Quantum Machine Intelligence (2025) 7:76
https://doi.org/10.1007/s42484-025-00295-z

RESEARCH ARTICLE

A hybrid quantum neural network for split learning

Hevish Cowlessur[1,2] · **Chandra Thapa**[2] · **Tansu Alpcan**[1] · **Seyit Camtepe**[2]

Received: 8 August 2024 / Accepted: 30 June 2025

Abstract
Quantum machine learning (QML) is an emerging field of research with potential applications to distributed collaborative learning, such as split learning (SL). SL allows resource-constrained clients to collaboratively train ML models with a server, reduce their computational overhead, and enable data privacy by avoiding raw data sharing. Although QML with SL has been studied, the problem remains open in resource-constrained environments where clients lack quantum computing capabilities. Additionally, data privacy leakage between client and server in SL poses risks of reconstruction attacks on the server side. To address these issues, we propose hybrid quantum split learning (HQSL), an application of hybrid QML in SL. HQSL enables classical clients to train models with a hybrid quantum server and curtails reconstruction attacks. Additionally, we introduce a novel qubit-efficient data-loading technique for designing a quantum layer in HQSL, minimizing both the number of qubits and circuit depth. Evaluations on real hardware demonstrate HQSL's practicality under realistic quantum noise. Experiments on five datasets demonstrate HQSL's feasibility and ability to enhance classification performance compared to its classical models. Notably, HQSL achieves mean improvements of over 3% in both accuracy and F1-score for the Fashion-MNIST dataset and over 1.5% in both metrics for the Speech Commands dataset. We expand these studies to include up to 100 clients, confirming HQSL's scalability. Moreover, we introduce a noise-based defense mechanism to tackle reconstruction attacks on the server side. Overall, HQSL enables classical clients to train collaboratively with a hybrid quantum server, improving model performance and resistance against reconstruction attacks.

Keywords Hybrid quantum neural network · Qubit-efficient data-loading technique · Split learning · Data privacy leakage · Reconstruction attacks

1 Introduction

Quantum machine learning (QML) is a rapidly evolving research area at the intersection of quantum computing and machine learning (ML). A few examples of quantum analogs to some classical ML algorithms are quantum support vector machines (Li et al. 2015), quantum neural networks (QNN) (Jia et al. 2019), and quantum principal component analysis (Lloyd et al. 2014). These leverage the unique properties of qubit systems, such as superposition and entanglement, to study the potential of improving ML tasks using quantum computing. While fault-tolerant quantum computers are still a few years away, a practically feasible route for addressing real-world problems in the noisy-intermediate scale quantum (NISQ) era (Preskill 2018) is by adopting a hybrid approach. The hybrid approach integrates classical deep neural networks with the quantum layers, i.e., hybrid quantum neural network (HQNN) (Liang et al. 2021; Liu et al. 2021b). However, there are few works on HQNN, and it remains open in various aspects, including a resource-constrained distributed learning environment such as ML in the internet of things domain.

In distributed collaborative learning, split learning (SL) (Vepakomma et al. 2018) is a popular technique in resource-constrained environments where clients have low computing power. The ML model is split into client and server components, where the client trains the initial few layers on local data, and then the server continues training the remaining

✉ Hevish Cowlessur
hcowlessur@student.unimelb.edu.au

Chandra Thapa
chandra.thapa@data61.csiro.au

Tansu Alpcan
tansu.alpcan@unimelb.edu.au

Seyit Camtepe
seyit.camtepe@data61.csiro.au

[1] Department of Electrical and Electronic Engineering, University of Melbourne, Parkville 3010, VIC, Australia

[2] Data61, CSIRO, Sydney, NSW, Australia

Published online: 01 August 2025

layers. SL allows collaborative training without sharing raw data. In the quantum computing domain, quantum split learning (QSL) splits a QNN to leverage SL advantages (Park et al. 2023). However, in resource-constrained NISQ era environments, it is crucial to allow clients (no quantum computing capabilities) to compute in the classical domain. This motivates HQNN in SL, which has not been studied before. As SL can suffer from data privacy leakage[1], there is a risk of reconstruction attacks on the server side if the server is honest but curious (Joshi et al. 2022; Li et al. 2022). Moreover, this security aspect has not been investigated in HQNN with SL.

We present a novel (to the best of our knowledge, the first) application of HQNN to SL, which we call hybrid quantum split learning (HQSL), and investigate its resistance against reconstruction attacks due to data privacy leakage. In a simple HQSL setup, the HQNN model is divided into two parts: the client-side model, which has only classical neural network layers, and the server-side model, which consists of a quantum layer followed by classical neural network layers. This design reflects realistic deployment settings, where clients are typically resource-constrained (e.g., edge or IoT devices) and can only support a small number of lightweight classical layers, while compute-intensive operations and all quantum processing are offloaded to the more capable server. The quantum layer consists of a quantum circuit that converts classical inputs into quantum states, performs quantum computations, and outputs classical data. Considering the limitations of NISQ computers, which restrict the number of qubits and circuit size, we employ a low-depth 2-qubit quantum circuit within the quantum layer. This also ensures our simulations have reasonable runtimes. To allow a relatively larger dimension of classical data inputs into our quantum circuit, we propose a qubit-efficient data-loading technique. To tackle data privacy leakage in HQSL, we integrate a noise layer based on Laplacian distribution into the client-side model. In contrast to existing works on the Laplacian noise layer, we tune the noise parameters considering the rotational properties of the quantum layer's encoding gates. This paper is also the first, to the best of our knowledge, to study reconstruction attacks in the hybrid quantum-classical domain.

Overall, our contributions are the following:

1. *Feasibility and performance of HQSL with a single client:* We propose a novel HQSL architecture by combining SL with HQNN. This allows resource-limited classical clients to collaboratively train HQNN models with a hybrid quantum server featuring a 2-qubit quantum circuit layer followed by classical dense layers. Despite the small size of the quantum circuit, we successfully showcase a modest improvement in the test accuracy and F1-score over the classical counterpart of HQSL by conducting extensive experiments with five standard datasets as a proof of concept. This demonstrates the feasibility and superior performance of HQSL over SL. We further validate the practicality of HQSL deployment by evaluating its performance on multiple real quantum devices and noisy simulators.
2. *Quantum circuit design:* We devise a qubit-efficient data-loading scheme, consisting of only 2 qubits and 3 layers of encoding and parameterized gates corresponding to a 3-dimensional input. Through experimental analyses on the Fashion-MNIST dataset (Xiao et al. 2017), we show that HQSL with our proposed circuit in its quantum layer features a smaller simulation runtime and better classification performance than deeper quantum circuits.
3. *HQSL with multiple clients:* We extend our studies of HQSL by including up to 100 clients, illustrating its scalability. This advancement highlights HQSL's ability to support numerous classical clients in collaborative HQNN model training.
4. *Resistance to reconstruction attacks:* We study the resistance of HQSL against reconstruction attacks happening at inference time on the server side and propose a quantum encoding gate-based Laplacian noise layer defense mechanism as a countermeasure. A comparative study on reconstruction attacks in hybrid and classical settings demonstrates the higher resilience of HQSL compared to SL.

The rest of this paper is organized as shown in Table 1.

2 Related work

This section reviews relevant works on hybrid QML, SL, and quantum split learning. We discuss the need for combining

Table 1 Table of the organization of the paper in sections

Section 2	Related work
Section 3	Problem statement and research questions
Section 4	HQSL architecture, quantum circuit design and selection process, HQSL model training with single and multiple clients
Section 5	Experiments and findings with HQSL involving single and multiple clients
Section 6	Enhancing the resistance of HQSL against reconstruction attacks, proposed defense mechanism, experiments to test defense mechanism
Section 7	Conclusion and future work

[1] Data privacy leakage in split learning is the risk of private information being revealed or reconstructed from exchanged information during collaborative training, despite not sharing raw data directly.

classical and quantum computing in the NISQ era and describe the SL scheme and its study in the pure quantum domain.

Combining classical and quantum resources Currently available quantum computers exhibit noise, due to numerous factors, e.g., qubit decoherence and gate errors, constraining the complexity of quantum algorithms. Thus, a common practice to leverage the power of quantum computations in the NISQ era is by combining it with classical methods. Hybrid QML models, e.g., hybrid quantum neural network (HQNN), are becoming a common area of research, where a quantum computer computes only part of the model. For instance, in HQNNs, classical and quantum computing nodes are concatenated, with the classical layers reducing high-dimensional data to low dimensions to suit small quantum circuits that make up the quantum layer (Schetakis et al. 2022; Alam et al. 2021). This approach is particularly important for current-generation quantum hardware, which is constrained by the number of available qubits, circuit size, and depth. Moreover, in the generative adversarial network (GAN) domain, notable works have proposed the use of quantum circuits with classical neural network layers that aid in pre-processing (Liu et al. 2021a) and post-processing (Tsang et al. 2023) giving rise to hybrid quantum-classical GANs, which further highlight the versatility of quantum resources in hybrid architectures. In this work, we propose, for the first time to the best of our knowledge, yet another combination of quantum and classical resources in the form of *an HQNN for the split learning domain*.

Split learning (SL) and the data privacy leakage in SL In the distributed learning domain, SL has addressed computational requirement problems with federated learning (FL). In FL, each client has to run the entire ML model, which can be too computationally intensive for resource-constrained clients (Thapa et al. 2022). By offloading the compute-intensive parts of the model, SL enables more efficient resource allocation on the client devices, benefiting resource-constrained clients.

However, while SL prevents the sharing of clients' raw data with the server, information leakage still occurs from smashed data[2]. This is a concerning security issue because the smashed data contains latent information about the raw input data and can be used to stage an input data reconstruction attack (Abuadbba et al. 2020) at the server side. In the literature, several countermeasures have been proposed against data privacy leakage in SL. Joshi et al. (2022) proposed increasing the number of layers in the client-side model. This reduces the mutual information, hence reducing the similarity between the smashed data and raw data. The downside of this technique is that it increases the computational overhead on the client side, which can be problematic for lightweight devices. NoPeek, an information-theoretic measure to reduce information leakage, is another method suggested (Vepakomma et al. 2020) to reduce data privacy leakage. They do so by augmenting the classification loss function with the distance correlation function to reduce information leakage. However, a privacy-utility tradeoff is a possible consequence of NoPeek. Other methods involve differential privacy (Gawron and Stubbings 2022) and homomorphic encryption, which come at the cost of model performance degradation (Shokri and Shmatikov 2015) and too computationally intensive, respectively. In this work, we investigate the problem of data privacy leakage in the hybrid quantum split learning domain and *propose a novel noise-based defense mechanism designed considering the rotational properties of single-qubit encoding gates.* The methodology, experiments, and results are presented in Section 6.

Quantum split learning (QSL) In a pure QML framework, splitting a QNN has been trialed to examine the potential of QSL in preserving privacy and enhancing accuracy compared to quantum federated learning (QFL) (Park et al. 2023). However, this work assumed a fault-tolerant regime whereby client devices are equipped with quantum capabilities—an unrealistic assumption given the current generation of quantum computers and resource-constrained clients. To address this, we propose a solution that allows clients to compute in the classical domain while the server operates in the hybrid quantum domain. This relaxation of the assumption ensures a more practical approach.

3 Problem statement

In this section, we define our research problem by breaking it down into specific research questions. This method enables us to systematically tackle the complexities of our study and offers a structured framework for our investigation.

RQ 1: How can we leverage SL concepts to bring advantages of quantum computing to resource-constrained clients without quantum computing capabilities?—addressed in Sections 4 and 5.

