Understanding the Resource Cost of Fully Homomorphic Encryption in Quantum Federated Learning

Lukas Böhm[1,5] Arjhun Swaminathan[2,3] Anika Hannemann[1,4,5] Erik Buchmann[1,5]
[1]Department of Computer Science, Leipzig University
[2]Medical Data Privacy and Privacy-preserving Machine Learning (MDPPML), University of Tübingen
[3]Institute for Bioinformatics and Medical Informatics (IBMI), University of Tübingen
[4]School of Engineering, Zurich University of Applied Sciences
[5]Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI) Dresden/Leipzig, Leipzig University
Email: l.boehm@uni-leipzig.de, arjhun.swaminathan@uni-tuebingen.de, anika.hannemann@zhaw.ch, buchmann@informatik.uni-leipzig.de

***Abstract*—Quantum Federated Learning (QFL) enables distributed training of Quantum Machine Learning (QML) models by sharing model gradients instead of raw data. However, these gradients can still expose sensitive user information. To enhance privacy, homomorphic encryption of parameters has been proposed as a solution in QFL and related frameworks.**

In this work, we evaluate the overhead introduced by Fully Homomorphic Encryption (FHE) in QFL setups and assess its feasibility for real-world applications. We implemented various QML models including a Quantum Convolutional Neural Network (QCNN) trained in a federated environment with parameters encrypted using the CKKS scheme. This work marks the first QCNN trained in a federated setting with CKKS-encrypted parameters. Models of varying architectures were trained to predict brain tumors from MRI scans. The experiments reveal that memory and communication overhead remain substantial, making FHE challenging to deploy.

Keywords-Quantum Computing; Quantum Federated Learning; Full Homomorphic Encryption.

I. Introduction

Federated Learning (FL) [1] allows to train machine learning models in a distributed way. It is an iterative protocol in which multiple clients train local models using their own private datasets. Instead of sharing raw data, clients transmit the corresponding gradients to a central party, which then aggregates them to update a global model. FL plays a crucial role in ensuring compliance with data protection regulations.

Adapting classical algorithms to quantum environments is becoming increasingly relevant. Quantum methods may offer significant speedups in a variety of machine learning tasks compared to conventional ML techniques [2]. [3] has already demonstrated the feasibility of enabling FL workflows in a quantum environment using TensorFlow Quantum (TFQ) [4].

While FL circumvents direct data sharing, the weight parameters of local models can still potentially reveal sensitive information. In a classical federated learning setup, a semi-honest central party could exploit these weights to infer or extract specific data points, e.g., by leveraging *Deep Leakage from Gradient* [5], trying *Membership Inference Attacks* [6], or executing *Model Inversion Attacks* [6] if it has access to the final or an intermediate model.

Since Quantum Federated Learning (QFL) frameworks often transmit parameters over insecure networks, it is crucial to secure these parameters before transmission [3]. Options to protect clients against untrustworthy central parties include leveraging Differential Privacy (DP) [8][9], using Secure Multi-Party Computation (SMPC) [10], or employing Fully Homomorphic Encryption (FHE) [11]. FHE refers to encryption schemes that allow additions and multiplications to be performed directly on encrypted data, such that decrypting the results yields the same value as performing the corresponding algebraic operations on the original plain text. [12] posit that FHE inherently provides a robust solution to ensure privacy in federated learning. Notably, QFL encodes data as quantum states and may achieve the same accuracy with fewer parameters than traditional FL [13], potentially making FHE more feasible since fewer parameters need to be encrypted.

This work provides a robust implementation of FHE within a QFL setup, building upon the foundation laid by [14]. While their study employed simpler Quantum Machine Learning (QML) models, we implemented models without quantum layers, with simple quantum layers, and with a Quantum Convolutional Neural Network (QCNN) [15]. For feature extraction, two variations are tested: a self-trained Convolutional Neural Network (CNN) and a pre-trained ResNet-18 [16]. In addition, we analyzed the overhead introduced by FHE with in an experiment predicting tumors using MRI scans [17]. Thus, we make two contributions:

1) We describe an implementation of a QCNN within FL that incorporates FHE for parameter encryption.
2) We analyze the overhead caused by FHE in terms of training times, CPU/memory usage and communication by experimenting with six model types.

Our analysis revealed that integrating FHE into QFL results in substantial overhead. To mitigate this overhead, the number of parameters must be reduced, which has an impact on classification performance. Hence, introducing FHE into QFL comes with a trade-off between privacy and model complexity. Owing to space limitations, we refer to a preprint [18] for detailed statistics and discussions.

