Defending Against Byzantine Attacks in Quantum Federated Learning

Qi Xia
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
qxia@cs.wm.edu

Zeyi Tao
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
ztao@cs.wm.edu

Qun Li
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
liqun@cs.wm.edu

***Abstract*—By combining the advantages of both quantum computing and deep learning, quantum neural networks have become popular in recent research. In order to collaborate multiple quantum machines with local training data to train a global model, quantum federated learning is proposed. However, similar to classic federated learning, when communicating with multiple machines, quantum federated learning also faces the threats of Byzantine attacks. The byzantine attack is a kind of attack in a distributed system when some machines upload malicious information instead of the honest computational results to the server. In this article, we compare the differences of Byzantine problems between classic distributed learning and quantum federated learning, and modify the previously proposed four kinds of Byzantine tolerant algorithms to the quantum version. We conduct simulated experiments to show a similar performance of the quantum version with the classic version.**

***Index Terms*—quantum neural networks, federated learning, byzantine problems**

I. Introduction

Both quantum computing [25] and deep neural network [20] have been greatly developed since the last several decades. In the early 1980s, Paul Benioff first proposed the novel idea of building a Turing machine using quantum computing [4]. The explorations on quantum computing based Turing machine then were conducted by Richard Feynman [14], David Deutsch [11], etc. In 1994, Shor's algorithm was proposed to use a quantum machine to factor a large integer in polynomial time and its time complexity is exponentially faster than the fastest algorithm on classic computer [24]. This makes people start to believe in the capability of better performance in the quantum machine than the classic machine. More recently, research teams from Google AI [2] and USTC [36] respectively claimed quantum supremacy for tasks that are infeasible on any classic computer. In the meantime, deep neural network [20] has been found efficient in many practical tasks such as computer vision [16], [19], [21], [27], natural language processing [7], [9], [12], [26], etc. It uses a hierarchical neural architecture to learn from the training data and nowadays is used everywhere in our daily life from online shopping to work and entertainment.

In order to reduce the huge computational cost in larger and deeper classic neural networks, in recent years, scientists start to explore the combination of quantum computing and deep learning, and thus quantum neural network was proposed [17]. A quantum neural network uses the idea of a classical neural network in a quantum way to learn from the training data. By utilizing the main property of qubit superposition and entanglement in quantum mechanics, it tries to improve the computational efficiency and reduce the long training time and heavy computational resources in deep learning [3], [15], [22], [23], [28]. However, in the training process of quantum deep learning, sometimes it is necessary to train a model through multiple quantum machines in a distributed manner. When there are multiple quantum machines with local quantum data, in order to collaboratively train the global model, Xia et al. proposed the quantum federated learning framework [30].

Because quantum federated learning is also a special kind of distributed learning, it also suffers the threat of Byzantine attacks. When working with distributed learning, the Byzantine problem happens naturally in an environment of multiple nodes. Byzantine faults were first investigated by Lamport et al. in 1982 [18]. Byzantine faults [6], [13] describe a general problem in distributed computing systems that one or more computing nodes may fail and provide adversarial or empty computational results where there is imperfect information from the server side on the failure information. There are several existing Byzantine-resilient algorithms in the classic distributed machine learning area. Basically, there are four directions to defend against Byzantine attacks: score-based, median-based, and distance-based, reference dataset-based algorithms. Krum is the first Byzantine-resilient algorithm in distributed deep learning area [5]. It is proposed by Blanchard et al. to measure the scores for each uploaded gradient in the PS (Parameter Server) and chose the gradient with the highest score as the aggregated gradient. Later they also extend their work to asynchronous distributed learning [10]. There is much following work focus on using geometric median or its variant to defend against Byzantine attacks. For example, Xie et al. proposed geometric median, marginal median, and median-around-median [32], Yin et al. proposed coordinate-wised median [35], Lili et al. proposed a batch normalized median [8], Alistarh et al. proposed a more complicated modification of median-based methods called ByzantineSGD [1]. For distance-based direction, Xia et al. proposed an alternative method called FABA [31]. Instead of using median-based methods, they used Euclidean distance to remove outlier gradients. They adaptively remove outliers based on the center current remaining gradients. They later provided another Byzantine-resilient algorithm for large-scale distributed machine learning [29]. As for the reference dataset-based method, Xie et al. proposed methods based on a reference dataset to give scores for each node to solve fault-tolerance problems in distributed machine learning [33], [34]. Xia et al. proposed a self-adaptive reference dataset-based two-filter algorithm ToFi to defend against Byzantine problems in federated learning.