Applying SL concepts in a pure QML model, e.g., splitting a QNN between a client and a server, assumes quantum computing capabilities on the client side. However, in resource-constrained environments, clients typically are resource-limited, e.g., IoT devices. Relaxing this assumption entails a client in the classical domain and a server equipped with quantum resources. The immediate sub-questions that

[2] In split learning, smashed data refers to the intermediate output or activations produced by the last layer of the client-side portion of a split neural network.

arise are: (1) How can we model and train such a hybrid quantum split learning (HQSL) architecture? (2) How do we design a quantum circuit of low enough complexity for the quantum node/layer? (3) How can we model a classical counterpart to benchmark HQSL's classification performance? (4) Would HQSL scale as well as SL with an increasing number of clients?

RQ 2: How do we strengthen such hybrid quantum split learning schemes against data privacy leakage and reconstruction attacks?—addressed in Section 6.

Data privacy leakage from intermediate data transfers between clients and a server is an overarching problem in SL, as highlighted in Section 2. It leads to private input data reconstruction attacks. In this work, we have the following sub-questions: (1) How do we model a threat scenario in which we can investigate HQSL's vulnerability to data privacy leakage and risks of reconstruction attacks? (2) Can we leverage the hybrid structure of HQSL to develop a countermeasure against reconstruction attacks? (3) How does this defense setup perform in the classical setting?

4 Hybrid quantum split learning (HQSL)

This section addresses our first research question, *RQ 1*, as identified in Section 3. We first discuss some preliminaries around gate-based quantum circuits and their components necessary to understand our implementation of HQSL. Then, we describe how we design our HQSL model architecture. We discuss the construction and selection process for the quantum circuit introduced as a quantum layer into the server-side model of HQSL. We also describe how we devise the classical counterpart of the quantum node to allow benchmarking of HQSL's performance. We then propose 2 variants of HQSL depending on the data type they operate on and explain the training algorithm. Lastly, we describe an algorithm to scale HQSL to accommodate multiple clients.

4.1 Gate-based quantum circuits

Before moving into the details of our HQSL model architecture, we first discuss the concept of gate-based quantum circuits, which are the fundamental components of quantum computing. Further background information on quantum computing can be found in Appendix A. Gate-based quantum circuits (or quantum circuits) use quantum gates to manipulate qubits and perform computations during circuit evolution. Encoding classical data into quantum states is a critical initial step, achieved through various encoding methods such as basis, amplitude, or phase encoding. We can describe the encoding process generically as follows: $\phi(\mathbf{X}) = U_e(\mathbf{X})|0\rangle^{\otimes Q}$, where $\mathbf{X}$ is the classical data vector to encode, $|0\rangle^{\otimes Q}$ represents the initial quantum state of Q qubits, typically all initialized to $|0\rangle$, and U_e represents the unitary transformation that encodes the classical vector $\mathbf{X}$ into a quantum state. Another critical part of a quantum circuit is called the *ansatz*, which is a predefined structure of quantum gates. The *ansatz* is parameterized by a set of k free parameters $\Theta = \{\theta_1, \theta_2, \ldots, \theta_k\}$ that are optimized during training, e.g., in QML applications. We can represent the quantum state, $\Psi(\mathbf{X}, \mathbf{\Theta})$, at the end of the *ansatz* as follows: $\Psi(\mathbf{X}, \mathbf{\Theta}) = U_p(\mathbf{\Theta})U_e(\mathbf{X})|0\rangle^{\otimes Q}$, where U_p is the unitary representing the *ansatz*. Quantum measurement, the final step of the circuit, involves collapsing the quantum state into a classical outcome, providing the results of computation. We measure a certain observable, $\hat{\mathbf{A}}$, after each evolution of a quantum circuit and find the expectation value, E, of the measurement as

$$\mathrm{E} = \langle \hat{\mathbf{A}} \rangle = \langle \Psi | \hat{\mathbf{A}} | \Psi \rangle \tag{1}$$

In our work, we use the Pauli-Y observable, i.e., $\hat{\mathbf{A}} = \sigma_y$, where a projective measurement at each qubit would generate ± 1, which are the eigenvalues for the σ_y operator, for the qubit in the $\psi_{y_+} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ i\end{bmatrix}$ and $\psi_{y_-} = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\ -i\end{bmatrix}$ states respectively. The experimental implementation of measurements of a Q-qubit quantum circuit consists of evolving the quantum algorithm M times (or shots) and computing the average for each qubit, $q \in \{1, \ldots, Q\}$, as an approximation of Eq. 1, decomposed for each q as,

$$e_q = \langle \hat{\mathbf{A}} \rangle_q \approx \frac{M_{-1,q}}{M} \cdot (-1) + \frac{M_{+1,q}}{M} \cdot (+1), \tag{2}$$

where $M_{-1,q}$ and $M_{+1,q}$ are the number of times -1 and +1 measurements are obtained for qubit q, respectively. This average serves as an approximation of the probability that a given qubit is in a particular quantum state. For a Q-qubit system, we compute Eq. 2 for each $q \in \{1, \ldots, Q\}$ and obtain a feature vector, $\mathbf{E}$, consisting of classical expectation measurement output at each qubit q as

$$\mathbf{E} = (e_1, e_2, \ldots, e_Q). \tag{3}$$

The three steps described above (encoding of classical data into quantum states, parameterization via an ansatz, and measurement) form the foundation of parameterized (or variational) quantum circuits (PQC or VQC). Recent works have explored PQCs in various advanced configurations and structures, such as Block-Ring topologies for enhanced expressivity and entangling capacity (Liu et al. 2023), and quantum convolutional structures for classification and code recognition tasks (Wu et al. 2024). However, for its practicality in the NISQ generation, this work investigates a small-scale quantum circuit featuring a novel data-loading

technique, illustrated in Fig. 2 and described in detail in the next section.

4.2 Hybrid quantum split learning model

While several SL configurations have been proposed (Vepakomma et al. 2018), this work considers the fundamental vanilla SL setup. In this setup, the labels and smashed data are transmitted to the server-side model at the split layer, while the raw data remain on the client side. During backpropagation, the gradients are transmitted from the server side to the client side across the split layer (see Fig. 1). In HQSL, the quantum layer is introduced as the first layer of the server-side model of SL. The quantum layer consists of a quantum circuit with classical features as input and output. The classical inputs are encoded to quantum states and processed by the quantum circuit. Expectation measurements of the resultant quantum states for each qubit lead to classical features output from the quantum layer (Eq. 3). These are then fed to the next classical layer on the server-side model. The quantum circuit is designed to be of practical importance in the near-term quantum computing era. A general structure of HQSL is shown in Fig. 1. Next, we discuss the construction and selection of the quantum circuit used in this work.

4.2.1 Construction and selection of quantum circuit

To construct the quantum circuit in HQSL's quantum layer, we consider the limitations of current-generation quantum computers (see Appendix A for details). Specifically, the

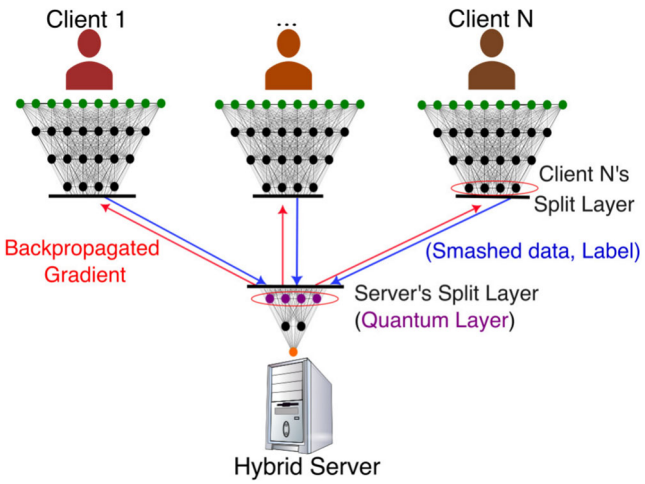


Fig. 1 General structure of a hybrid quantum split learning (HQSL) model with N clients and 1 hybrid quantum server. The clients have a classical model (assumed to be the same for all clients), and the server model has a quantum layer as its first layer (layer with purple units). The black, green, and orange units can be replaced by any classical neural network component (e.g., convolutional layers) to suit specific classification problems

circuit width (number of qubits) and depth (maximum number of gates per qubit) are critical considerations that need to be kept as small as possible. The presence or absence of entangling gates is also another important consideration. We trialed 6 different quantum circuit configurations by adjusting the number of qubits, entangling gates, and encoding type. Each configuration was tested by incorporating the circuit as a quantum layer in our HQSL model. The quantum circuit was selected based on the comparison of accuracy and F1-score between HQSL and its classical analogue. We chose the circuit that consistently outperformed SL in all our experiments. We detail these experiments in Appendix C.

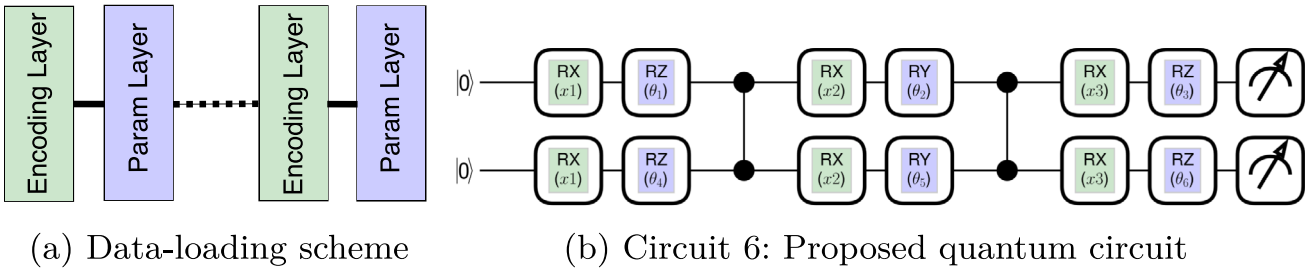
The quantum circuit with the optimal performance is shown as circuit 6 in Fig. 2b. The circuit consists of two qubits, each receiving a 3-dimensional classical input feature vector $\mathbf{X} = (x_1, x_2, x_3)$. These features are encoded into each qubit at three distinct points along the circuit using RX-gates. This encoding strategy is inspired by the data re-uploading technique proposed by Pérez-Salinas et al. (2020), where classical information is introduced multiple times (using U-gates) throughout the circuit layers to enhance representational capacity. However, our designed circuit differs in two fundamental ways: (i) encoding input data $\mathbf{X} \in \mathbb{R}^n$ at n loading points ($n = 3$ for our case) using single-qubit RX-rotations alternating between single-qubit parameterized gates without using U-gates, and (ii) avoiding layer repetitions (re-uploading) to achieve lower circuit depth and fewer parameters while improving performance (see Table 2). We provide more details on these differences and how we designed our circuit next.

The data re-uploading technique involves multiple repetitions of a layer, L_i, consisting of the generic single-qubit rotation gate, $U \in SU(2)$. The U-gate is used in the encoding layer as $U(\mathbf{X})$ and in the parameterized layer as $U(\mathbf{\Theta})$, where $\mathbf{X} = (x_1, x_2, x_3)$ and $\mathbf{\Theta_i} = (\theta_1^i, \theta_2^i, \theta_3^i)$, where $\mathbf{\Theta_i}$ represents the parameters for layer L_i. Thus, each layer can be represented as $L_i = U(\mathbf{X})U(\mathbf{\Theta_i})$. L_i, for $i = 1, \ldots, N$, is repeated, giving rise to the data re-uploading mechanism, and the single-qubit classifier is introduced, given enough repeats, N, are performed.

