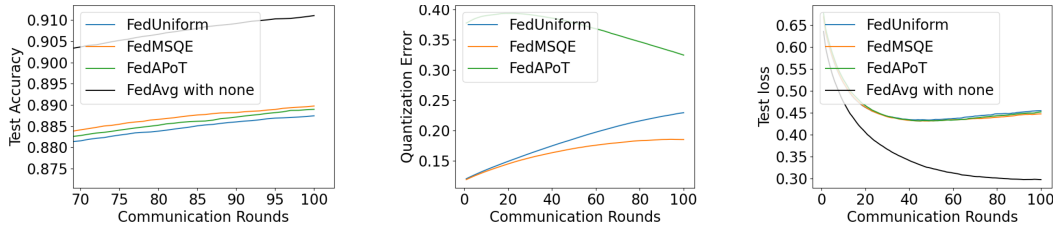
we use the IID data distribution, i.e., each client randomly and uniformly selects data samples from the dataset. We obtain the test accuracy of the global model and the average quantization error of the selected clients. In each round, we randomly select $z = 10$ clients to participate in the MNIST dataset training, $z = 5$ to participate in the CIFAR10 dataset training, and the quantization bits are $b = 3$ and $b = 5$, respectively.

Similar to the single-node, Figs. 9, 10, 11, and 12 show the results for the three schemes under 3-bit and 5-bit quantization on both MNIST and CIFAR10 datasets, where in each figure, the



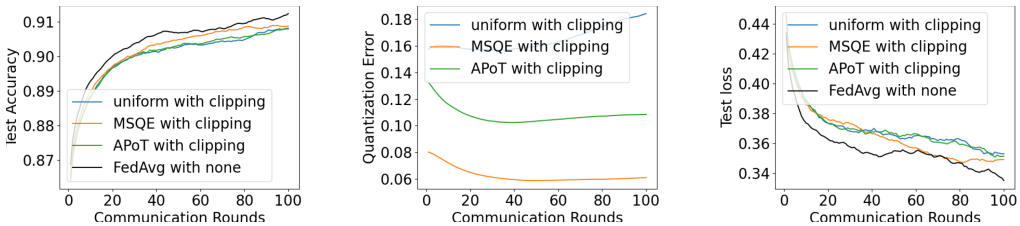
(a) Test accuracy of 3-bit CIFAR10 (b) Quantization error of 3-bit CIFAR10 (c) Test loss of 3-bit CIFAR10

Fig. 11. 3-bit quantization under FL on CIFAR10 dataset.



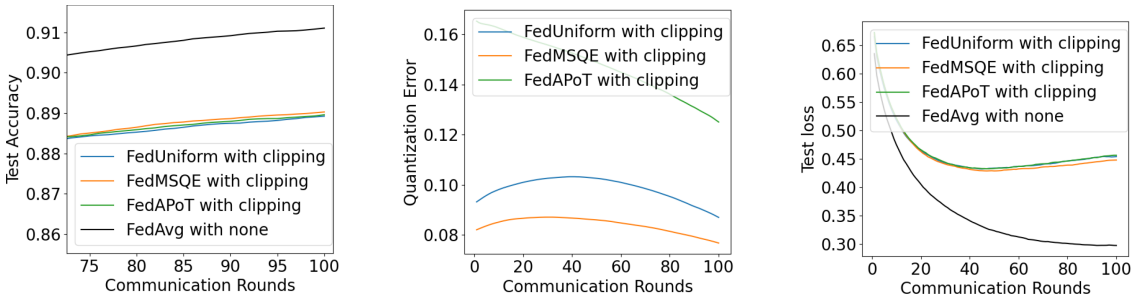
(a) Test accuracy of 5-bit CIFAR10 (b) Quantization error of 5-bit CIFAR10 (c) Test loss of 5-bit CIFAR10

Fig. 12. 5-bit quantization under FL on CIFAR10 dataset.



(a) Test accuracy of 5-bit CIFAR10 (b) Quantization error of 5-bit CIFAR10 (c) Test loss of 5-bit CIFAR10

Fig. 13. Clipping: 5-bit quantization under single-node on CIFAR10 dataset.



(a) Test accuracy of 5-bit CIFAR10 (b) Quantization error of 5-bit CIFAR10 (c) Test loss of 5-bit CIFAR10

Fig. 14. Clipping: 5-bit quantization under FL on CIFAR10 dataset.

subplots (a), (b), and (c) demonstrate the accuracy, quantization error, and test loss, respectively.