However, all the current proposed algorithm is designed for classic distributed learning or federated learning framework. In quantum federated learning framework such as QuantumFed [30], the update unitaries instead of computed gradients or weights in classic distributed learning are transmitted. Because they are in different spaces, the attack and defense algorithms need to be modified in a quantum environment. In this paper, we focus on the Byzantine problems in the quantum federated learning scenario. In summary, our contributions are:

- We compare the difference of Byzantine problems between classic distributed learning and quantum federated learning, extend the Byzantine problems in the QuantumFed framework and theoretically define this problem.
- We modify the previous proposed Byzantine tolerant algorithms Krum, FABA, and ToFi to the quantum environment and discuss the reasons that median-based algorithm is not able to use in the quantum scenario.
- We conduct several simulation experiments to compare the modified Byzantine-resilient algorithms in the QuantumFed framework with Byzantine attacks.

II. Preliminaries

A. Quantum Computing Basis

In quantum computing, the qubit is the basic unit to represent the information. A qubit has two basis states $|0\rangle$ and $|1\rangle$ like the classic bit in a traditional computer, but it can also be in a superposition, which is a combination of the two basis states $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha^2 + \beta^2 = 1$. Therefore, a qubit is capable to express more information than a classic bit. When observing the qubit, it will collapse to one of the basis states with corresponding probability, and thus we can get a statistically accurate estimation after sufficient times of observations. Besides, the entanglement of qubits allows more qubits to have correlations with each other and n qubits, in this scenario, will have 2^n basis states and can be in a superposition among them, which carries an exponentially increasing amount of information.

In order to perform computations on the qubits, there are several common quantum logic gates: Pauli-X, Pauli-Y, Pauli-Z, Hadamard, Controlled Not. Unlike the AND and OR gate from classic computer, quantum operators are always reversible and will obtain an output with the same dimension, and thus can be represented as a unitary. If we represent the input qubits state as a column vector, for example, $|\psi\rangle = \frac{1}{\sqrt{6}}|00\rangle + \frac{1}{\sqrt{6}}|01\rangle + \frac{1}{\sqrt{3}}|10\rangle + \frac{1}{\sqrt{3}}|11\rangle \rightarrow [\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}]^T$, the output of the quantum operators are unitaries left multiplying states. We list the unitary representation of some common quantum logic gates in Table I.

Gate	Unitary
Pauli-X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Controlled Not	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

TABLE I
UNITARY REPRESENTATION OF COMMON QUANTUM LOGIC GATES.

B. Quantum Neural Network

There are lots of explorations of implementing deep neural networks in a quantum way, that is, using a quantum perceptron in order to get similar generality as classic neural networks. In this article, we adopt a widely used quantum deep neural network architecture as Figure 1. Assume in layer

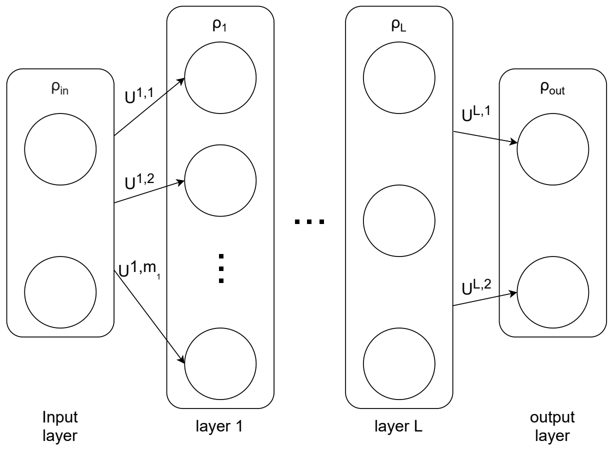

Fig. 1. An architecture example of the quantum neural network.

l, the input is a state ρ^{l-1} of m_{l-1} qubits and this layer will give an output of m_l qubits, then the l-th layer transition map $\mathcal{E}^l$ is given by:

$$\mathcal{E}^l(\rho^{l-1}) = \mathrm{tr}_{l-1}(U^l(\rho^{l-1} \otimes |0\cdots0\rangle_l\langle 0\cdots 0|)U^{l^\dagger}) \quad (1)$$

U^l is the $2^{m_{l-1}+m_l} \times 2^{m_{l-1}+m_l}$ dimensional perceptron unitary of layer l. A partial trace operation is performed to get the output state of layer l. For simplicity, we apply U^l by sequentially applying m_l independent perceptron unitaries $U^{l,j}$ that act on m_{l-1} input qubits and j-th qubit in layer l, that is, $U^l = \prod_{j=m_l}^{1} U^{l,j}$. Note that $U^{l,j}$ here are acting on the current layer, which means $U^{l,j}$ is actually $U^{l,j} \otimes \mathbb{I}^l_{1,\cdots j-1,j+1,\cdots m_l}$. In this way, we can feedforward the input state layer by layer to get an output state:

$$\rho^{out} = \mathcal{E}^{out}(\mathcal{E}^L(\cdots\mathcal{E}^2(\mathcal{E}^1(\rho^{in}))\cdots)) \quad (2)$$

The hyperparameters of the quantum deep neural network are the unitaries, so as long as we have the network structure and unitaries, we can describe a model. The method to derive the unitaries is very similar to backpropagation, which is discussed in [3]. In order to update the model in each iteration, we apply an update unitary to the current unitaries. So, the training process is actually to use the training data to derive the update unitary.