Paper structure: Sections II and III cover related work and background on QFL and FHE. Sections IV–VI present the setup, results, and discussion. Section VII concludes.

II. Related Works

Integrating homomorphic encryption techniques into conventional FL setups has gained significant attention in recent years, leading to numerous works [19]–[25]. For example, FedML-HE [23] provides an FL framework that selectively encrypts only the most sensitive gradients using CKKS [26]. This approach aims to reduce communication and computational overhead while maintaining the desired level of privacy. The authors demonstrated that encrypting the top 30% most sensitive parameters, along with the first and last layers, effectively mitigates inversion attacks. A very effective approach to employing CKKS in a FL setup was proposed by [27], though their method relies on hardware acceleration and advanced, GPU-based cryptographic implementations to mitigate the overhead of homomorphic encryption. In contrast, the present work focuses on a general, hardware-agnostic assessment of the resource requirements for FHE in FL and QFL environments.

Several methods have also been proposed to secure user data in QFL [13][28] setups. Examples include Quantum Secure Aggregation [29], which secures parameters by encoding them into qubits and transmitting them via quantum channels, and CryptoQFL [30], which uses Quantum One-Time Pads (QOTP) to encode quantum states. In CryptoQFL, the QOTP keys are homomorphically encrypted before transmission to the central party, offering a dual-layer security approach. Additionally, it is feasible to directly apply homomorphic encryption to user data within a quantum delegated framework [2].

However, the use of FHE to directly encrypt parameters in QFL setups remains largely unexplored. The most relevant work in this area is presented by [14], who provide a practical implementation of a QFL framework integrating CKKS encryption. They evaluated four different models: a conventional CNN and a CNN combined with simple quantum layers, each tested both with and without FHE. The models were assessed on multiple datasets, including CIFAR-10 [31], Brain MRI [17], and PCOS [32]. While the use of FHE resulted in longer training times, test accuracy was minimally affected. For example, the QNN model trained on the Brain MRI dataset achieved an accuracy of 88.75% with FHE compared to 89.71% without FHE. Training time increased from 110.6 minutes to 116.5 minutes when employing FHE. However, it is worth noting that the source code indicates not all layers of the model were encrypted.

In a nutshell, the integration of FHE into FL frameworks has been widely explored, but connections to models leveraging quantum capabilities remain underdeveloped or depend heavily on specific hardware and network configurations [14][21][24][33].

III. Background

This work simulates an environment where multiple clients collaboratively train QML models with the aid of a central party [34]. Following the FL protocol, the central party initializes gradients before the first training round. After receiving these gradients, each client trains a local model combining conventional and quantum layers. Before passing the updated gradients back to the central party, they are homomorphically encrypted. It is important to note that the central party only has access to the public key, whereas clients share the secret key. This approach prevents the central party, as well as any intermediary attempting to intercept the communicated data, from accessing plaintext gradients. The central party is then responsible for aggregating the weights and distributing them again among the clients. This process is typically repeated for multiple rounds. Figure 1 visualizes this setup.

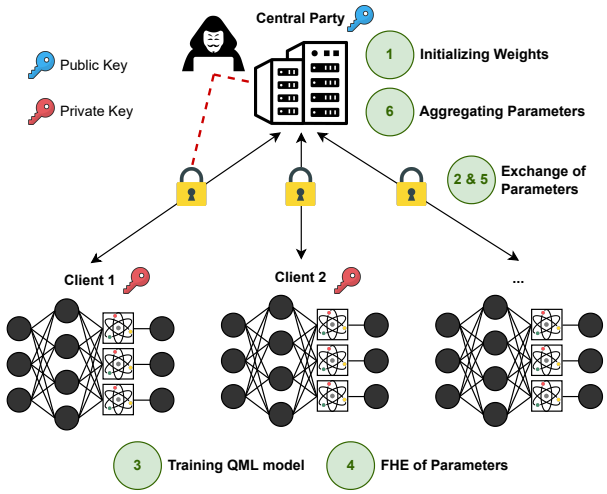

Figure 1. Overview of a QFL setup utilizing FHE for parameter encryption.

A. Variational Quantum Circuits

Within a QFL environment, clients have access to quantum computing resources, enabling them to collaboratively train models that combine classical and quantum layers. This is achieved through training Variational Quantum Circuits (VQCs) [13]. VQCs, also referred to as Quantum Neural Networks (QNNs), leverage qubit rotations to parametrize quantum circuits. The corresponding parameters control the extent of rotation, enabling a learnable unit block whose parameters can be trained similarly to weights in conventional neural networks [35]. The general idea of VQCs is that a quantum routine $E(x)$ transforms classical data x into quantum states. Subsequently, a quantum circuit $W(\phi)$, which depends on a set of parameters ϕ, is applied. In our simulations, these parameters are iteratively updated using conventional machine learning methods on a classical computer. Finally, the output of the quantum circuit is obtained by performing measurements [13].