In the Fig. 9(a), the accuracy of the non-quantization case is about 2.11% higher than that of the FEDMSQE, 3.12% higher than that of the FEDUNIFORM, and 2.42% higher than the FEDAPOT for the 3-bit quantization FL case. For Fig. 9(b), it can be seen that the quantization error of the FEDMSQE is much lower than that of the other schemes. Finally, Fig. 9(c) shows that among the three schemes, the test loss of the FEDMSQE is the best, which is close to the non-quantization case. Next we

can see from Fig. 10(a), (b) and (c), the accuracy and test loss of the FEDMSQE achieve the best performance among the three schemes, which is due to the minimum quantization error of the FEDMSQE. In particular, the quantization error of FEDMSQE is about 78% lower than that of the FEDUNIFORM and 57% lower than that of the FEDAPOT. We can conclude that the performance of the proposed FEDMSQE outperforms the other two schemes.

As can be seen from Fig. 11(a), (b) and (c), the FEDMSQE has the best performance in terms of accuracy and test loss, since the quantization error is steadily lower than that of the other two schemes. In particular, the accuracy of FEDMSQE is about 8.4% lower than the non-quantization case, 0.7% higher than the FEDUNIFORM, and 4.5% higher than the FEDAPOT. And the quantization error of FEDMSQE is about 42% lower than that of the FEDUNIFORM and 72% lower than that of the FEDAPOT. Regarding the 5-bit quantization scenario, we can see from Fig. 12(a), (b) and (c), that the performance of FEDMSQE is still the best among the three schemes, yet the performance gap becomes diminishing compared to the 3-bit quantization case. Therefore, it confirms the superiority of our proposed FEDMSQE algorithm, especially for the coarse-quantization scenarios, e.g., IoMT field.

D. Results of Clipping

We adopt the clipping method in Section IV, and test the effect of clipping parameters via experiments. Figs. 13 and 14 show the results for the 5-bit quantization on CIFAR10 under single-node and FL scenarios, respectively.

Fig. 13(a), (b) and (c) show the accuracy, quantization error and test loss of all three quantizers with clipping for the single-node case. We can see from the figures that among the three quantizers, the performance of the MSQE quantizer with clipping is still the best. In addition, by comparing Fig. 13 and Fig. 8, we can see that the performance can be significantly enhanced by adopting clipping, in terms of accuracy and quantization error. Similarly, from Fig. 14, we can see that our proposed FEDMSQE algorithm with clipping can also achieve better performance than the other two schemes, as well as those without clipping. In summary, we can conclude that the performance of the MSQE quantizer is the best with/without clipping, which validates the advantage of the MSQE quantizer and FEDMSQE algorithm.

VII. CONCLUSION

This paper designs an MSQE quantizer that can achieve the smallest quantization error, which shows great performance advantage for devices using coarse-quantization, such as IoMT devices. In addition, we propose FEDMSQE algorithm, which applies this quantizer in federated learning scenario. It can improve the security level while guaranteeing a reasonable performance. In our experiments, we have validated that the performance of our scheme is better than uniform and APoT quantizers on single-node and federated learning scenarios.

It is worth pointing out that the MSQE quantizer also has some limitations. Since the quantization boundaries are flexible rather than fixed, it brings about some additional overheads. In general, the overhead is very limited. However, if the quantization is applied to a small amount of parameters, the overhead will become relatively large. Therefore, in the future work, we will study the impact of how to select the number of parameters for quantization, which is a tradeoff between performance gain and overhead. Furthermore, our proposed MSQE quantizer can be applied to many fields, including IoMT area. Our next step is to deploy the MSQE quantizer on real IoMT devices to facilitate further research.