In order to represent the input and output data in a quantum way, for data that is stored by classic bits, we need to first transform the data to qubit representation. One way to do this is that we can use a d-qubits state $|\psi\rangle_d$ to represent a superposition of 2^d basis states in Hilbert space $\mathcal{H}^{2^d}$, that is, $|\psi\rangle_d = \sum_{i=1}^{2^d} \alpha_i |z_i\rangle$ where α_i is the complex amplitude and z_i is a basis state in $\mathcal{H}^{2^d}$. In this way, we can transform the classic data to quantum data, and then we can use the quantum data to train the quantum neural network and perform the inference.

C. *Quantum Federated Learning*

Quantum federated learning is a collaborative way to train a global model using multiple quantum machines where each machine keeps its own quantum data. QuantumFed framework was first proposed by Xia et al. in [30]. Similar to federated learning in classic deep learning, each node keeps their private data and does not share it with each other or the central server. Therefore, the training data in QuantumFed is non-i.i.d.

Basically, the QuantumFed framework includes two major components: QuanFedNode, which is the local update running on the node side; and QuanFedServer, which is the global update running on the server side. QuanFedNode is an analog of local training in classic federated learning. It fetches the current global model from the central server, derives the update unitary using local training data and hyperparameters such as learning rate, local and global data volume. After deriving the update unitary, the node sends it to the central server as the update information. QuanFedServer algorithm takes the update unitaries derived by each node as the input and using them to update the global model. We will keep updating the global model until converging to a feasible model. In addition, like classic federated learning, in each training round, it is not necessary that all nodes will perform the local update. Assume there are total N nodes in the quantum federate learning system, we randomly select N_p ($N_p \leq N$) nodes to perform the computation.

III. Byzantine Problems in QuantumFed

A. *Problem Definition*

Like classic federated learning, when the central server communicates with multiple quantum devices, the quantum federated learning also faces the threats of Byzantine attacks. In QuantumFed, byzantine problems happen when some of the quantum nodes perform the computation maliciously. We use Figure 2 as an example, when transmitting the update information with the central server, instead of uploading the real update unitaries, the byzantine nodes (Node 4) send a

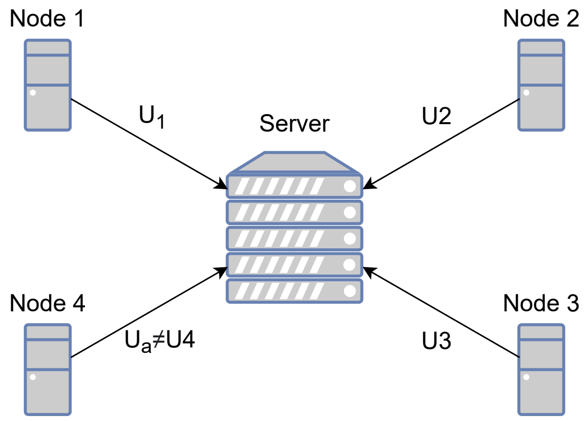

Fig. 2. An example of Byzantine problem in a four-node quantum federated learning environment.

randomly generated or malicious unitary to the server. Theoretically, we define the byzantine model in QuantumFed as follows. Assume that U_i is the correct update unitary computed by node i with local training data, U_i^R is the actual update unitary received by the central server from node i, and U_a is an attack unitary which is the same shape as U_i, we have:

Definition 1 (Byzantine Model in QuantumFed):

$$U_i^R = \begin{cases} U_i & \text{if node } i \text{ is honest} \\ U_a \neq U_i & \text{otherwise} \end{cases} \tag{3}$$

Because the QuanFedServer updates the global model by successively applying the update unitaries received by each participating node, the Byzantine attacks may significantly harm the training process or even make the whole global model converge to a totally wrong model. Therefore, it is necessary to check if the previously proposed algorithms are still capable of defending against Byzantine attacks in quantum federated learning.

In order to transform classic Byzantine-resilient algorithms into the quantum scenario, some modifications are needed because of the gap between classic federated learning and quantum federated learning. The gap comes from the update information between the central server and each node. In classic federated learning, the update information is the gradient or weight computed by each node and it can be represented as a tensor. However, in quantum federated learning, the update information can only be represented as a unitary matrix, which is in a different space and the distance and norm are totally different from the tensor space. Therefore, in the following sections, we discuss the quantum versions of distance-based algorithm FABA, score-based algorithm Krum, and reference dataset-based algorithm ToFi. In the end, we also discuss the reason why meidan-based algorithms are difficult to transform to quantum versions.