A crucial component of this process is embedding classical data into qubits. To accomplish this, it is common to reduce the dimensionality of the input data to align with the number of available qubits. Subsequently, a *variational encoding* scheme is employed, utilizing input values as rotation angles for

single-qubit rotation gates. Given typical constraints on circuit depth and the number of qubits, hybrid learning schemes are often adopted to embed data into qubits. For example, a (pretrained) CNN can be employed to convert image data into a compact feature vector with significantly reduced dimensionality. The length of this feature vector is designed to match the number of qubits. Using the *variational encoding* scheme, this vector is then transformed into qubit states [13].

Alongside simpler VQCs, this work also leverages QCNNs, which are a specialized type of VQCs inspired by classical convolutional neural networks. QCNNs inherit a similar structure, utilizing convolutional and pooling layers before making predictions via qubit measurements.

B. CKKS

To secure the trained parameters shared with the central party, we employ CKKS [26] to homomorphically encrypt gradients. CKKS supports fast, approximate homomorphic computations on encrypted data, enabling arithmetic operations that yield results close to the true values. This approach is practical because, in many real-world applications, obtaining approximately correct results is sufficient.

Assuming a message m is encrypted, the encryption and decryption process begins by scaling the input message m with a global scale factor Δ, which manages precision. Due to rounding of the scaled value, this introduces a small approximation error. A typical value for Δ is 2^{40}. Since CKKS is based on the Ring Learning with Errors (RLWE) problem, the plaintext message must be encrypted using a secret key sk into a polynomial in the ring

$$\mathcal{R}_Q := \mathcal{R}/Q\mathcal{R} \quad \text{with} \quad \mathcal{R} := \mathbb{Z}[X]/(X^N + 1). \tag{1}$$

Here, N is a power of two and determines the polynomial degree. This parameter N plays a key role in managing precision and capacity: a larger N allows encoding of more data slots or larger numbers, which reduces information loss and improves accuracy. The ring $\mathcal{R}_Q$ consists of polynomials whose coefficients are reduced modulo the integer Q. Importantly, Q is not a single large modulus but a product of multiple moduli, often referred to as levels:

$$Q = q_0 \times q_1 \times \cdots \times q_i, \quad i \in \mathbb{N}_0. \tag{2}$$

Alongside the global scale Δ and the polynomial degree N, the sizes and structure of these moduli q_i are crucial parameters for CKKS, as they directly influence precision, noise growth, and the number of homomorphic operations that can be securely performed. Other important concepts in CKKS include rescaling, which manages the multiplicatively growing global scale Δ and noise level, as well as relinearization, which mitigates the increasing size (or degree) of ciphertexts resulting from multiplications.

IV. Experimental Setup

The experiments were implemented in Python, leveraging TenSEAL [36] for homomorphic encryption, Flower [37] for federated learning, and Pennylane [38] for quantum simulation. Gradients were aggregated using FedAvg [1]. All experiments were conducted on a high-performance computing cluster running Rocky Linux 8. Each evaluation was executed on a single compute node, requesting 48 logical CPU cores of type AMD EPYC 7551P (2.0–3.0 GHz) and 200 GB of ECC DDR4 memory.

A. Model Types and Quantum Integration

We analyzed six model types:

- `cnn`: A self-trained CNN without any quantum layers.
- `cnn-qnn`: A self-trained CNN followed by 6 Basic Entangled Layers [39] tailored for 4 qubits.
- `cnn-qcnn`: A self-trained CNN followed by a QCNN using 8 qubits.
- `resnet18`: A pre-trained ResNet-18 without any quantum layers.
- `resnet18-qnn`: A pre-trained ResNet-18 followed by 6 Basic Entangled Layers tailored for 4 qubits.
- `resnet18-qcnn`: A pre-trained ResNet-18 followed by a QCNN using 8 qubits.

In the following, model names with prefix *FHE* (e.g., `FHE-cnn`) denote experiments with FHE, and model names with prefix *Standard* indicate experiments without FHE. The models have a similar number of parameters, ranging between 2,052 and 2,241, except for the `resnet18-qcnn` model, which has 4,145 parameters due to additional parameters required to combine the pre-trained model with the QCNN.