The U-gate consists of parameters $\mathbf{\Phi} = (\phi_1, \phi_2, \phi_3)$ and can be expressed as $U(\phi_1, \phi_2, \phi_3) \in SU(2)$. We considered the following decomposition of the U-gate:

$$U(\mathbf{\Phi}) = U(\phi_1, \phi_2, \phi_3) = RZ(\phi_3)RY(\phi_2)RZ(\phi_1) \tag{4}$$

For our work, we adapted the above decomposition of this layer, L_i, taking into consideration that the width and depth of a circuit should be kept as small as possible, given the limitations due to current quantum devices. We considered a circuit consisting of only a single layer L consisting of 2 qubits and without any entangling gates. This circuit has a



(a) Data-loading scheme (b) Circuit 6: Proposed quantum circuit

Fig. 2 **a** Our qubit-efficient data-loading scheme consists of alternate layers of encoding and parameterized (Param) gates. **b** The quantum circuit that we utilize in this work has RX gates serving as data-loading points (encoding layer). It is designed to make efficient use of qubits while loading data to the quantum circuit

depth of 6 and consists of a total of 6 trainable parameters $(\theta_1, \ldots, \theta_6)$. We represent this circuit as circuit 7 in Fig. 3.

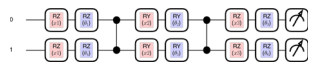
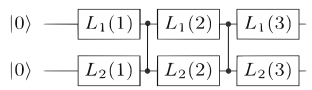
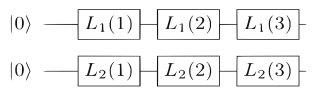
As discussed by Pérez-Salinas et al. (2020), entangling gates can improve the classification performance of the circuit. To introduce entanglement within the circuit consisting of a single layer, L_i, we decomposed the U-gate into RZ–RY–RZ gates and reorganized the circuit such that for each qubit, we uploaded a feature, x_k, for $k \in \{1, 2, 3\}$ using rotational gates, followed immediately by a single-qubit parameterized rotational gate of the same type and a CZ entangling gate between the 2 qubits. This is repeated until all the features have been uploaded. We omit the CZ gate at the end of the circuit. Hence, this circuit still has a total of 6 trainable parameters, but has now a depth of 8. We show this circuit as circuit 8 in Fig. 4.

We built both circuits 7 and 8 and tested their performance when utilized in the quantum layer of HQSL. We compared their performance for the classification of the Fashion-MNIST dataset (Xiao et al. 2017) using the accuracy and F1-score metrics—we experimented only on the Fashion-MNIST dataset because it had more room for improvement compared to the other datasets we studied in this paper. All the details for this dataset and these experiments are provided in Section 5. The results, shown in Table 2, demonstrate that the use of entanglement indeed can improve the accuracy and F1-score during the classification of the Fashion-MNIST dataset at the expense of 2 additional CZ gates, increasing the depth by 2, but keeping the number of trainable parameters to 6.

To develop our proposed circuit (circuit 6), we carried out a final modification to circuit 8. Specifically, we replaced the embedding gates for loading $\mathbf{X} = (x_1, x_2, x_3)$ into the circuit with RX-gates, as shown in Fig. 2b. We note that this modification kept the number of trainable parameters and circuit depth of our proposed circuit similar to circuit 8. This change to RX gates for encoding further improved the classification performance, as shown in Table 2.

The last two rows of Table 2 show the performance when we adopt the standard data re-uploading technique in a 2-qubit circuit with 3 repeats of layer L_i, with (circuit 9) and

Table 2 Table comparing the quantum resource utilization (number of trainable parameters and circuit depth) and 5-fold mean accuracy and F1-score of HQSL models consisting of circuits 7, 8, 6, 9, and 10 during classification of Fashion-MNIST dataset

ID	Circuit	Number of trainable parameters	Circuit depth	Mean accuracy	Mean F1-score
7		6	6	0.812	0.815
8		6	8	0.823	0.825
6		6	8	0.839	0.840
9		18	20	0.824	0.826
10		18	18	0.821	0.823

Circuits 9 and 10 are 3-layer versions of circuit 7 ($L_i(j)$), demonstrating the data re-uploading technique with and without entangling CZ gates, respectively ($L_i(j)$ represents the j^{th} layer on qubit i). Circuit 6, which is a modified version of circuits 7 and 8, achieves $1-2\%$ accuracy and F1-score improvements with fewer parameters and more than half the circuit depth of standard data re-uploading circuits 9 and 10

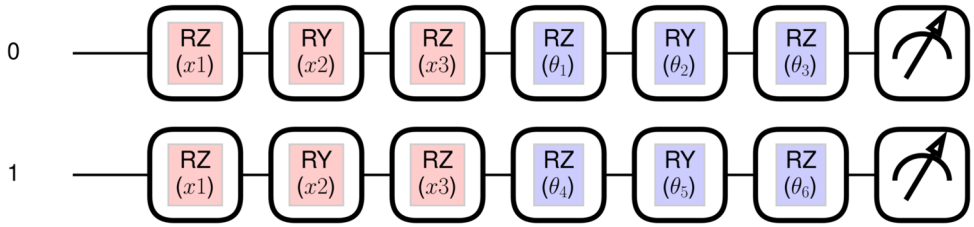


Fig. 3 Circuit 7: Representation of a 2-qubit circuit with a single layer, L, without entanglement consisting of the arrangement $U(\mathbf{X}) - U(\mathbf{\Theta})$, where the U-gate is decomposed here in RZ–RY–RZ gates for each qubit

without entanglement (circuit 10), respectively. We used the same entanglement strategy proposed in Pérez-Salinas et al. (2020), i.e., using CZ-gates after each layer L_i, except the last layer. In Table 2, we also provide a comparison of quantum resource utilization in terms of the number of trainable parameters and circuit depth of our proposed quantum circuit compared to the data re-uploading circuits proposed in Pérez-Salinas et al. (2020).

Table 2 shows that our proposed shallow (8-deep with 6 trainable parameters) qubit-efficient data-loading circuit (circuit 6) provides better accuracy and F1-score than the much deeper circuits (circuits 9 and 10) that use the standard data re-uploading scheme. By loading features exclusively with one type of single-qubit gates (RX-gates in our case) and omitting repeated layers, circuit 6 maintains a shallow depth (8) and minimal trainable parameter count (6), compared to the 3-layer data re-uploading circuits (circuits 9 and 10), which require 18 parameters and depths of 18–20. Our proposed data loading method allows us to achieve better accuracy and F1-score with fewer parameters, improving the trainability and efficiency of our design compared to the standard data re-uploading circuits. *Our proposed circuit 6 is "qubit-efficient," as it provides a method to encode a classical input feature vector*, $\mathbf{X}$, *of arbitrary dimension onto the same qubit. However, this comes at the cost of increasing circuit depth proportionally with input dimensionality.* Hence, in this work, we restrict the circuit depth by only considering 3-dimensional data being loaded onto a 2-qubit quantum circuit. *By imposing this restriction, we also keep our simulations within reasonable runtimes, while still providing high performance during classification with a shallow quantum circuit.*

To demonstrate this, we carried out experiments with circuit 6 by increasing the number of qubits and circuit depth. The resulting accuracy and simulation time taken for 5-fold training can be seen in Fig. 5. In the figure, "shallow" refers to a low-depth circuit with 3 data loading points, while "deep" corresponds to a circuit twice as deep as the shallow one, i.e., 6 data-loading points. Hence, a shallow circuit would require 3-dimensional classical features, while a deep circuit would require a data dimension of 6.

We highlight here that although as we increased the number of qubits and circuit depth, accuracy increased marginally, and the runtime during simulation increased very rapidly (comparing the shallow circuit with 2 qubits and the deep circuit with 8 qubits, an improvement of only around 1.5% causes the time taken for training to increase by more than 500 h). We also highlight that the deeper (maximum number of gates) and wider (number of qubits) a circuit is, the more difficult it is to implement on currently available quantum computers. We also face the risk of the barren plateau phenomenon when the number of trainable parameters is excessively high, resulting in vanishing gradients (McClean et al. 2018). For these reasons, it is desirable to keep our quantum circuit shallow and narrow (3 data-loading points and 2 qubits), while still obtaining high accuracy. In Section 5.1, we effectively provide initial empirical evidence that HQSL with our proposed quantum circuit can be deployed on currently available quantum hardware, providing performance similar to noise-free simulations. *Hence, our empirical findings show that our proposed quantum circuit keeps the training time during simulations within reasonable limits, is more likely to be feasible on near-term devices, and provides a better classification performance than a deep circuit with the data re-uploading technique comprising of 3 repeats of layer* L_i.

This qubit-efficient data-loading methodology for quantum circuit design can be extended beyond the scope of HQSL, presenting a versatile framework with broad applicability. Further research is warranted to comprehensively evaluate its wider implications and potential use cases across various domains, which is beyond the scope of this paper.

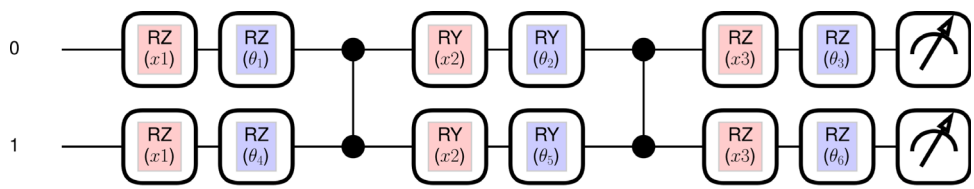


Fig. 4 Circuit 8: Representation of a reorganization of circuit 7 shown in Fig. 3 to introduce CZ entanglement between each qubit

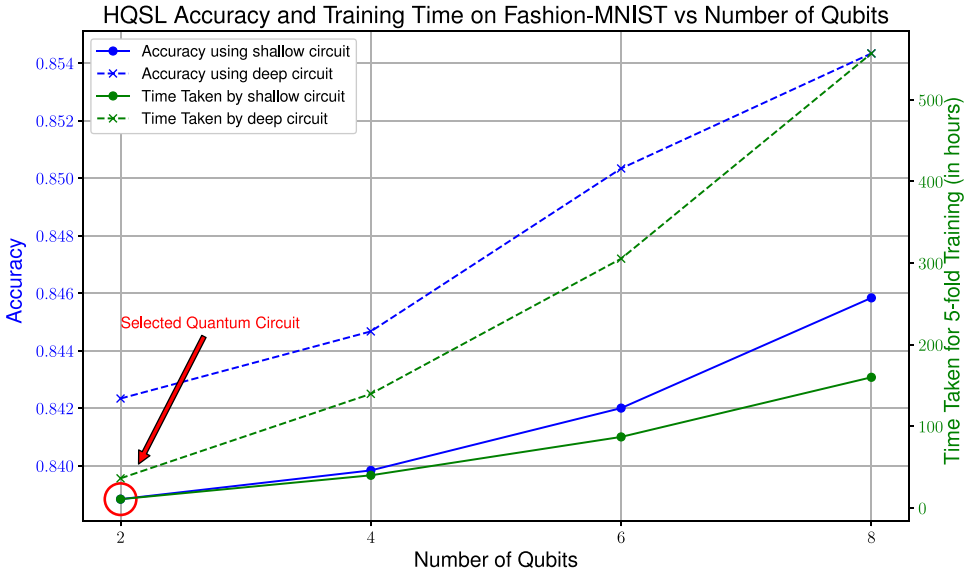


Fig. 5 Effect of the number of qubits and circuit depth on the mean test accuracy and 5-fold training time of HQSL on Fashion-MNIST dataset. Accuracy improves only slightly as we increase the number of qubits and circuit depth at the expense of rapidly increasing training time

Classical benchmarking of HQSL

Our objective is to create an equivalent classical model to benchmark the performance of HQSL. To achieve this, we construct the classical counterpart of our proposed quantum circuit (circuit 6) as follows: It consists of a dense layer with the same number of output neurons as the number of qubits (or measurement outputs) of the quantum layer—two output neurons with `ReLU` activation units (the best-performing activation unit based on our experiments). The number of input neurons in this layer corresponds to the dimension of the classical input (3 input features). The aim is to compare our compact quantum layer to a simple classical dense layer, enabling a fair comparison, taking into consideration the size limitations of quantum circuits. In Appendix C, we provide a comparative analysis of the performances of different quantum circuits and their corresponding classical counterparts when introduced as a layer in the split learning models. This experimental analysis allowed us to construct and select the quantum circuit to be used in HQSL, as we described earlier.