REFERENCES

[1] W. Wang et al., "Blockchain and PUF-based lightweight authentication protocol for wireless medical sensor networks," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8883–8891, Jun. 2022.
[2] K. Yu et al., "Securing critical infrastructures: Deep-learning-based threat detection in IIoT," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 76–82, Oct. 2021.
[3] L. Tan, K. Yu, F. Ming, X. Chen, and G Srivastava, "Secure and resilient artificial intelligence of things: A HoneyNet approach for threat detection and situational awareness," *IEEE Consum. Electron. Mag.*, vol. 11, no. 3, pp. 69–78, May 2022.
[4] C. Feng, B. Liu, K. Yu, S. K. Goudos, and S. Wan, "Blockchain-empowered decentralized horizontal federated learning for 5G-enabled UAVs," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3582–3592, May 2022.
[5] L. Yang, K. Yu, S. X. Yang, C. Chakraborty, Y. Lu, and T. Guo, "An intelligent trust cloud management method for secure clustering in 5G enabled internet of medical things," *IEEE Trans. Ind. Inform.*, p. 1, 2021.
[6] W. Wei et al., "A framework for evaluating gradient leakage attacks in federated learning," Apr. 2020, *arXiv:2004.10397*.
[7] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 16 937–16 947, 2020.
[8] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 2512–2520.
[9] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Adv. Neural Inf. Process. Syst.*, vol. 32, pp. 14 774–14 784, 2019.
[10] Y. Sun, J. Liu, K. Yu, M. Alazab, and K. Lin, "PMRSS: Privacy-preserving medical record searching scheme for intelligent diagnosis in IoT healthcare," *IEEE Trans. Ind. Inform.*, vol. 18, no. 3, pp. 1981–1990, Mar. 2022.
[11] M. Hopkins, M. Mikaitis, D. R. Lester, and S. Furber, "Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations," *Philos. Trans. Roy. Soc. A*, vol. 378, no. 2166, 2020, Art. no. 20190052.
[12] A. Goncharenko, A. Denisov, S. Alyamkin, and E. Terentev, "Fast adjustable threshold for uniform neural network quantization," *Int. J. Comput. Inf. Eng.*, vol. 13, no. 9, pp. 495–499, 2019.
[13] S. Liu, G. Wei, Y. Song, and D. Ding, "Set-membership state estimation subject to uniform quantization effects and communication constraints," *J. Franklin Inst.*, vol. 354, no. 15, pp. 7012–7027, 2017.
[14] B. Li, Z. Wang, Q.-L. Han, and H. Liu, "Distributed quasiconsensus control for stochastic multiagent systems under Round-Robin protocol and uniform quantization," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 6721–6732, 2022.
[15] L. Sheng, Z. Wang, W. Wang, and F. E. Alsaadi, "Output-feedback control for nonlinear stochastic systems with successive packet dropouts and uniform quantization effects," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 47, no. 7, pp. 1181–1191, Jul. 2017.
[16] M. Bashar et al., "Uplink spectral and energy efficiency of cell-free massive MIMO with optimal uniform quantization," *IEEE Trans. Commun.*, vol. 69, no. 1, pp. 223–245, Jun. 2021.
[17] Y. Long, J. H. Park, and D. Ye, "Frequency-dependent fault detection for networked systems under uniform quantization and try-once-discard protocol," *Int. J. Robust Nonlinear Control*, vol. 30, no. 2, pp. 787–803, 2020.
[18] J. L. McKinstry et al., "Discovering low-precision networks close to full-precision networks for efficient inference," in *Proc. IEEE 5th Workshop Energy Efficient Mach. Learn. Cogn. Comput.-NeurIPS Ed.*, 2019, pp. 6–9.
[19] S. Migacz, "8-bit inference with tensorrt," in *Proc. GPU Technol. Conf.*, 2017, p. 5.