When quantum layers were integrated, most parameters were used for feature extraction. For example, the actual QCNN model requires only 21 parameters; the remainder corresponds to a conventional model that reduces dimensionality to enable embedding image data into qubits. In line with [13], we use a pre-trained ResNet-18 model for feature extraction instead of training a separate CNN from scratch. In this case, only one layer needs to be tuned and sent over the network. Thus, combining ResNet-18 with quantum layers enabled a complex model for feature extraction while keeping the number of parameters low for using FHE.

We ran experiments with each model type five times with FHE, and five times without, resulting in a total of 60 runs. All experiments consisted of one central party and 20 clients, except for the large ResNet-18. Those experiments had to be limited to a single client, when using FHE and a QCNN. Clients used a batch size of 32 and a learning rate of 0.001 to train the model for 20 rounds with 10 epochs each.

B. Dataset

The experiments use the Brain Tumor MRI dataset [17], as it is a common benchmark and the medical domain is privacy-sensitive. It contains a total of 7,023 Magnetic Resonance Imaging (MRI) scans, each categorized into one of four classes: glioma, meningioma, pituitary tumor or no tumor. The class distribution is shown in Table I.

The Brain Tumor MRI dataset is already split into training and test sets. We assign the test dataset to the central party. If

TABLE I. CLASS DISTRIBUTION OF MRI DATASET

Class	Train	Test
glioma	1321	300
meningioma	1339	306
pituitary	1457	300
no tumor	1595	406

parameters are not encrypted, the central party can use these samples to compute accuracy and loss. The training dataset is distributed among the clients, with train-validation split of 90/10 for local training and validation. All images were normalized and resized to 224×224 pixels.

V. RESULTS

Owing to space limitations, we present a condensed set of results in this paper. Comprehensive results and detailed statistics are available in a preprint [18].

A. Training and Parameter Aggregation Times

Time-related metrics are common to measure overhead [14][21][24][33]. Employing FHE to encrypt roughly 2,000 parameters reliably increases training time as shown in Table II. For instance, `Standard-cnn` required a median training time of 205 minutes, while running the same model with FHE resulted in a median training time of 240 minutes—an increase of 17.07%. Similarly, the `cnn-qnn` models exhibited a 11.41% increase in training time, and the `cnn-qcnn` models required 3.82% more training time. It stands out that the `cnn-qcnn` and `resnet18-qcnn` models have significantly longer training durations. This is because these models require 8 qubits instead of 4, and quantum computations were simulated on CPU. Runtimes are expected to be differ when executed on an actual quantum computer. The `FHE-resnet18-qcnn` model required a median training time of 462.30 minutes, a similar value to `FHE-resnet18-qnn`, even though it used only one client.

The median client round time shows only moderate increases. In some cases, it even decreases slightly when applying FHE. For example, the `Standard-resnet18` model has a median client round time of 17.82 minutes, whereas the corresponding FHE model requires only 17.50 minutes. In contrast, the central party round time consistently increases with FHE. This suggests that most of the increase in training time is due to central party-specific tasks such as parameter aggregation. This assumption is further supported by the observed encryption, decryption, and parameter aggregation times that are displayed in Table III. Encryption and decryption of all parameters is very fast, taking between one and two seconds. However, performing calculations on homomorphically encrypted data is expensive. Models without FHE typically require less than one second to aggregate parameters, as the central party simply calculates a weighted average of the 20 input matrices received from the clients. When using FHE, however, models that leverage 20 clients report a median aggregation time between 56.18 and 63.58 seconds.

TABLE II. MEDIAN VALUES FOR TOTAL TRAINING TIME, CLIENT ROUND TIME, AND CENTRAL PARTY ROUND TIME (IN MINUTES) BY MODEL.

Model	Total Time	Client Round Time	CP Round Time
Standard-cnn	205.06	9.74	9.99
FHE-cnn	239.86	9.93	10.63
Standard-cnn-qnn	292.01	13.85	14.23
FHE-cnn-qnn	325.33	14.12	14.82
Standard-cnn-qcnn	682.40	32.50	33.53
FHE-cnn-qcnn	708.45	32.75	33.72
Standard-resnet18	375.02	17.82	17.99
FHE-resnet18	401.26	17.50	18.65
Standard-resnet18-qnn	446.67	21.49	21.85
FHE-resnet18-qnn	484.15	21.95	22.68
Standard-resnet18-qcnn	848.01	40.91	41.63
FHE-resnet18-qcnn*	462.30	22.26	22.62

* The `FHE-resnet18-qcnn` model was trained with only one client.

TABLE III. MEDIAN VALUES FOR ENCRYPTION TIME, DECRYPTION TIME, AND PARAMETER AGGREGATION TIME (IN SECONDS) BY MODEL.