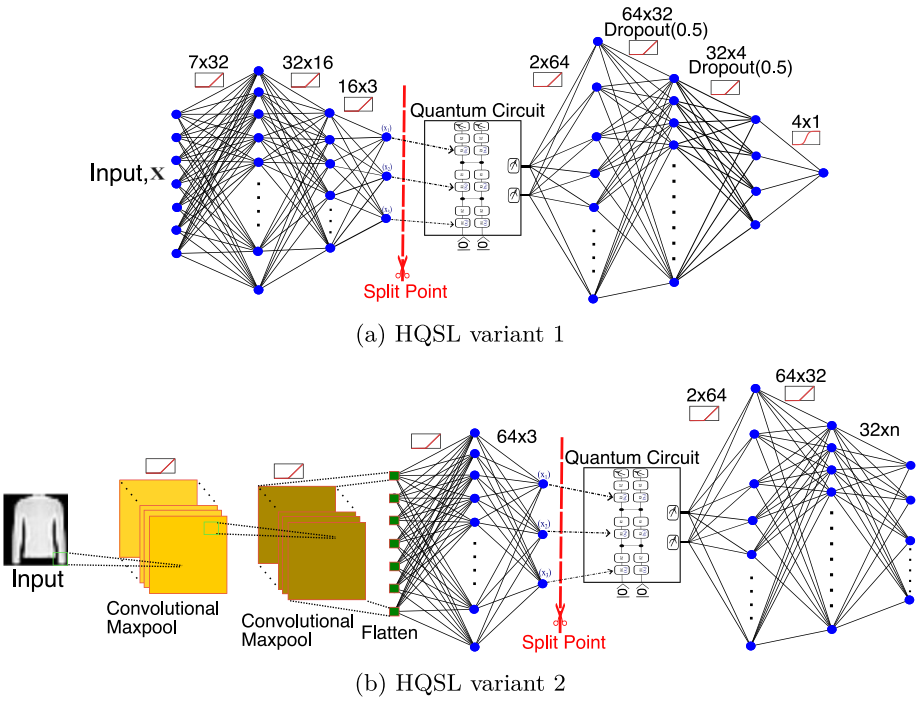
4.2.2 HQSL model variants

We propose two variants of HQSL designed for multivariate binary and multi-class single-channel image dataset classification. Our approach involves constructing an HQNN and subsequently dividing it at the *split point* (see Fig. 6) so that the initial part of the HQNN represents the client side (classical neural network layers) and the remaining segment functions as the server side (hybrid quantum neural network layers). This explicit division or splitting of the HQNN into client-side classical layers and server-side hybrid quantum-classical layers constitutes the key novelty of HQSL, distinguishing it from existing HQNN approaches that do not address resource-constrained deployment or collaborative training frameworks.

We assume here that the clients, e.g., IoT devices, have sufficient classical resources to perform client-side computations and lack quantum resources, as stated in Section 3. This is a fair assumption because typical IoT devices have sufficient processing capabilities to execute lightweight classical neural network models, but generally lack the resources and infrastructure for quantum computations. The client-side model portion extracts compressed feature representations (smashed data) to transmit low-dimensional data to the server-side model portion, reducing communication overhead and enabling a shallow quantum circuit in the quantum layer for further processing on the server side.

HQSL variant 1 The client-side model consists of 4 fully connected dense layers with `ReLU` activation functions with a 7-dimensional input and outputs smashed data of dimension 3. For the server-side model, we start with the quantum layer, constructed using our proposed quantum circuit in Section 4.2.1, which has an input dimension of 3 and an output dimension of 2. We concatenate 4 additional fully connected classical dense layers with the `ReLU` activation function (except the last layer, which uses `Sigmoid`) with an output dimension of 1 (for binary classification). We apply `Dropout` layers on the first two classical layers on the server-side model to improve regularization. For the classical counterpart of this variant, we replace the quantum layer with the classical dense layer as described earlier.

HQSL variant 2 The client-side model consists of two convolutional neural network filters, each with `ReLU` activation function and a `MaxPool2d` layer, with a stride of 2 and



(a) HQSL variant 1

(b) HQSL variant 2

Fig. 6 HQSL variants: The split point splits the network into the client-side portion (left) and server-side portion (right). **a** HQSL variant 1: For binary classification on multivariate datasets. **b** HQSL variant 2: For n-class classification on single-channel image datasets. We can set the number of output nodes, n, depending on the number of classes in the dataset

kernel size of 2, followed by a `flatten` layer, and 2 fully connected layers to bring the feature dimensions down to 3. This is the client-side portion of the model. For the server side, we introduce our quantum circuit as a quantum layer and concatenate it with 3 more fully connected classical dense layers with an output dimension of n (for n-class classification). We apply the `ReLU` activation function on the first 2 classical layers on the server side. For the classical version, we replace the quantum layer with its equivalent classical dense layer, acting as a benchmark for this HQSL variant. Figure 6 depicts the two HQSL model variants that we study in this work.

4.2.3 Hybrid quantum split learning model training

In our HQSL framework, the training process follows a client–server paradigm where data is processed in two distinct stages. The client side, which consists of a classical neural network, processes the raw input data $\mathbf{X}$ and extracts feature representations. These extracted features, referred to as the smashed data $\mathbf{Z}$, are then transmitted to the server side, together with the true labels, $\mathbf{Y}$.

The server side consists of a hybrid quantum-classical sub-model, where, first, the quantum layer processes, $\mathbf{Z}$, and outputs measurement expectation values $\mathbf{E}$. $\mathbf{E}$ is now input to the remaining layers of the server-side model portion which, in our HQSL model, are all classical dense layers with $\hat{\mathbf{Y}}$ as the final output. This structured split ensures that the client operates solely with classical resources, while the server-side quantum component allows the client to benefit from quantum processing without requiring direct access to quantum hardware. The placement of the quantum layer as the first layer of the server-side model portion is a deliberate design choice aimed at enhancing resistance to reconstruction attacks in split learning, as further discussed in Section 6. Next, we outline the computation and backpropagation of gradients through the classical and quantum layers of HQSL to enable effective training.

For the classical layers of HQSL, gradients are computed from the loss function, $\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}})$, using the classical backpropagation algorithm via the chain rule. However, it becomes non-trivial to do the same for the quantum layer. Schuld et al. devised the parameter-shift rule, which permits backpropagation through the quantum layer (Schuld et al. 2019). We briefly discuss the main ideas in the following:

Consider a quantum circuit, $f_q : \mathbf{Z} \mapsto \mathbf{E}$ ($f_q : \mathbb{R}^M \rightarrow \mathbb{R}^N$), where f_q is parameterized by a set of k free parameters, $\mathbf{\Theta} = \{\theta_1, \theta_2, \ldots, \theta_k\}$, and maps input, $\mathbf{Z} = (z_1, z_2, \ldots, z_M)$ to the measurement outputs, $\mathbf{E} = (e_1, e_2, \ldots, e_N)$. In HQSL, the quantum layer can be considered as being sandwiched between two sets of classical layers: the client-side model layers, $f_c : \mathbf{X} \mapsto \mathbf{Z}$ ($f_c : \mathbb{R}^{in} \rightarrow \mathbb{R}^M$), and the server-side classical layers, $f_s : \mathbf{E} \mapsto \hat{\mathbf{Y}}$ ($f_s : \mathbb{R}^N \rightarrow \mathbb{R}^{out}$), where $\hat{\mathbf{Y}}$ is the output of HQSL. The smashed data $\mathbf{Z}$ is then transferred to the server side to continue forward propagation, together with the true labels $\mathbf{Y}$, to compute the loss $\mathcal{L}$.

The derivatives of the expectation value of the measurement, $\mathbf{E}$, of the quantum circuit, f_q, with respect to the gate parameters, $\mathbf{\Theta}$, are computed by evolving the circuit twice per parameter, with a ($\pm$) shift in that parameter. Representing the

partial derivative of $f_q(\boldsymbol{\Theta}, \mathbf{Z}) = \mathbf{E}$ with respect to parameter θ_i for $i \in \{1, 2, \ldots, k\}$ as $\frac{\delta f_q}{\delta \theta_i}$, we can express the parameter-shift rule as

$$\frac{\delta f_q}{\delta \theta_i} = r[f_q(\theta_i + \frac{\pi}{4r}, \mathbf{Z}) + f_q(\theta_i - \frac{\pi}{4r}, \mathbf{Z})] = \frac{\delta \mathbf{E}}{\delta \theta_i}, \quad (5)$$

where r is a shift constant. This gives us the exact gradient with respect to each free parameter θ_i in the quantum circuit. Similarly, we can compute the derivative with respect to input z_j, for $j \in \{1, 2, \ldots, M\}$, to the quantum circuit:

$$\frac{\delta f_q}{\delta z_j} = r[f_q(\boldsymbol{\Theta}, z_j + \frac{\pi}{4r}) + f_q(\boldsymbol{\Theta}, z_j - \frac{\pi}{4r})] = \frac{\delta \mathbf{E}}{\delta z_j} \quad (6)$$

In summary, on the server side, for the quantum layer, during forward propagation, the smashed data $\mathbf{Z}$ are fed as classical features to the quantum layer, and the quantum circuit is evolved for a default of 1000 shots. At the end of the quantum circuit evolutions, an expectation value for the measurement is calculated as per Eq. 2. These calculated values are then used as classical features for the next classical layer. During backpropagation, to compute the derivatives with respect to parameter $\theta_i \in \{\theta_1, \theta_2, \ldots, \theta_k\}$ and inputs $z_i \in \{z_1, z_2, \ldots, z_M\}$, two evaluations of the circuit are computed with a $\pm$ shift in the parameter θ_i. These two computations are then added according to Eqs. 5 and 6. This is repeated until the derivatives with respect to all θ_i and z_j are computed. Using the computed derivatives, we update the gate parameters, $\boldsymbol{\Theta}$, accordingly. The derivative $\frac{\delta f_q}{\delta z_j}$ is used to compute the gradient with respect to the quantum layer input z_j as $\frac{\delta \mathcal{L}}{\delta z_j}$ using the chain rule. This gradient is transmitted across the split layer to the client-side model to continue the backpropagation of the loss function, $\mathcal{L}$, to the client-side classical layers. We summarize the HQSL training in Algorithm 1. In the algorithm, $\frac{\delta(\cdot)}{\delta(\cdot)}$ represents the Jacobian matrix that stores the element-wise gradients of the vector in the numerator with respect to the vector in the denominator.