B. *Quantum Version of FABA*

FABA is a distance-based algorithm to defend against Byzantine problems in classic distributed deep neural networks [31]. The input of FABA are the gradients computed by each worker, which may include malicious update information uploaded by the Byzantine worker. This algorithm is running on the parameter server and outputs the updated global model after aggregate the update information. The main idea of FABA is based on the observation that in classic distributed deep neural networks, the dataset in each worker is i.i.d., and thus the computed gradients for the honest workers are close to each other. Therefore, when the server receives all the update information from each worker, the server can filter out those abnormal gradients by conducting the distance information. The main steps of the FABA algorithm are: (1) compute the average gradient of the current gradients; (2) remove the gradient that is farthest from the average gradient; (3) repeat these two steps based on the estimated proportion of Byzantine workers.

The major change of the distance-based algorithm FABA in quantum computing is from the change of norm and distance metric. Because here our operations are based on unitaries, we cannot use the classic Euclidean distance in tensor operation. Here we use an element-wise matrix Euclidean distance instead. Assume U_1, U_2 are unitaries of size $n \times n$ and $u_1^{i,j}, u_2^{i,j}$ are corresponding entries in U_1 and U_2 where $i, j = 1, 2, \cdots, n$. The distance between U_1 and U_2 $d(U_1, U_2)$ is defined as:

$$d(U_1, U_2) = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(u_1^{i,j} - u_2^{i,j})^2} \tag{4}$$

Other than the distance metric, we also need to compute the average of some unitaries when implementing FABA. Here we use the similar element-wise average over unitaries. If we assume $U_3, \cdots, U_m$ and $u_3^{i,j}, \cdots, u_m^{i,j}$ similar as before, each element $u_a^{i,j}$ of the average matrix U_a is defined as:

$$u_a^{i,j} = \frac{1}{m}\sum_{k=1}^{m} u_k^{i,j} \tag{5}$$

Note that U_a is not necessarily a unitary.

Based on these changes we can implement FABA in our QuantumFed framework. The algorithm is described in Algorithm 1.

Algorithm 1 Quantum Version of FABA

```
Input:
    The number of selected nodes $N_p$. Without loss of generality, assume they are node 1, node 2, $\cdots$, node $N_p$;
    The received update unitaries computed from selected nodes: $G_U = \{U_1^R, U_2^R, \cdots, U_{N_p}^R\}$;
    The assumed proportion of Byzantine workers: $\alpha$;
    Initialize $k = 1$.
Output:
    The updated global model.
1: If $k < \alpha \cdot N_p$, continue, else go to Step 5;
2: Compute mean of $G_U$ as $U_0$ using (5);
3: For every $U_i^R$ in $G_U$, compute the difference $d(U_0, U_i^R)$ using (4). Delete the one that has the largest difference from $G_U$;
4: $k = k + 1$ and go back to Step 1;
5: Use the rest update unitaries in $G_U$ to continue the QuanFedServer algorithm in QuantumFed framework and derive the updated global model.
```

C. Quantum Version of Krum

Krum is a score-based algorithm that is the first proposed algorithm to defend against Byzantine problems in in classic distributed deep neural networks [5]. Krum is also based on the idea to remove those gradients who have abnormal distance information. However, in the design of Krum, it brings a scoring system to this algorithm. This scoring system is used to give each uploaded gradient a score to measure how close it is to its neighbors. Because this algorithm also has the assumption that dataset in each worker is i.i.d., the scoring metric can use the distance information to decide how possible the update information is uploaded from a Byzantine worker. In the end, the server only chooses the one that is least possible to be malicious as the aggregation result and only use this gradient to update the global model. The main steps of the Krum algorithm are: (1) for each gradient, find its several closest neighbor gradients; (2) compute the score of each gradient by the distance summation to its neighbor gradients; (3) choose the one that has the lowest score for the model update.

For the score-based algorithm Krum, the gap between the classic version and quantum version is roughly similar to the gap in FABA. In the Krum algorithm, the core step is to get the k-nearest neighbors of each uploaded gradients. In the quantum scenario, the analog is to find the k-nearest neighbors of each uploaded update unitaries. Therefore, we must have a new distance metric between unitaries to transform Krum into a quantum version. As we discussed in Section III-B, the distance metric between two unitaries is defined as (4). Here we can adopt the same element-wise matrix Euclidean distance to the Krum algorithm. We describe the quantum version of Krum in Algorithm 2.