Model	Enc.	Dec.	Param. Agg.
Standard-cnn	-	-	0.03
FHE-cnn	1.68	1.54	58.45
Standard-cnn-qnn	-	-	0.03
FHE-cnn-qnn	1.53	1.55	60.06
Standard-cnn-qcnn	-	-	0.03
FHE-cnn-qcnn	1.52	1.64	63.58
Standard-resnet18	-	-	0.01
FHE-resnet18	1.34	1.51	56.18
Standard-resnet18-qnn	-	-	0.016
FHE-resnet18-qnn	1.32	1.56	57.85
Standard-resnet18-qcnn	-	-	0.016
FHE-resnet18-qcnn*	1.29	2.48	7.47

* The `FHE-resnet18-qcnn` model was trained with only one client.

B. RAM and CPU Usage

FHE is very memory intensive: Models without FHE required roughly 1 GiB of real RAM on the client side based on median values, as shown in Table IV. All models except `FHE-resnet18-qcnn` have approximately 2,000 parameters. Applying FHE increases the median RAM usage of clients to ~4 GiB. For example, for `resnet18`, employing FHE increased the median RAM usage from 968.88 MiB to 4,390.12 MiB, representing a 353% increase. `FHE-resnet18-qcnn`, which has 4,145 parameters, reports an even higher median RAM usage of 7,555.04 MiB. Interestingly, the presence of quantum layers does not appear to significantly affect real memory usage of clients.

On the central party side, applying FHE has an even stronger effect on real memory usage (see Table IV). While the central party requires less than 1 GiB when training a model without FHE, this value skyrockets to over 50 GiB when FHE is employed. For instance, `Standard-resnet18-qnn`

requires 914.64 MiB based on median values, whereas `FHE-resnet18-qnn` requires 51.35 GiB — an increase of 5,648.44%. Even `FHE-resnet18-qcnn` reported a substantial increase of 1,355.51% (from 915.58 MiB to 13,326.36 MiB), despite `Standard-resnet18-qcnn` training with 19 more clients.

TABLE IV. MEDIAN VALUES FOR REAL RAM USAGE (IN MIB) BY MODEL FOR CLIENTS AND CENTRAL PARTY.

Model	Client RAM	CP RAM
Standard-cnn	930.04	816.50
FHE-cnn	3,638.55	51,454.87
Standard-cnn-qnn	962.34	821.91
FHE-cnn-qnn	3,704.29	52,613.89
Standard-cnn-qcnn	1,041.23	824.47
FHE-cnn-qcnn	3,857.98	54,598.11
Standard-resnet18	968.88	911.90
FHE-resnet18	4,390.12	51,495.80
Standard-resnet18-qnn	971.16	914.64
FHE-resnet18-qnn	4,375.12	52,577.55
Standard-resnet18-qcnn	984.95	915.58
FHE-resnet18-qcnn*	7,555.04	13,326.36

* The `FHE-resnet18-qcnn` model was trained with only one client.

FHE does not appear to have an impact on client CPU usage. Each client typically used around two full CPU cores. However, models employing FHE often exhibit higher maximum CPU usage values compared to their non-FHE counterparts. Since the central party spends most of its time waiting for clients, its overall CPU usage is generally low, except for parameter aggregation. Figure 2 illustrates this behavior for two exemplary runs of the `cnn` model.

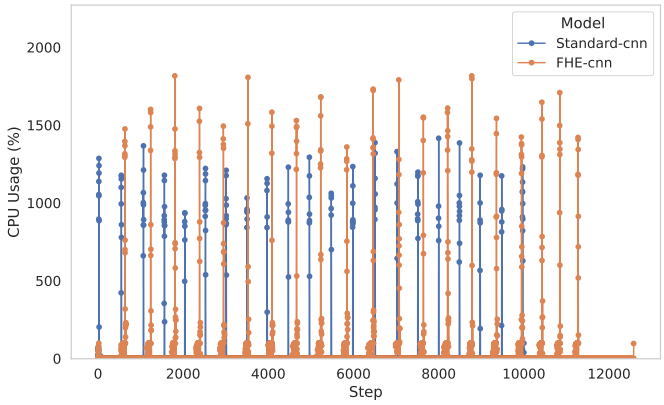

Figure 2. Central party CPU usage over two exemplary runs: `FHE-cnn` and `Standard-cnn`. Each step on the x-axis represents a data point where the metric was recorded during the training process. A value of 100% on the y-axis represents one CPU core under full load.