4.3 Hybrid quantum split learning with multiple clients

Scaling HQSL to accommodate a larger number of clients enables multiple clients to leverage the quantum resources at the central server. The method we use to scale HQSL with K clients follows the basic round-robin protocol and is as follows: We initialize the model parameters of the client side and server side as $\mathbf{W^C}$ and $(\boldsymbol{\Theta}, \mathbf{W}^S)$, respectively. $\boldsymbol{\Theta}$ corresponds to the quantum layer trainable gate parameters that lie on the server side. We randomly choose a client k ($\mathsf{k} \in \{0, 1, 2, \ldots, \mathsf{K}-1\}$) and train it in collaboration with the server. This consists of one round of forward and backward propagation and makes 1 local epoch for client k. The updated client k model weights $\mathbf{W}^{C'}$ are then sent to the next client (k′) that updates its model weights before another round of forward and backward propagation with the server. This

Algorithm 1 Hybrid quantum split learning training with a single classical client and hybrid quantum server.

Input: Number of Training epochs N, learning rate η
Output: Optimal weights, $\tilde{\mathbf{W}} = (\tilde{\mathbf{W}}^C, \tilde{\mathbf{W}}^S)$, for classical layers and optimal parameters, $\tilde{\boldsymbol{\Theta}}$, for quantum layer
Notations: Client's inputs: $\mathbf{X}$, client's labels: $\mathbf{Y}$, predicted labels: $\hat{\mathbf{Y}}$, client's smashed data: $\mathbf{Z}$, quantum layer expectation measurement output: $\mathbf{E}$, loss function: $\mathcal{L}$
Initialization: Initialize client-side layers (f_c) weights, $\mathbf{W}^C$; server-side quantum layer (f_q) parameters, $\boldsymbol{\Theta}$, and classical layers (f_s) weights, $\mathbf{W}^S$.

START OF TRAINING
for $n = 1, \ldots, N$ *Training Epochs* **do**
Client side: `// Forward Pass`
$\mathbf{Z} = f_c(\mathbf{X}, \mathbf{W}_n^C)$; Send $\mathbf{Z}$ and $\mathbf{Y}$ to the server
Server side: `// Forward Pass, Compute Loss, Backward Pass`
Quantum Layer: $\mathbf{E} = f_q(\mathbf{Z}, \boldsymbol{\Theta}_n)$ `// Compute` $\mathbf{E}$ `as per Eq. 3`
Classical Layers: $\hat{\mathbf{Y}} = f_s(\mathbf{E}, \mathbf{W}_n^S)$
Compute Loss, $\mathcal{L} = \mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}})$, and gradient w.r.t. output $\hat{\mathbf{Y}}$ as $\frac{\delta \mathcal{L}}{\delta \hat{\mathbf{Y}}}$
Compute $\frac{\delta \hat{\mathbf{Y}}}{\delta \mathbf{W}_n^S} = \frac{\delta f_s(\mathbf{E}, \mathbf{W}_n^S)}{\delta \mathbf{W}_n^S}$ and $\frac{\delta \hat{\mathbf{Y}}}{\delta \mathbf{E}} = \frac{\delta f_s(\mathbf{E}, \mathbf{W}_n^S)}{\delta \mathbf{E}}$
Compute gradient w.r.t. $\mathbf{W}_n^S$ as $\frac{\delta \mathcal{L}}{\delta \mathbf{W}_n^S} = \frac{\delta \mathcal{L}}{\delta \hat{\mathbf{Y}}} \cdot \frac{\delta \hat{\mathbf{Y}}}{\delta \mathbf{W}_n^S}$
`// Chain rule`
Compute gradient w.r.t. $\mathbf{E}$ as: $\frac{\delta \mathcal{L}}{\delta \mathbf{E}} = \frac{\delta \mathcal{L}}{\delta \hat{\mathbf{Y}}} \cdot \frac{\delta \hat{\mathbf{Y}}}{\delta \mathbf{E}}$
`// Chain rule`
Update server classical layers weights $\mathbf{W}^S$ as: $\mathbf{W}_{n+1}^S \leftarrow \mathbf{W}_n^S - \eta \frac{\delta \mathcal{L}}{\delta \mathbf{W}_n^S}$
Quantum Layer:
Compute $\frac{\delta \mathbf{E}}{\delta \boldsymbol{\Theta}_n} = \frac{\delta f_q(\mathbf{Z}, \boldsymbol{\Theta}_n)}{\delta \boldsymbol{\Theta}_n}$ and $\frac{\delta \mathbf{E}}{\delta \mathbf{Z}} = \frac{\delta f_q(\mathbf{Z}, \boldsymbol{\Theta}_n)}{\delta \mathbf{Z}}$
`// Parameter-shift rule Eqs. 5, 6`
Compute gradient w.r.t. $\boldsymbol{\Theta}_n$ as $\frac{\delta \mathcal{L}}{\delta \boldsymbol{\Theta}_n} = \frac{\delta \mathcal{L}}{\delta \mathbf{E}} \cdot \frac{\delta \mathbf{E}}{\delta \boldsymbol{\Theta}_n}$
`// Chain rule`
Compute gradient w.r.t. $\mathbf{Z}$ as $\frac{\delta \mathcal{L}}{\delta \mathbf{Z}} = \frac{\delta \mathcal{L}}{\delta \mathbf{E}} \cdot \frac{\delta \mathbf{E}}{\delta \mathbf{Z}}$
`// Chain rule`
Update Quantum Layer Parameters $\boldsymbol{\Theta}$ as: $\boldsymbol{\Theta}_{n+1} \leftarrow \boldsymbol{\Theta}_n - \eta \frac{\delta \mathcal{L}}{\delta \boldsymbol{\Theta}_n}$
Send $\frac{\delta \mathcal{L}}{\delta \mathbf{Z}}$ to client side to continue backward pass
Client side: `// Backward Pass`
Compute $\frac{\delta \mathbf{Z}}{\delta \mathbf{W}_n^C} = \frac{\delta f_c(\mathbf{X}, \mathbf{W}_n^C)}{\delta \mathbf{W}_n^C}$
Compute gradient w.r.t. $\mathbf{W}_n^C$ as $\frac{\delta \mathcal{L}}{\delta \mathbf{W}_n^C} = \frac{\delta \mathcal{L}}{\delta \mathbf{Z}} \cdot \frac{\delta \mathbf{Z}}{\delta \mathbf{W}_n^C}$
`// Chain rule`
Update client model weights $\mathbf{W}^C$ as: $\mathbf{W}_{n+1}^C \leftarrow \mathbf{W}_n^C - \eta \frac{\delta \mathcal{L}}{\delta \mathbf{W}_n^C}$
End of Epoch n
END OF TRAINING
return $\tilde{\mathbf{W}} = (\tilde{\mathbf{W}}^C, \tilde{\mathbf{W}}^S)$, $\tilde{\boldsymbol{\Theta}}$

marks the end of client k''s local epoch. After all K clients have been served in that 1 global epoch, we move to the next global epoch and resume training client k. It is important to highlight that during each communication round between a client and the server, only low-dimensional data is transmitted, as the output layer of the client-side model consists of only 3 nodes, as shown in Fig. 6. This significantly reduces the communication overhead, especially when considering a very large number of clients, e.g., IoT devices in edge networks. A detailed analysis or optimization of communication efficiency is beyond the scope of this work. We leave a thorough investigation of strategies to further minimize communication overhead in HQSL for future research.

The differences due to the presence of the quantum layer in HQSL are as follows: A forward and backward propagation round consists of encoding classical smashed data within the quantum layer. At the measurement stage, we use a default of 1000 shots to sample the expectation value of the measurements due to the probabilistic nature of quantum circuits. Finally, we use the parameter-shift rule described earlier for computing gradients with respect to the parameters ($\boldsymbol{\Theta}$) and inputs ($\mathbf{Z}_\mathsf{k}$) of the quantum circuit. We summarize the method for including multiple clients in Algorithm 2 (presented in two parts: initialization and training phase).

Algorithm 2: Hybrid Quantum Split Learning Training with K Classical Clients and Hybrid Quantum Server (Initialization Phase)

Input: Number of clients K, number of training epochs N, learning rate η
Output: Optimal weights, $\tilde{\mathbf{W}} = (\tilde{\mathbf{W}}^C, \tilde{\mathbf{W}}^S)$, for classical layers and optimal parameters, $\tilde{\boldsymbol{\Theta}}$, for quantum layer
Notations: Client k's inputs: $\mathbf{X}_\mathsf{k}$, labels: $\mathbf{Y}_\mathsf{k}$, predicted labels: $\hat{\mathbf{Y}}_\mathsf{k}$, smashed data: $\mathbf{Z}_\mathsf{k}$, quantum layer outputs when client k is being served: $\mathbf{E}_\mathsf{k}$, loss function: $\mathcal{L}$
Initialization: Initialize client-side layers (f_c) weights, $\mathbf{W}^C$; server-side quantum layer (f_q) parameters, $\boldsymbol{\Theta}$, and classical layers (f_s) weights, $\mathbf{W}^S$.

5 Experiments and results

In this section, we empirically assess the feasibility and scalability of HQSL by comparing its performance in classification tasks to that of their corresponding SL models. We first outline our experiments and then present our results and discuss our findings. All our programs were written using Python 3.11.3 and PyTorch 2.1.0 libraries. We simulate the quantum part of our HQSL models using the Pennylane library with its PyTorch backend. The experiments were conducted on an NVIDIA GeForce RTX 2080 Ti GPU machine system.

We use five publicly available datasets in this work to test and validate our HQSL architectures. We summarize these datasets in Table 3. Detailed descriptions, including dataset processing methods, can be found in Appendix B. Setting a fixed seed, we split our datasets into five non-overlapping folds in preparation for five-fold cross-validation for our

Algorithm 2 Hybrid Quantum Split Learning Training with K Classical Clients and Hybrid Quantum Server (Training Phase)