D. Quantum Version of ToFi

ToFi is a reference dataset-based algorithm that is proposed to defend against Byzantine problems in federated learning. It is the first work to study Byzantine problems in the federated learning area. In federated learning, the dataset of each worker is stored locally and therefore, the distribution may be non-i.i.d. As a result of that, it is very hard to distinguish the Byzantine update information and honest update information just based on the distance information. Here ToFi uses a small-sized reference dataset to help distinguish the malicious information. This reference dataset has the same distribution with the whole dataset and can be used to examine the performance of the update information uploaded by each worker. The main steps of the ToFi algorithm are: (1) examine the loss based on the reference dataset for each uploaded update weights; (2)

Algorithm 2 Quantum Version of Krum

```
Input:
    The number of selected nodes $N_p$. Without loss of generality, assume they are node 1, node 2, $\cdots$, node $N_p$;
    The received update unitaries computed from selected nodes: $G_U = \{U_1^R, U_2^R, \cdots, U_{N_p}^R\}$;
    The assumed proportion of Byzantine workers: $\alpha$;
Output:
    The updated global model.
1: Compute the estimated Byzantine tolerant parameter $f = ceiling(N_p \cdot \alpha)$;
2: For each $U_i^R$ in $G_U$, find the $N_p - f - 2$ closest update unitaries. Denote the set of those $N_p - f - 2$ update unitaries of $U_i^R$ as $C_i$. Here we use the distance metric in (4) to define the distance;
3: For each $U_i^R$, compute the corresponding score $s_i$ using:
       $s_i = \sum_{j \in C_i} d(U_i^R, U_j^R)$    (6)
4: Derive the index $i_*$ with the smallest score:
       $i_* = \arg_i \min s_i$    (7)
5: Use the rest update unitary $U_{i_*}^R$ to continue the QuanFedServer algorithm in QuantumFed framework and derive the updated global model.
```

Algorithm 3 Quantum Version of ToFi

```
Input:
    The number of selected nodes $N_p$. Without loss of generality, assume they are node 1, node 2, $\cdots$, node $N_p$;
    The received update unitaries computed from selected nodes: $G_U = \{U_1^R, U_2^R, \cdots, U_{N_p}^R\}$;
    Reference dataset $\xi_R$;
    Model parameter unitary $U$;
    Predefined loss filter parameter $\tau$;
Output:
    The updated global model.
1: Here we let $f(U, \xi_R)$ be the loss function on the model parameter unitary $U$ and reference dataset $\xi_R$. Examine the loss for each node with reference dataset $l_i = f(U_i^R U, \xi_R), i = 1, 2, \cdots, N_p$;
2: Compute the mean $\mu = \frac{1}{N_p}\sum_{i=0}^{N_p} l_i$ and standard deviation $\sigma = \sqrt{\frac{\sum_{i=0}^{N_p}(l_i-\mu)^2}{N_p}}$ for $l_i$;
3: Compute the normalized loss $L_i = \frac{l_i-\mu}{\sigma}$;
4: Filter the uploaded update unitaries with the normalized loss $G_f = \{U_i^R | e^{-L_i} < \tau\}$;
5: Use the rest update unitaries in $G_f$ to continue the QuanFedServer algorithm in QuantumFed framework and derive the updated global model.
```

normalize the examined loss; (3) filter out those weights who result in too big loss.

Implementing the reference dataset-based algorithm ToFi in classic federated learning and quantum federated learning is roughly similar. However, there are some small differences listed below.

- The update similarity-based filter is based on the update unitaries rather than computing by the local updated weight. The similarity is based on the element-wise matrix Euclidean distance defined in (4).
- Because the optimal value of our cost function is 1, we update the reference dataset-based loss filter by filter out the update unitaries with small examined loss.
- Because there is no α-weight in our QuantumFed framework, we skip the softmax function that assigns the α-weight.

We also keep a small reference dataset in the central server to examine the loss and filter out abnormal update unitaries. According to the above changes, we can implement our ToFi algorithm in the QuantumFed framework. The quantum version of ToFi is described in Algorithm 3.

E. Discussion about Median-based Algorithms

There are many geometric median-based algorithm to defend against Byzantine attacks in classic distributed deep neural networks such as [1], [8], [32], [35]. In classic distributed machine learning, we usually use simple average to aggregate the uploaded gradients or weights, but instead of taking the average, all these algorithms follow a similar idea of using the geometric median as the aggregation result. Geometric median is an important location estimator in statistics. It is point in Euclidean space that minimizes the sum of distances to the sample points in a discrete set. Theoretically, it is defined as:

Definition 2 (Geometric Median): For a given set of m points $x_1, x_2, \ldots, x_m$, with each $x_i \in \mathbb{R}^n$, the geometric median is defined as

$$\underset{y \in \mathbb{R}^n}{arg\,min} \sum_{i=1}^{m} \|x_i - y\|_2 \tag{8}$$

From the definition, geometric median can express the majority information of the set of received update information, which is with high possibility to be honest, and can also keep the property of aggregating the update information.