C. Communication Overhead

Applying FHE causes a large communication overhead, as shown in Table V. For example, the parameter size of `Standard-cnn` models at the beginning of each training round is approximately 9 KiB. When encrypted, this size skyrockets to 258.76 MiB, representing an increase of 2,875,680%. With 20 clients, the central party receives around 13 GiB of data each round instead of just 0.18 MiB. Models with a similar number of parameters show comparable results; this includes `cnn-qnn`, `cnn-qcnn`, `resnet18`, and `resnet18-qnn`. Because `FHE-resnet18-qcnn` was trained with 4,145 parameters and a single client, the bytes sent each round doubles, while the received package size is lower in total but higher relative to the number of clients.

TABLE V. MEDIAN VALUES FOR COMMUNICATION OVERHEAD (IN MIB) BY MODEL DURING ONE TRAINING ROUND. PLEASE NOTE THAT MIB SENT REFERS TO THE PARAMETER SIZE SENT TO EACH INDIVIDUAL CLIENT, WHEREAS MIB RECEIVED DEPENDS ON THE TOTAL NUMBER OF CLIENTS.

Model	MiB Sent (per Client)	MiB Received (Total)
Standard-cnn	0.01	0.18
FHE-cnn	258.76	13,185.30
Standard-cnn-qnn	0.01	0.19
FHE-cnn-qnn	264.27	13,465.86
Standard-cnn-qcnn	0.01	0.20
FHE-cnn-qcnn	280.41	14,288.30
Standard-resnet18	0.01	0.16
FHE-resnet18	256.76	13,083.33
Standard-resnet18-qnn	0.01	0.17
FHE-resnet18-qnn	262.26	13,363.84
Standard-resnet18-qcnn	0.02	0.33
FHE-resnet18-qcnn*	518.65	1,321.42

* The `FHE-resnet18-qcnn` model was trained with only one client.

D. Classification Performance

Considering classification metrics shown in Table VI, it becomes clear that drastically reducing the number of parameters diminishes accuracy. This is reasonable, since this work leveraged models with only 2,068 to 4,145 parameters. This illustrates the trade-off between privacy and model complexity introduced by FHE. If higher privacy standards necessitate the use of FHE, one might also reduce model complexity, which negatively impacts accuracy.

What stands out are the even lower values for central accuracy. The `Standard-cnn` reports the highest value in this category, at only 65.55%. We assume that these values would be higher if fewer clients were used. However, this metric serves as a good indicator of overall model performance since it tests the model using a separate hold-out dataset at the central party level. It might also indicate a performance drop compared to centralized learning. Applying FHE itself, however, only slightly impacted classification metrics.

Using a ResNet-18 model without quantum layers yielded promising results. The `Standard-resnet18` model achieved an aggregated F1-score of 0.89. When FHE was applied, the F1-score remained stable, and the aggregated accuracy even improved slightly. However, the central accuracy stayed low at 52.63%. This represents a partial success,

TABLE VI. MEDIAN VALUES OF CENTRAL ACCURACY (IN %), AGGREGATED ACCURACY (IN %), AND AGGREGATED F1-SCORE BY MODEL.

Model	Central Accuracy	Agg. Accuracy	Agg. F1
Standard-cnn	65.55	72.54	0.71
FHE-cnn	–	71.65	0.71
Standard-cnn-qnn	61.05	68.63	0.67
FHE-cnn-qnn	–	70.78	0.69
Standard-cnn-qcnn	61.43	68.26	0.66
FHE-cnn-qcnn	–	70.77	0.69
Standard-resnet18	52.63	90.20	0.89
FHE-resnet18	–	90.55	0.89
Standard-resnet18-qnn	44.21	46.00	0.44
FHE-resnet18-qnn	–	76.81	0.74
Standard-resnet18-qcnn	39.48	66.65	0.62
FHE-resnet18-qcnn*	–	76.86	0.75

* The `FHE-resnet18-qcnn` model was trained with only one client.

demonstrating that a model leveraging FHE can be trained while maintaining moderate overhead and high classification metrics. Combining the ResNet-18 model with quantum layers did not fulfill this promise, as those models struggled to converge and hence reported low classification metrics. It is important to note that results would likely differ if quantum computations were not simulated on a CPU.

VI. DISCUSSION

Due to space limitations, we refer to our preprint [18] for a detailed discussion.

A. Findings

The overhead introduced by FHE is considerable. Median training time typically increased by 20 to 30 minutes when parameter encryption was employed, primarily due to longer parameter aggregation times on the central party. However, the use of quantum layers had an even more pronounced impact on training duration. This might change when real quantum computers are available.