START OF TRAINING
for $n = 1, \ldots, N$ *Training Epochs* **do**
  Start of global epoch n
  for *each client* k, $\mathsf{k} \in \{0, \ldots, \mathsf{K}-1\}$ **do**
    Set next client k' as $\mathsf{k}' = (\mathsf{k}+1) \pmod{\mathsf{K}}$
    Start of client k's local epoch
    Client k side: `// Forward Pass`
      $\mathbf{Z}_\mathsf{k} = f_c(\mathbf{X}_\mathsf{k}, \mathbf{W}^C_{n,\mathsf{k}})$; Send $\mathbf{Z}_\mathsf{k}$ and $\mathbf{Y}_\mathsf{k}$ to server
    .....
    Server side: `// Forward Pass, Compute Loss, Backward Pass`
      Quantum Layer: $\mathbf{E}_\mathsf{k} = f_q(\mathbf{Z}_\mathsf{k}, \boldsymbol{\Theta}_{n,\mathsf{k}})$ `// Compute` $\mathbf{E}_\mathsf{k}$ `as per Eq. 3`
      Classical Layers: $\hat{\mathbf{Y}}_\mathsf{k} = f_s(\mathbf{E}_\mathsf{k}, \mathbf{W}^S_{n,\mathsf{k}})$
        Compute Loss, $\mathcal{L}_\mathsf{k} = \mathcal{L}(\mathbf{Y}_\mathsf{k}, \hat{\mathbf{Y}}_\mathsf{k})$, and gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \hat{\mathbf{Y}}_\mathsf{k}}$
        Compute $\frac{\delta \hat{\mathbf{Y}}_\mathsf{k}}{\delta \mathbf{W}^S_{n,\mathsf{k}}} = \frac{\delta f_s(\mathbf{E}_\mathsf{k}, \mathbf{W}^S_{n,\mathsf{k}})}{\delta \mathbf{W}^S_{n,\mathsf{k}}}$ and $\frac{\delta \hat{\mathbf{Y}}_\mathsf{k}}{\delta \mathbf{E}_\mathsf{k}} = \frac{\delta f_s(\mathbf{E}_\mathsf{k}, \mathbf{W}^S_{n,\mathsf{k}})}{\delta \mathbf{E}_\mathsf{k}}$
        Compute gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{W}^S_{n,\mathsf{k}}} = \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \hat{\mathbf{Y}}_\mathsf{k}} \cdot \frac{\delta \hat{\mathbf{Y}}_\mathsf{k}}{\delta \mathbf{W}^S_{n,\mathsf{k}}}$ `// Chain rule`
        Compute gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{E}_\mathsf{k}} = \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \hat{\mathbf{Y}}_\mathsf{k}} \cdot \frac{\delta \hat{\mathbf{Y}}_\mathsf{k}}{\delta \mathbf{E}_\mathsf{k}}$ `// Chain rule`
        Update $\mathbf{W}^S$ as: $\mathbf{W}^S_{n,\mathsf{k}'} \leftarrow \mathbf{W}^S_{n,\mathsf{k}} - \eta \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{W}^S_{n,\mathsf{k}}}$
      Quantum Layer:
        Compute $\frac{\delta \mathbf{E}_\mathsf{k}}{\delta \boldsymbol{\Theta}_{n,\mathsf{k}}} = \frac{\delta f_q(\mathbf{Z}_\mathsf{k}, \boldsymbol{\Theta}_{n,\mathsf{k}})}{\delta \boldsymbol{\Theta}_{n,\mathsf{k}}}$ and $\frac{\delta \mathbf{E}_\mathsf{k}}{\delta \mathbf{Z}_\mathsf{k}} = \frac{\delta f_q(\mathbf{Z}_\mathsf{k}, \boldsymbol{\Theta}_{n,\mathsf{k}})}{\delta \mathbf{Z}_\mathsf{k}}$
        `// Parameter-shift rule Eqs. 5, 6`
        Compute gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \boldsymbol{\Theta}_{n,\mathsf{k}}} = \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{E}_\mathsf{k}} \cdot \frac{\delta \mathbf{E}_\mathsf{k}}{\delta \boldsymbol{\Theta}_{n,\mathsf{k}}}$ `// Chain rule`
        Compute gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{Z}_\mathsf{k}} = \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{E}_\mathsf{k}} \cdot \frac{\delta \mathbf{E}_\mathsf{k}}{\delta \mathbf{Z}_\mathsf{k}}$ `// Chain rule`
        Update $\boldsymbol{\Theta}$ as: $\boldsymbol{\Theta}_{n,\mathsf{k}'} \leftarrow \boldsymbol{\Theta}_{n,\mathsf{k}} - \eta \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \boldsymbol{\Theta}_{n,\mathsf{k}}}$
        Send $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{Z}_\mathsf{k}}$ to client k to continue backward pass
    .....
    Client k side: `// Backward Pass`
      Compute $\frac{\delta \mathbf{Z}_\mathsf{k}}{\delta \mathbf{W}^C_{n,\mathsf{k}}} = \frac{\delta f_c(\mathbf{X}_\mathsf{k}, \mathbf{W}^C_{n,\mathsf{k}})}{\delta \mathbf{W}^C_{n,\mathsf{k}}}$
      Compute gradient $\frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{W}^C_{n,\mathsf{k}}} = \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{Z}_\mathsf{k}} \cdot \frac{\delta \mathbf{Z}_\mathsf{k}}{\delta \mathbf{W}^C_{n,\mathsf{k}}}$ `// Chain rule`
      Update client k model weights $\mathbf{W}^C$ as: $\mathbf{W}^C_{n,\mathsf{k}'} \leftarrow \mathbf{W}^C_{n,\mathsf{k}} - \eta \frac{\delta \mathcal{L}_\mathsf{k}}{\delta \mathbf{W}^C_{n,\mathsf{k}}}$
    End of client k's local epoch
    Send $\mathbf{W}^{C'} = \mathbf{W}^C_{n,\mathsf{k}'}$ to client k', and start local epoch for client k'
  End of global epoch n
  Start global epoch $n+1$ with client k with updated weights, $\mathbf{W}^C_{n+1,\mathsf{k}'}$, $\mathbf{W}^S_{n+1,\mathsf{k}'}$, and $\boldsymbol{\Theta}_{n+1,\mathsf{k}'}$
END OF TRAINING
return $\tilde{\mathbf{W}} = (\tilde{\mathbf{W}}^C, \tilde{\mathbf{W}}^S)$, $\tilde{\boldsymbol{\Theta}}$

Table 3 Datasets used for our experiments

Dataset	Training samples	Testing samples	# features/ Image size	# classes
Botnet DGA (Suryotrisongko 2022b)	800	200	7	2
Breast Cancer (Zwitter and Soklic 1988)	455	114	7	2
MNIST (Deng 2012)	4800	1200	28 x 28	10
Fashion-MNIST (Xiao et al. 2017)	4800	1200	28 x 28	10
Speech Commands (Warden 2018)	6388	1597	28 x 28	2

experiments. Specifically, we use four of the folds for collectively training our model and the remaining fold to test it. For the multi-class image datasets (MNIST and Fashion-MNIST), we split the datasets in a stratified manner to ensure an even and balanced distribution of classes.

5.1 Hybrid quantum split learning versus classical split learning: single client experiments

In this section, we describe our experiments to determine HQSL's feasibility and performance by considering a single client-single server case. We then discuss the results of our experiment.

First, we constructed the quantum circuit shown in Fig. 2b using PennyLane's `default-qubit` device and converted it to a quantum layer. To build HQSL variants 1 and 2 as introduced in Section 4.2.2, we first constructed their respective client-side and server-side model portions separately, as illustrated in Fig. 6, ensuring that each variant adhered to the split learning framework. We emphasize again here that the quantum layer was incorporated as the initial layer of the server-side model portion of each HQSL variant. This separation allows for the collaborative training of the client-side and server-side model portions in HQSL while maintaining the distinct roles of the client and the server as per the split learning protocol.

We built HQSL variant 1 for training on Botnet DGA and Breast Cancer datasets and HQSL variant 2 on MNIST, Fashion-MNIST, and Speech Command spectrogram datasets. In HQSL variant 2, we set the number of output nodes to $n = 10$ (10 classes) for the MNIST and Fashion-MNIST datasets and $n = 2$ (2 classes) for the Speech Commands spectrograms dataset. We paired each of these 5 experiments with their classical counterparts to benchmark the performance of our HQSL model variants.

For our hyperparameters, we utilized the `Adam` optimizer with a learning rate of 10^{-3} across all experiments, except for the Speech Commands dataset, where a learning rate of 10^{-4} provided better convergence. We used the binary cross entropy loss function for classifying the 2 multi-variate datasets and the cross entropy loss function for the 3 image classification tasks. We trained our HQSL model variants as per Algorithm 1 for 100 epochs for Botnet DGA and Breast Cancer datasets and 50 epochs for MNIST, FMNIST, and Speech Commands spectrograms datasets. We report the mean and standard deviation of the accuracies and F1-scores over five folds of the datasets in Fig. 7. Across all pairs of experiments, we set the random seed to be a constant and fixed the training and testing datasets for each of the 5 folds. These experiments were also performed on the centralized (unsplit) versions of each split learning model. We highlight here that the training process of the centralized version of HQSL is identical to that of a single-node HQNN.

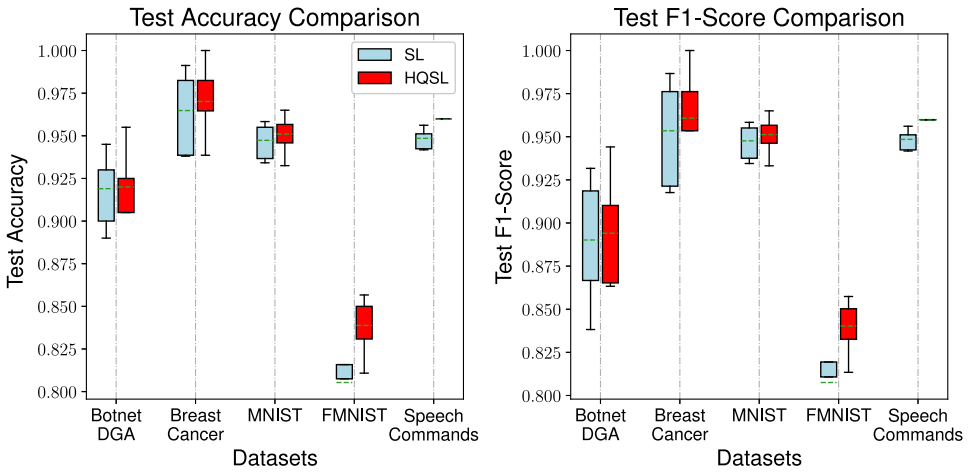


Fig. 7 Performance comparison in terms of test accuracy and F1-score of SL (blue) versus HQSL (red) with 5-fold cross-validation. The dashed green lines spanning the width of each box represent the mean test results. In every case, HQSL outperforms or is nearly on par with SL. On the FMNIST and Speech Commands datasets, a mean improvement of approximately 3% and 1.5%, respectively, due to HQSL is observed

We next compare the classification performance results of single-client HQSL against their classical SL equivalent, as discussed in the next section.

Hardware and noisy simulator experiments To evaluate the feasibility of deploying HQSL on real or noisy quantum hardware, we tested the model on both real IBMQ devices and simulated noisy environments using `qiskit_aer` (Javadi-Abhari et al. 2024) with increasing single and two-qubit depolarizing noise levels ($p = 0.05$ to 0.09). We reference calibrated error rates from the 127-qubit `ibm_brisbane` backend, where median single-qubit gate errors (SX) are typically below $3e^{-4}$, and two-qubit gate (ECR) errors have a median of $7e^{-3}$, which are substantially below the noise levels used in our noisy simulation experiments. These simulations allow us to test the limits of HQSL's robustness under exaggerated noise conditions that exceed those observed on current hardware. We benchmarked against Pennylane's noise-free `default.qubit` device. Due to the high execution time associated with accessing real quantum hardware, we limited our evaluation to a single fold of the Botnet DGA dataset. Specifically, we trained single-client HQSL using the training set on the `default.qubit` simulator and evaluated its performance on the various backends using the corresponding testing set. These results are presented in Fig. 8.

Results and discussions

We present the results of our tests on HQSL and SL for a single client in Fig. 7. We highlight that the test results for the split and centralized HQSL versions are identical, confirming that the splitting process does not degrade model accuracy and F1-score. This demonstrates that HQSL successfully maintains the benefits of split learning (collaborative training of a resource-constrained client with a quantum-enhanced server) while achieving the same performance as its centralized single-node HQNN counterpart.