However, for the Median-based algorithm, it is very hard to be directly transformed to the quantum version. The reasons are below:

- There is no quantum analog for a geometric median of unitaries. The definition of geometric median in classic Euclidean space is the point minimizing the sum of distances to the sample points. Although we can still define the similar geometric median in unitary space to find the unitary that minimizes the sum of distances to other unitaries and use the element-wise matrix Euclidean distance, it is not feasible here because of two reasons. First, it is very hard to get this unitary by simply modifying the current algorithm to find the geometric median in Euclidean space. A new algorithm needs to be designed. Second, even if we can solve this geometric median, it may not be suitable in a quantum scenario because the geometric median in Euclidean space and unitary space have different properties and meanings.

- In classic distributed learning, the aggregation method is by taking the average of the uploaded gradients. Geometric median also takes similar properties and it can achieve a similar performance without Byzantine attacks. However, in the QuantumFed framework, the global update is performed by successively applying the update unitaries. Simply changing it to apply by a geometric median unitary may cause incorrect training.

Therefore, we choose not to derive a quantum version of median-based algorithms. It needs more explorations for median-based algorithms in the quantum scenario.

IV. Experiment Results

A. Environment Setup

In the experiment, we conduct the same QuantumFed environment as [30] to simulate the synthetic training data, quantum neural network architecture, and heterogeneous federated learning environment. Our experiment is simulated by QuTip library [1] (Quantum Toolbox in Python).

First, in order to get the training data, we randomly generate a global unitary U_g which is the unitary we would like to approximate. Then we randomly generate the training data input and apply the global unitary to the input to get the corresponding output. We use the randomly generated input and output pair as the clean training data. The same method is applied to generate the test data. In this way, we can generate clean training data $(|\phi^{in}_{n,x}\rangle, U_g|\phi^{in}_{n,x}\rangle)$ on the node n side, and test data $(|\phi^{in}_{test,x}\rangle, U_g|\phi^{in}_{test,x}\rangle)$ on the central server side. In order to show the robustness of the training, we also pollute a proportion of training data with randomly generated input and output to get noisy training data.

Second, as for the quantum neural network architecture, because the experiments that we conduct are in a simulated environment using a classic computer and the computational complexity increases exponentially with the width of the network increases, we choose to train small size quantum neural networks with a width that are not greater than 3. In this section, if not specified, we choose a network of size 2-3-2.

Third, in order to simulate the heterogeneous federated learning environment, we put similar training data into the same node. We first gather all the generated training data from all nodes, sort them by their vector representation value, and divide them to each node in order. In this way, we can somehow guarantee that the data on each node is not i.i.d.

Fourth, we measure the experiment results using two metrics. First metric is the fidelity cost function that is defined in [3], to show the probability that the output state will be identified as the output label in a measurement. We also adopt another metric mean square error (MSE) that are widely used in classic machine learning as a comparison. The MSE is defined below:

$$\text{MSE} = \frac{1}{N}\sum_{x=1}^{N} \|\rho_x^{out} - |\phi_x^{out}\rangle\langle\phi_x^{out}|\|^2 \tag{9}$$

We examine our experiments using both metrics on the test data to show the performance.

Fifth, to simulate the Byzantine environment, we use Gaussian distribution to generate a random hermitian matrix and transform it to a random unitary as the attack update unitary. We choose 30% as the Byzantine ratio which means 30% of the total quantum nodes are Byzantine nodes. We also simulate a clean environment without Byzantine attacks as a comparison.

We use no operation as a comparison to see the performance if we take no operation for the Byzantine attack. We use [2, 3, 2] quantum neural network, fidelity and MSE loss metrics, and set $\eta = 1.0, \epsilon = 0.1$.

B. All-Node Participating Quantum Federated Learning

In this experiment, we set $N = 10, N_p = 10$, which means there are a total 10 nodes and in each epoch, and all nodes are selected to perform computations in each synchronization. We compare FABA, ToFi, Krum, no operation, and ideal cases when there are no Byzantine attacks in Byzantine environment and clean environment. The experiment results are presented in Figure 3.

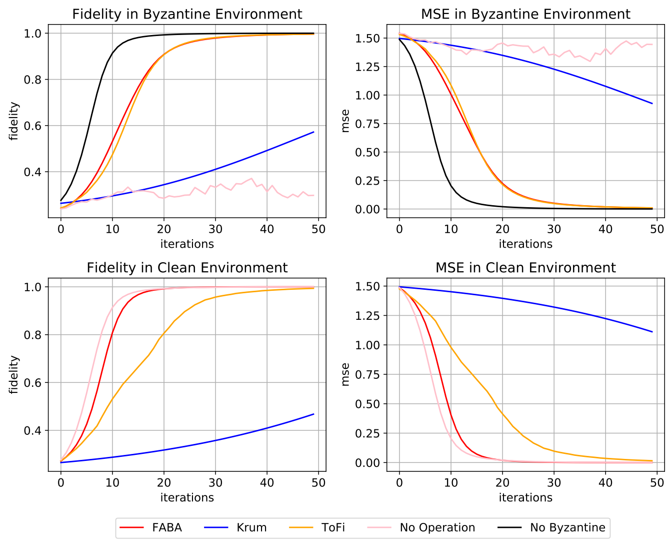

Fig. 3. Experiment results of a [2, 3, 2] quantum network with FABA, Krum, ToFi, No Operation and No Byzantine in all-participating quantum federated learning Byzantine and clean environment.