FHE results in a massive memory overhead and higher CPU usage peaks. Client side memory increased from 1 to 4 GiB per client, when encrypting around 2,000 parameters using FHE. This makes FHE impractical for cross-device setups. We tested only models with a low parameter count. Even small fully connected layers of more complex models can have millions of trainable parameters. This highlights the challenges of deploying QFL setups with FHE. On the central party, the required memory surged from less than 1 GiB of RAM (20 clients, no FHE) to over 50 GiB with FHE.

The effect on communication overhead is similar. Without FHE, the central party sent and received ≈0.02 MiB per parameter to/from each client. With FHE, this jumps to between 256.76 MiB and 280.41 MiB. For models with ≈2,000 parameters, the central party received ≈13 GiB every round.

B. Limitations

This work employed TenSEAL to enable CKKS for parameter encryption, and used the library's default values [40]. An optimal parameter selection might reduce the overhead, similar to FedSHE [24]. Furthermore, clients and central partys were simulated on the same machine. This is sufficient to assess the practical impact of FHE, but using dedicated hosts would allow more accurate measurements.

The quantum models in this study were simulated on classical hardware. Real quantum devices could have an impact on aspects such as gradient estimation or state collapse upon measurement, which is not captured in our simulations. Thus, our findings support conclusions the regarding FHE overhead in (Q)CNNs, but do not allow to assess quantum machine learning models.

VII. CONCLUSION

This work provides an extensive overview of the overhead and implications introduced by FHE when integrated into a QFL framework. In a scenario involving distributed training on medical data, multiple models were trained to classify brain MRI scans, encompassing a variety of classical and quantum machine learning models. The experiments showed that leveraging FHE in QFL setups introduces excessive memory and communication overhead, making it potentially impractical for many real-world scenarios. This especially applies to cross-device setups. To mitigate this issue, it is desirable to keep the number of trainable parameters low, although this impacts classification performance. Hence, this work demonstrates a clear trade-off between data privacy and model complexity when applying FHE. This underlines the need for more sophisticated methods to apply FHE effectively in QFL setups.

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, A. Singh and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, PMLR, Apr. 2017, pp. 1273–1282.

[2] W. Li and D.-L. Deng, "Quantum delegated and federated learning via quantum homomorphic encryption," *Research Directions: Quantum Technologies*, vol. 3, e3, 2025. DOI: 10.1017/qut.2025.2.

[3] M. Chehimi and W. Saad, "Quantum federated learning with quantum data," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Singapore, Singapore, 2022, pp. 8617–8621. DOI: 10.1109/ICASSP43922.2022.9746622.

[4] M. Broughton *et al.*, *Tensorflow quantum: A software framework for quantum machine learning*, 2021. arXiv: 2003.02989 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2003.02989 (visited on 08/22/2025).

[5] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, H. Wallach *et al.*, Eds., vol. 32, Curran Associates, Inc., 2019.

[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 3–18. DOI: 10.1109/SP.2017.41.

[8] M. Abadi *et al.*, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, Vienna, Austria: ACM, 2016, pp. 308–318, ISBN: 9781450341394. DOI: 10.1145/2976749.2978318.

[9] M. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Advances in Neural Information Processing Systems*, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., vol. 23, Curran Associates, Inc., 2010.

[10] O. Goldreich, "Secure multi-party computation," *Manuscript. Preliminary version*, vol. 78, no. 110, pp. 1–108, 1998.

[11] C. Gentry, "A fully homomorphic encryption scheme," PhD thesis, Stanford University, Stanford, CA, 2009.

[12] S. Pati *et al.*, "Privacy preservation for federated learning in health care," *Patterns*, vol. 5, no. 7, p. 100 974, 2024, ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2024.100974.

[13] S. Y.-C. Chen and S. Yoo, "Federated quantum machine learning," *Entropy*, vol. 23, no. 4, 2021, ISSN: 1099-4300. DOI: 10.3390/e23040460.

[14] S. Dutta *et al.*, *Federated learning with quantum computing and fully homomorphic encryption: A novel computing paradigm shift in privacy-preserving ml*, 2024. arXiv: 2409.11430 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2409.11430.

[15] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Physics*, vol. 15, no. 12, pp. 1273–1278, 2019. DOI: 10.1038/s41567-019-0648-8.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[17] M. Nickparvar, *Brain tumor mri dataset*, 2021. DOI: 10.34740/KAGGLE/DSV/2645886.

[18] L. Böhm, A. Swaminathan, A. Hannemann, and E. Buchmann, *Understanding the resource cost of fully homomorphic encryption in quantum federated learning*, 2026. arXiv: 2603.02799 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2603.02799.