By applying the same approach to their equivalent SL model, we conduct a comparative analysis of each model's performance in terms of accuracy and F1-score. Figure 7 illustrates the 5-fold test accuracies and F1-scores obtained across all datasets in box plot format. The mean values are represented by the dashed green line within each box, while

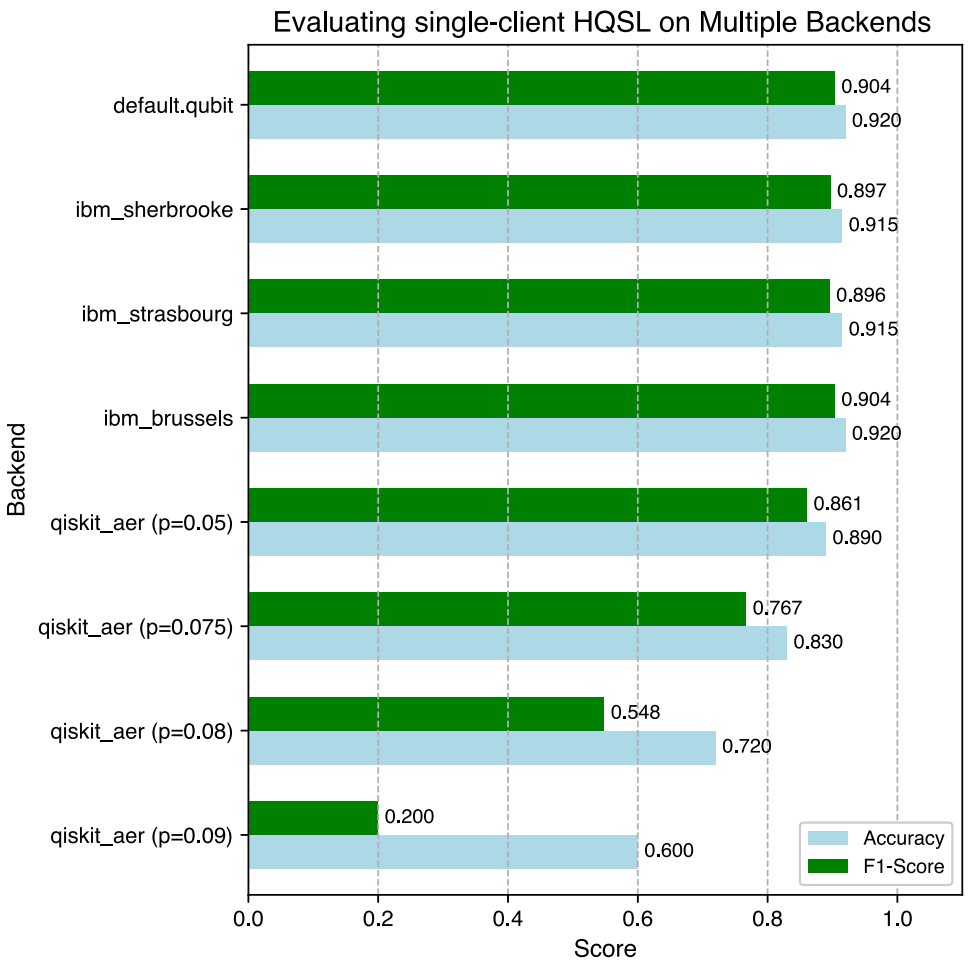


Fig. 8 Deployment accuracy and F1-score of HQSL evaluated on the Botnet DGA dataset across multiple backends. `default.qubit` represents HQSL's noise-free simulation results, which match the performance obtained on IBMQ hardware (`ibm_sherbrooke`, `ibm_strasbourg`, `ibm_brussels`). Noisy simulators (`qiskit_aer`) with increasing depolarizing noise p demonstrate a drop in performance as noise levels increase. This shows HQSL's noise-resilience during deployment on currently available hardware

the heights of the boxes indicate the standard deviations in the testing results.

Although marginal, improvements in both mean accuracies and F1-scores of HQSL were obtained on the Botnet DGA (mean accuracy improvement: ↑ 0.1%, mean F1-score improvement: ↑ 0.4%), Breast Cancer (↑ 0.6%, ↑ 0.7%), and MNIST (↑ 0.4%, ↑ 0.4%) datasets. Significant outperformance in terms of mean accuracy and F1-score were obtained on the Fashion-MNIST (↑ 3%, ↑ 4%) and Speech Commands (↑ 1.5%, ↑ 1.5%) datasets. The visually narrow box plot for the Speech Commands dataset in the HQSL case is a result of the low variance across the five cross-validation folds. This indicates the higher stability of our HQSL model's performance on this dataset compared to the other datasets and to SL's performance of the Speech Commands dataset. For completeness, we provide the performance results for the Speech Commands dataset in Appendix F.

For all of the five datasets used in this study, HQSL proved feasible as it at least matched the testing performance of SL in terms of accuracies and F1-scores. As demonstrated by the results on the Fashion-MNIST and Speech Commands spectrograms datasets, HQSL has the potential to achieve higher accuracy and F1-score compared to its classical counterpart. We highlight that with just the introduction of 1 quantum layer consisting of a small 2-qubit quantum circuit, HQSL can slightly outperform classical split learning, showcasing the potential power of quantum computation beyond the NISQ era.

It is noteworthy that simulating a quantum computer using a classical computer is computationally intensive, despite having only 1 quantum layer consisting of a simple quantum circuit on the server side of HQSL. The training time for HQSL was significantly longer than that of SL, depending on the size of the dataset. For instance, 5-fold training for 100 epochs, each with 569 data points from the Breast Cancer dataset split in the train:test ratio of 4:1 took approximately 1 h to run on HQSL. On the other hand, 5-fold training with 6000 image-label pairs from the MNIST dataset took around 9 h to complete 50 training epochs of HQSL. In contrast, their classical analogues only took a few minutes to train, which is significantly faster than HQSL.

Evaluation results on real quantum devices and noisy simulators To assess the practical viability of HQSL under real-world noise, we evaluated the model on both real IBMQ backends and noisy simulators. As shown in Fig. 8, HQSL achieves consistently high accuracy and F1-scores on `ibm_sherbrooke`, `ibm_strasbourg`, and `ibm_brussels`, similar to the noise-free `default.qubit` simulator, indicating that currently available noisy quantum hardware is sufficient to support HQSL deployment with our proposed qubit-efficient data-loading quantum circuit in its quantum layer. Interestingly, significant performance drops occurred only when using unrealistically high depolarizing noise models in simulation (e.g., $p = 0.08$ or $p = 0.09$), suggesting that these levels represent overly pessimistic device conditions. These results further support the feasibility of deploying HQSL on real hardware.

In the single client case, HQSL with a quantum layer consisting of only a small quantum circuit is feasible and offers tangible improvements in classification accuracy and F1-score compared to SL. Moreover, evaluations on real quantum hardware and noisy simulators highlight HQSL's practical feasibility and robustness for deployment on currently available quantum devices.

5.2 Hybrid quantum split learning versus classical split learning: multiple-client experiments

In this section, we experiment with HQSL with an increasing number of clients to determine its potential for accommodating multiple clients. We then discuss our empirical findings comparing HQSL and SL in a multiple-client setup.

First, to assess the impact of the number of clients K on model accuracy and F1-score, we divide the training set into K independent and identically distributed (IID) subsets for $\mathsf{K} \in \{2, 3, 4, 5, 10, 20, 50, 100\}$. Each subset represents an IID distribution of the dataset assigned to each client, ensuring a balanced portion for each client.

The training process follows Algorithm 2. The hyperparameters used are the same as in the case of a single client in Section 5.1. At the end of each global epoch, we evaluate the model on the testing set. We repeat this training process for 100 global epochs on the multivariate datasets (Botnet DGA and Breast Cancer) and 50 global epochs on the image datasets (MNIST, FMNIST, and Speech Commands spectrograms). The test results are then compared to those of their classical equivalent. The reported results include the test accuracy and F1-score at the end of training comparing HQSL against SL. In this set of experiments, these tests are conducted for only one fold, and the test accuracies and F1-scores are reported to evaluate the scalability of HQSL compared to SL.

Results and discussions

Figure 9 depicts the accuracy and F1-score of HQSL as the number of clients K increases compared to SL.

For both HQSL and SL, the accuracy and F1-score do not show a significant drop in performance across all datasets as we increase the number of clients, K. The result suggests that HQSL can match, and in some cases potentially surpass, the performance of traditional SL as the client count increases. This finding underscores the viability of HQSL.

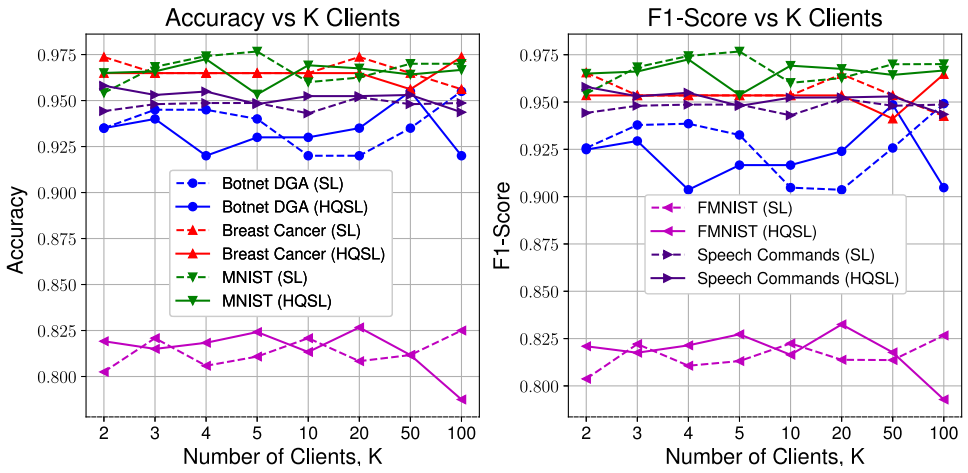


Fig. 9 Test accuracy and F1-score performances of HQSL compared with SL on all 5 datasets for up to $K = 100$ clients. HQSL maintains a high performance like SL while accommodating multiple clients

While our results are promising, we acknowledge the presence of some fluctuations in the testing outcomes. These variations can be attributed to our current experimental design, which utilized a single fold of the dataset. Although we cannot assert HQSL's superiority over SL in multi-client scenarios based on the current result, the evidence strongly supports HQSL's feasibility and scalability. The comparable performance between HQSL and SL across increasing client numbers is a significant finding, opening up new avenues for quantum-enhanced distributed learning in real-world applications.

We used an IID dataset distribution among our K clients to ensure an even and balanced distribution of the dataset for each client. Moreover, in this study, we only considered the round-robin communication protocol for training each client with a centralized server. However, there are more sophisticated methods available in academic literature, such as asynchronous training. In addition, increasing the number of clients increases the training latency for a particular client in the network, which is further exacerbated by the round-robin communication protocol we use. While this study of HQSL's scalability with multiple clients does not address cases involving unbalanced dataset distribution across clients, more advanced communication protocols among clients, or the reduction of the training latency of each client, we defer these for future research.

By leveraging the scalability property of split learning, HQSL can accommodate multiple clients without dropping performance. The performance trend is similar to that of the SL counterpart.

6 Enhancing the resistance of HQSL against reconstruction attacks

In this section, we address the second research question *RQ 2* stated as follows: *How do we strengthen such hybrid quantum split learning schemes against data privacy leakage and reconstruction attacks?* Specifically, we investigate the susceptibility of HQSL to reconstruction attacks using smashed data on the server side. First, we introduce the threat model, outlining the potential risks and vulnerabilities of HQSL to reconstruction attacks. Then, we present the reconstruction attack models we consider in this work to recreate a client's raw input data. These models are designed to exploit the smashed data on the server side.