[20] A. Malinin and M. Gales, "Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[21] C. Baskin et al., "UNIQ: Uniform noise injection for non-uniform quantization of neural networks," *ACM Trans. Comput. Syst.*, vol. 37, no. 1–4, pp. 1–15, 2021.
[22] Y. Yuan, C. Chen, X. Hu, and S. Peng, "CNQ: Compressor-based non-uniform quantization of deep neural networks," *Chin. J. Electron.*, vol. 29, no. 6, pp. 1126–1133, 2020.
[23] M. Gennari do Nascimento, T. W. Costain, and V. A. Prisacariu, "Finding non-uniform quantization schemes using multi-task gaussian processes," in *Proc. 16th Eur. Conf. Comput. Vis.*, 2020, pp. 383–398.
[24] S. Ouyang, G. Han, Y. Fang, and W. Liu, "LLR-distribution-based non-uniform quantization for RBI-MSD algorithm in MLC flash memory," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 45–48, Jan. 2018.
[25] S. Seo and J. Kim, "Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer," *Appl. Sci.*, vol. 9, no. 12, 2019, Art. no. 2559.
[26] Y. Zhang et al., "A deep learning inference scheme based on pipelined matrix multiplication acceleration design and non-uniform quantization," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Intell. Syst.*, 2021, pp. 281–285.
[27] Y. Sun, L. Li, and D. W. Ho, "Quantized synchronization control of networked nonlinear systems: Dynamic quantizer design with event-triggered mechanism," *IEEE Trans. Cybern.*, pp. 1–13, 2021, doi: 10.1109/TCYB.2021.3090999.
[28] S. Karthick, R. Sakthivel, Y.-K. Ma, S. Mohanapriya, and A. Leelamani, "Disturbance rejection of fractional-order TS fuzzy neural networks based on quantized dynamic output feedback controller," *Appl. Math. Comput.*, vol. 361, pp. 846–857, 2019.
[29] F. Liu et al., "Improving neural network efficiency via post-training quantization with adaptive floating-point," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 5281–5290.
[30] M. Fournarakis and M. Nagel, "In-hindsight quantization range estimation for quantized training," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 3063–3070.
[31] S. Garg, A. Jain, J. Lou, and M. Nahmias, "Confounding tradeoffs for neural network quantization," Feb. 2021, *arXiv:2102.06366*.
[32] H. Shi, M. Hou, Y. Wu, J. Guo, and D. Feng, "Leader-following consensus of first-order multi-agent systems with dynamic hybrid quantizer," *Int. J. Control, Automat. Syst.*, vol. 18, pp. 2765–2773, 2020.
[33] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or –1," Feb. 2016, *arXiv:1602.02830*.
[34] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
[35] H. Pouransari, Z. Tu, and O. Tuzel, "Least squares binary quantization of neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 698–699.
[36] Q. Hu, P. Wang, and J. Cheng, "From hashing to CNNs: Training binary weight networks via hashing," in *Proc. 32nd AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018.
[37] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," May 2016, *arXiv:1605.04711*.
[38] H. Dbouk, H. Sanghvi, M. Mehendale, and N. Shanbhag, "DBQ: A differentiable branch quantizer for lightweight deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 90–106.
[39] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.
[40] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," Feb. 2019, *arXiv:1902.08153*.
[41] N. Singh, D. Data, J. George, and S. Diggavi, "SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization," *IEEE Trans. Autom. Cont.*, to be published, doi: 10.1109/TAC.2022.3145576.
[42] J. Guo, J. Wang, C.-K. Wen, S. Jin, and G. Y. Li, "Compression and acceleration of neural networks for communications," *IEEE Wireless Commun.*, vol. 27, no. 4, pp. 110–117, Aug. 2020.
[43] B. Liu and Z. Ding, "A consensus-based decentralized training algorithm for deep neural networks with communication compression," *Neurocomputing*, vol. 440, pp. 287–296, 2021.
[44] M. Leinonen and M. Codreanu, "Low-complexity vector quantized compressed sensing via deep neural networks," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 1278–1294, 2020.
[45] M. Alazab, S. P. R. M., P. M., P. Reddy, T. R. Gadekallu, and Q.-V. Pham, "Federated learning for cybersecurity: Concepts, challenges and future directions," *IEEE Trans. Ind. Inform.*, vol. 18, no. 5, pp. 3501–3509, May 2022.
[46] X. Dai et al., "Hyper-sphere quantization: Communication-efficient SGD for federated learning," Nov. 2019, *arXiv:1911.04655*.
[47] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "An exact quantized decentralized gradient descent algorithm," *IEEE Trans. Signal Process.*, vol. 67, no. 19, pp. 4934–4947, Oct. 2019.
[48] S. P. Ramu et al., "Federated learning enabled digital twins for smart cities: Concepts, recent advances, and future directions," *Sustain. Cities Soc.*, vol. 79, 2022, Art. no. 10366.
[49] S. Chen, C. Shen, L. Zhang, and Y. Tang, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6804–6819, Oct. 2021.
[50] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
[51] Y. Yu, J. Wu, and L. Huang, "Double quantization for communication-efficient distributed optimization," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.
[52] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Federated learning with quantization constraints," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 8851–8855.
[53] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2021–2031.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[55] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," 2019, *arXiv:1909.13144*.