[19] F. Qiu, H. Yang, L. Zhou, C. Ma, and L. Fang, "Privacy preserving federated learning using ckks homomorphic encryption," in *Wireless Algorithms, Systems, and Applications*, L. Wang, M. Segal, J. Chen, and T. Qiu, Eds., Cham: Springer Nature Switzerland, 2022, pp. 427–440, ISBN: 978-3-031-19208-1.

[20] P. Yao, H. Wang, C. Zheng, J. Yang, and L. Wang, "Efficient federated learning aggregation protocol using approximate homomorphic encryption," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2023, pp. 1884–1889. DOI: 10.1109/CSCWD57460.2023.10152829.

[21] N. M. Hijazi, M. Aloqaily, M. Guizani, B. Ouni, and F. Karray, "Secure federated learning with fully homomorphic encryption for iot communications," *IEEE Internet of Things Journal*, vol. 11, no. 3, pp. 4289–4300, 2024. DOI: 10.1109/JIOT.2023.3302065.

[22] H. Fang and Q. Qian, "Privacy preserving machine learning with homomorphic encryption and federated learning," *Future Internet*, vol. 13, no. 4, 2021, ISSN: 1999-5903. DOI: 10.3390/fi13040094.

[23] W. Jin *et al.*, *Fedml-he: An efficient homomorphic-encryption-based privacy-preserving federated learning system*, 2024. arXiv: 2303.10837 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2303.10837.

[24] Y. Pan *et al.*, "Fedshe: Privacy preserving and efficient federated learning with adaptive segmented ckks homomorphic encryption," *Cybersecurity*, vol. 7, no. 1, p. 40, 2024.

[25] A. Hannemann and E. Buchmann, "Is homomorphic encryption feasible for smart mobility?" In *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, IEEE, 2023, pp. 523–532.

[26] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds., Cham: Springer International Publishing, 2017, pp. 409–437, ISBN: 978-3-319-70694-8.

[27] L. Jiang and L. Ju, *Fhebench: Benchmarking fully homomorphic encryption schemes*, 2022. arXiv: 2203.00728 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2203.00728.

[28] N. Innan, M. A.-Z. Khan, A. Marchisio, M. Shafique, and M. Bennai, "Fedqnn: Federated learning using quantum neural networks," in *2024 International Joint Conference on Neural Networks (IJCNN)*, 2024, pp. 1–9. DOI: 10.1109/IJCNN60899.2024.10651123.

[29] Y. Zhang *et al.*, "Federated learning with quantum secure aggregation," *Cornell University - arXiv*, 2022. DOI: 10.48550/arxiv.2207.07444.

[30] C. Chu, L. Jiang, and F. Chen, "Cryptoqfl: Quantum federated learning on encrypted data," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 1231–1237. DOI: 10.1109/QCE57702.2023.00139.

[31] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[32] M. Hub *et al.*, "Auto-pcos classification challenge," *Authorea Preprints*, 2024.

[33] S. Dutta, N. Innan, S. B. Yahia, M. Shafique, and D. E. B. Neira, *Mqfl-fhe: Multimodal quantum federated learning framework with fully homomorphic encryption*, 2025. arXiv: 2412.01858 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2412.01858.

[34] A. Swaminathan and M. Akgün, "Distributed and secure kernel-based quantum machine learning," *Transactions on Machine Learning Research*, 2025.

[35] D. Gurung, S. R. Pokhrel, and G. Li, *Quantum federated learning: Analysis, design and implementation challenges*, 2023. DOI: 10.48550/arXiv.2306.15708. arXiv: 2306.15708 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2306.15708 (visited on 08/22/2025).

[36] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, *TenSEAL: A Library for Encrypted Tensor Operations Using Homomorphic Encryption*, arXiv:2104.03152 [cs], Apr. 2021. DOI: 10.48550/arXiv.2104.03152. [Online]. Available: http://arxiv.org/abs/2104.03152 (visited on 05/11/2025).

[37] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," *CoRR*, vol. abs/2007.14390, 2020. arXiv: 2007.14390. [Online]. Available: https://arxiv.org/abs/2007.14390.

[38] V. Bergholm *et al.*, *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2022. arXiv: 1811.04968 [quant-ph]. [Online]. Available: https://arxiv.org/abs/1811.04968 (visited on 08/22/2025).

[39] Y. Idan and M. N. Jayakody, *Variational quantum algorithm based circuit that implements the toffoli gate with multi inputs*, 2023. arXiv: 2305.18750 [quant-ph]. [Online]. Available: https://arxiv.org/abs/2305.18750.

[40] *OpenMined/TenSEAL*, Version 0.3.15, May 2025. [Online]. Available: https://github.com/OpenMined/TenSEAL (visited on 10/28/2025).