As we can see in Figure 3, both FABA and ToFi are capable of defending against Byzantine attacks in all participating quantum federated learning. FABA is a little bit faster than ToFi, but the performance is very similar. Krum seems to have the trend to defend against Byzantine attacks, but the speed is very slow and thus is not practical. In a clean environment without Byzantine attack, we can see that FABA

[1]QuTip: https://github.com/qutip/qutip

achieves better convergence speed than ToFi. It is because, in a clean environment, the loss performance of each node is similar. After the normalization, some useful information may be filtered out. Therefore, the training speed is slower than FABA. Krum still has the worst performance.

C. Selected-Node Participating Quantum Federated Learning

We set $N = 100, N_p = 10$ to simulate the selected-node participating quantum federated learning environment. We totally set 100 quantum nodes, among which 30 of them are Byzantine nodes. In each iteration, 10 nodes are randomly selected to perform computations. The experiment results are shown in Figure 4.

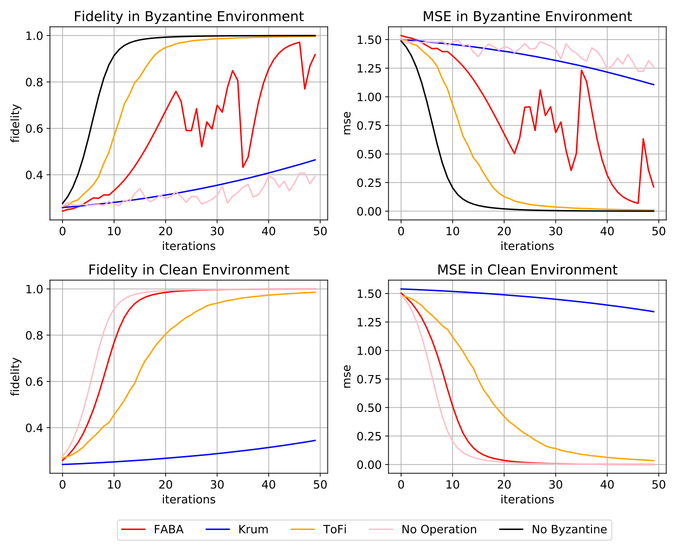

Fig. 4. Experiment results of a [2, 3, 2] quantum network with FABA, Krum, ToFi, No Operation and No Byzantine in selected-participating quantum federated learning Byzantine and clean environment.

From Figure 4, it is obvious that only ToFi can defend against Byzantine attacks in the selected-node participating quantum federated learning environment. FABA can somehow defend this attack, but the performance is worse and very unstable. Krum has the worst performance and the performance is even similar to the no operation case. As for the clean environment, the performance is similar to the all-node participating scenario.

D. Discussions

Our simulated experiments show that the previously proposed algorithm FABA and ToFi are still capable of defending against Byzantine attacks in quantum federate learning. The convergence speed of Krum is very slow and thus is not feasible in practice. Similar to classic federated learning, only ToFi is Byzantine resilient in selected-node participating federated learning. However, the exploration of Byzantine problems in quantum federated learning is still at an early stage. We only use Gaussian distribution to generate Byzantine attacks. It still needs further research about how to defend and attack quantum federate learning systems.

V. Conclusion

In this paper, we investigate the Byzantine problems in quantum federated learning. We compare the update differences between the quantum distributed system and the classic distributed system. Moreover, we modify the previously proposed three kinds of Byzantine-resilient algorithms into quantum version and discuss why median-based algorithm does not work in the quantum scenario. Several simulated experiments are conducted to compare the performance of different quantum version algorithms. Currently, the exploration of Byzantine problems in quantum federated learning is still at an early stage. We need more efforts in this direction.

Acknowledgment

This project was supported in part by US National Science Foundation grant CNS-1816399. This work was also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

References

[1] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, S. Bengio *et al.*, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 4613–4623. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/a07c2f3b3b907aaf8436a26c6d77f0a2-Paper.pdf

[2] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, Oct 2019. [Online]. Available: https://doi.org/10.1038/s41586-019-1666-5

[3] K. Beer *et al.*, "Training deep quantum neural networks," *Nature Communications*, vol. 11, p. 808, Feb 2020. [Online]. Available: https://doi.org/10.1038/s41467-020-14454-2

[4] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *Journal of Statistical Physics*, vol. 22, pp. 563–591, May 1980. [Online]. Available: https://doi.org/10.1007/BF01011339

[5] P. Blanchard *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems 30*, I. Guyon *et al.*, Eds. Curran Associates, Inc., 2017, pp. 119–129. [Online]. Available: http://papers.nips.cc/paper/6617-machine-learning-with-adversaries-byzantine-tolerant-gradient-descent.pdf

[6] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. USA: USENIX Association, 1999, p. 173–186.

[7] Q. Chen *et al.*, "Enhanced LSTM for natural language inference," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1657–1668. [Online]. Available: https://www.aclweb.org/anthology/P17-1152

[8] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, pp. 44:1–44:25, Dec. 2017. [Online]. Available: http://doi.acm.org/10.1145/3154503

[9] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 160–167. [Online]. Available: https://doi.org/10.1145/1390156.1390177

[10] G. Damaskinos *et al.*, "Asynchronous Byzantine machine learning (the case of SGD)," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 1145–1154. [Online]. Available: http://proceedings.mlr.press/v80/damaskinos18a.html

[11] D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer,” *Proceedings of the Royal Society of London Series A*, vol. 400, pp. 97–117, Jul. 1985.

[12] J. Devlin *et al.*, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: http://aclweb.org/anthology/N19-1423

[13] K. Driscoll *et al.*, “The real byzantine generals,” in *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, vol. 2, 2004, pp. 6.D.4–61.

[14] R. P. Feynman, “Simulating physics with computers,” *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, Jun 1982. [Online]. Available: https://doi.org/10.1007/BF02650179

[15] S. Gupta and R. Zia, “Quantum neural networks,” *Journal of Computer and System Sciences*, vol. 63, pp. 355–383, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000001917696

[16] K. He *et al.*, “Deep residual learning for image recognition,” *arXiv preprint arXiv:1512.03385*, 2015.

[17] S. Kak, “On quantum neural computing,” *Information Sciences*, vol. 83, pp. 143–160, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002002559400095S

[18] L. Lamport, R. Shostak, and M. Pease, “The byzantine generals problem,” *ACM Trans. Program. Lang. Syst.*, vol. 4, pp. 382–401, Jul. 1982. [Online]. Available: http://doi.acm.org/10.1145/357172.357176

[19] Y. Lecun *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[20] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[21] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[22] B. Ricks and D. Ventura, “Training a quantum neural network,” in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2003/file/505259756244493872b7709a8a01b536-Paper.pdf

[23] M. Schuld, I. Sinayskiy, and F. Petruccione, “The quest for a quantum neural network,” *Quantum Information Processing*, vol. 13, pp. 2567–2586, Nov 2014. [Online]. Available: https://doi.org/10.1007/s11128-014-0809-8

[24] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Review*, vol. 41, pp. 303–332, 1999. [Online]. Available: https://doi.org/10.1137/S0036144598347011

[25] A. Steane, “Quantum computing,” *Reports on Progress in Physics*, vol. 61, p. 117, 1998.

[26] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[27] A. Voulodimos *et al.*, “Deep learning for computer vision: A brief review,” *Computational Intelligence and Neuroscience*, vol. 2018, p. 7068349, Feb 2018. [Online]. Available: https://doi.org/10.1155/2018/7068349

[28] K. H. Wan *et al.*, “Quantum generalisation of feedforward neural networks,” *npj Quantum Information*, vol. 3, p. 36, Sep 2017. [Online]. Available: https://doi.org/10.1038/s41534-017-0032-4

[29] Q. Xia, Z. Tao, and Q. Li, “Defenses against byzantine attacks in distributed deep neural networks,” *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2020.

[30] Q. Xia and Q. Li, “Quantumfed: A federated learning framework for collaborative quantum training,” 2021.

[31] Q. Xia *et al.*, “Faba: An algorithm for fast aggregation against byzantine attacks in distributed neural networks,” in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4824–4830. [Online]. Available: https://doi.org/10.24963/ijcai.2019/670

[32] C. Xie, O. Koyejo, and I. Gupta, “Generalized byzantine-tolerant SGD,” *CoRR*, vol. abs/1802.10116, 2018. [Online]. Available: http://arxiv.org/abs/1802.10116

[33] ——, “Zeno++: Robust fully asynchronous {sgd},” 2020. [Online]. Available: https://openreview.net/forum?id=rygHe64FDS

[34] C. Xie, S. Koyejo, and I. Gupta, “Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance,” in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. Long Beach, California, USA: PMLR, 09–15 Jun 2019, pp. 6893–6901. [Online]. Available: http://proceedings.mlr.press/v97/xie19b.html

[35] D. Yin *et al.*, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5650–5659. [Online]. Available: http://proceedings.mlr.press/v80/yin18a.html

[36] H.-S. Zhong *et al.*, “Quantum computational advantage using photons,” *Science*, vol. 370, pp. 1460–1463, 2020. [Online]. Available: https://science.sciencemag.org/content/370/6523/1460