Considering the rotational properties of the encoding gates in our quantum layer, we propose a noise-based defense mechanism in the form of a Laplacian noise layer. The split structure of HQSL conveniently allows us to insert a noise layer at the client–server boundary, directly on the transmitted smashed data. Specifically, this layer adds controlled randomness to the smashed data before they are transmitted to the server-side model, making it more difficult for the reconstruction attack models to recreate private input raw data. Next, we conduct experiments under different noise parameter settings to extract the best-performing noise layer whereby HQSL has a clear advantage over SL in two key aspects: (i) impairing the performance of the reconstruction attack models and (ii) maintaining a high classification performance despite the presence of the noise defense layer. In our experiments, we investigate reconstruction attacks considering the image datasets introduced in Section 5, specifically, MNIST, Fashion-MNIST, and the spectrograms from the Speech Commands datasets.

6.1 Reconstruction attack from smashed data in HQSL

In this section, we present our threat model that describes how an honest but curious server adversary can recreate the private input raw data from the smashed data transmitted by the client-side model. The adversary uses a reconstruction attack model to recreate the client's input data at inference

time. Hence, we subsequently describe three such reconstruction attack models that can be employed by the adversary. We then describe how we design our proposed defense mechanism to mitigate the risk of reconstruction attacks in HQSL. This countermeasure entails the introduction of a Laplacian noise layer at the end of the client-side model, designed and configured considering the rotational properties of encoding gates in the server-side quantum layer.

6.1.1 Threat model

In this work, we operate under the premise that we have a server that collaborates with multiple clients or data owners according to the SL setup. The clients do not share their private dataset, $\mathcal{D}_{\text{priv}}$, with the server or other clients in the network. We assume that the server is honest but curious (semi-honest), i.e., the server does not deviate from the specified protocol instructions but attempts to infer information about the client's data. We assume that the server has access to a publicly available auxiliary dataset, $\mathcal{D}_{\text{aux}}$, that has a similar distribution as $\mathcal{D}_{\text{priv}}$, but $\mathcal{D}_{\text{priv}}$ and $\mathcal{D}_{\text{aux}}$ are non-overlapping disjoint datasets. The access to such a dataset is a common assumption made by previous works (Dougherty et al. 2023; Pasquini et al. 2021; Shokri et al. 2017). As explained in Pasquini et al. (2021), the adversary can have access to a "shadow" model, $\mathsf{f}_{\text{shadow}}$, to generate outputs, Z_{aux}, in the same feature space as the smashed data, Z_{priv}, received from a client. The shadow model is trained on the auxiliary dataset, $\mathcal{D}_{\text{aux}}$. The server then uses the pair $(\mathsf{Z}_{\text{aux}} = \mathsf{f}_{\text{shadow}}(\mathsf{X}_{\text{aux}}), \mathsf{X}_{\text{aux}})$, where $\mathsf{X}_{\text{aux}} \subset \mathcal{D}_{\text{aux}}$ to train a reconstruction attack model, f_{rec}, that recreates X_{aux}.

With the trained reconstruction attack model, f_{rec}, and the smashed data, Z_{priv}, generated from the private dataset, $\mathcal{D}_{\text{priv}}$, the honest-but-curious server reconstructs the private input of the clients. *In this work, we assume that reconstruction attacks happen at deployment/inference time. This work does not address split learning's susceptibility to data privacy leakage and reconstruction attacks during the training phase.*

6.1.2 Reconstruction attack mechanism

We consider three distinct reconstruction model architectures to reconstruct the private input data, $\mathsf{X}_{\text{priv}} \subset \mathcal{D}_{\text{priv}}$, during a reconstruction attack using the smashed data, Z_{priv}. We train the reconstruction models on the auxiliary dataset, $\mathcal{D}_{\text{aux}}$. During training, the inputs to these models are the smashed data, Z_{aux}, and the outputs are the reconstructed images, $\hat{\mathsf{X}}_{\text{aux}}$. These models are obtained from the literature, and we will assess the performance of our proposed defense mechanism (see Section 6.1.3) against these trained attack models on both HQSL and SL. Since the dimension of the smashed data, $dim(\mathsf{Z}_{\text{priv}}) = 3$, and the dimension of the client's input data, $dim(\mathsf{X}_{\text{priv}}) = 28$ x 28, for the image datasets (MNIST, FMNIST, and Speech Commands spectrograms), we adjust the input and output layer dimensions of the reconstruction models. We describe the three attack models we investigate in this work as follows:

1. **Reconstruction Model 1**: We use the reconstruction attack model architecture proposed by Vepakomma et al. 2020, which performs feature inversion to reconstruct images from low-dimensional intermediate features. The model is a neural network with transpose convolutions. It consists of skip connections such as those used in `ResNet` models, with `ReLU` activation and batch normalization.
2. **Reconstruction Model 2**: Li et al. 2018 proposed a fully convolutional autoencoder for image feature extraction. We adopt the decoder part of the autoencoder to reconstruct the input images. Upsampling layers are used to recover the feature maps along with deconvolutional layers with `ReLU` activation function. Batch normalization is used after each deconvolutional layer except for the last layer. The Euclidean/`MSELoss` loss function is used to compute the difference between the original and reconstructed images. We use a learning rate of 0.001 and `Adam` optimizer to train Reconstruction Model 2.
3. **Reconstruction Model 3**: In Balle et al. (2022), a fully connected model consisting of 2 hidden layers with a width of 1000 and the `ReLU` activation function is used as the decoder part of the autoencoder. The loss function used here is the mean absolute error (MAE/`L1Loss`) + mean square error (`MSELoss`). A learning rate of 0.001 with the `RMSprop` optimizer is used to train Reconstruction Model 3.

After training the reconstruction models, they are deployed to recreate a client's private input. In Fig. 10, we illustrate the setup of a reconstruction attack, on the private data from a client, X_{priv}, using their smashed data, Z_{priv}, to successfully recreate their private input data, represented by $\hat{\mathsf{X}}_{\text{priv}}$.

The smashed data, Z_{priv}, transmitted to the server model consists of latent information about the client's private input data, X_{priv}, which can be reconstructed by the setup shown in Fig. 10. In Section 6.3.1, the baseline results for the image comparison metrics demonstrate high similarities between the original (X_{priv}) and reconstructed images ($\hat{\mathsf{X}}_{\text{priv}}$). Hence, to maintain the privacy of the client's input data, the smashed data communicated to the server model must be desensitized to reduce the possibility of successful reconstruction attacks. Thus, we need a defense mechanism to mitigate the risks of reconstruction of private input raw images from smashed data in HQSL.

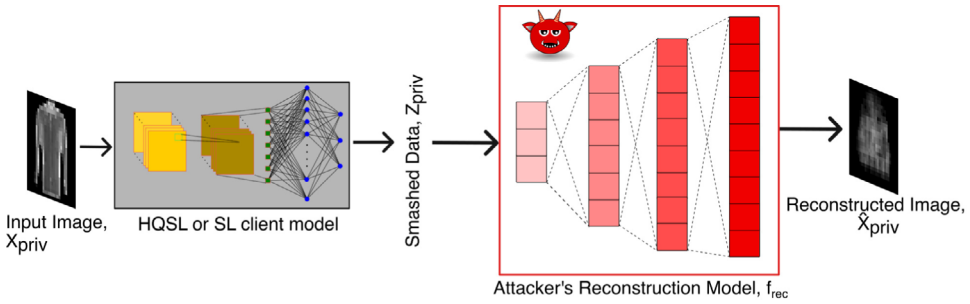


Fig. 10 Setup for reconstruction attacks using Z_{priv} obtained from either HQSL or SL's client model to reconstruct X_{priv}. $\hat{X}_{priv}$ is the reconstructed version of X_{priv}

6.1.3 Encoding gate-based noise layer defense mechanism

To defend HQSL against the risk of reconstruction attacks, we propose a Laplacian noise layer defense mechanism at the end of the client-side model. Previous works have considered the use of a noise layer defense mechanism to introduce perturbation to the smashed data and, hence, increase the difficulty of reconstruction by the proposed reconstruction models. However, such noise defense methods come with a privacy-utility trade-off (Titcombe et al. 2021; Na et al. 2024; Mireshghallah et al. 2020). In this section, we study the rotational properties of quantum encoding gates to design the noise layer to (i) effectively hinder the reconstruction attack and (ii) maintain a high classification performance (accuracy and F1-score) by the server during inference. This, hence, addresses the privacy-utility trade-off associated with the noise layer defense mechanism. We empirically compare these two aspects for the hybrid (HQSL) and classical (SL) cases in Sections 6.2 and 6.3.

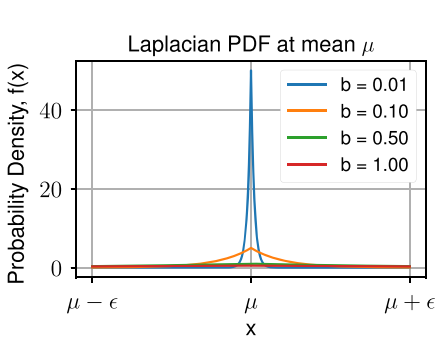
Laplacian noise is favored over other types of noise, such as Gaussian noise, for privacy protection or data obfuscation. This is because noise sampled from a Laplacian distribution, Laplace(μ, b), can be systematically added to the smashed data, Z_{priv}, to provide controlled randomness, while enabling the explicit quantification of the level of perturbation introduced. This is important for maintaining privacy while retaining data utility. Laplacian noise is also commonly utilized in the context of differential privacy (Shokri and Shmatikov 2015; Sarathy and Muralidhar 2011).

The location (mean) parameter μ shifts the distribution, while the scale parameter b controls its spread. In our setting, this is critical: the scale parameter determines how tightly noise samples concentrate around the mean. As shown in Fig. 11a, smaller b values result in a sharper distribution centered at μ, keeping the noise values sampled close to the desired noise level. This sharpness allows us to maintain high fidelity in the quantum state encoding while still introducing enough variability to hinder adversarial reconstruction, as we show next.

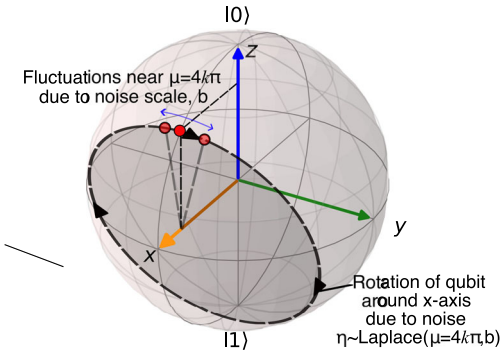
Noise layer design based on rotational properties of RX-gates Adding noise sampled from a Laplacian distribution, with appropriately tuned mean and scale parameters, has the potential to benefit the quantum circuit on the server side by maintaining high classification performance while impeding reconstruction of raw data via the added jitter. In Section 6.3, we will demonstrate that a purely classical server cannot simultaneously achieve these goals.

We consider the following to develop a better understanding of our selection of Laplacian noise parameters. The Bloch sphere gives a geometrical representation of the pure-state space of a qubit. Single-qubit rotation gates, e.g., the RX-gate, move the state vector around the surface of the Bloch sphere, as shown in Fig. 11b. In our proposed HQSL circuit (Fig. 2b), each RX-gate has a 4π period and encodes a

Fig. 11 **a** As the scale value b decreases, the Laplacian noise becomes more concentrated around the mean, μ, resulting in a higher density of data points near μ. **b** The addition of this noise (mean $\mu = 4k\pi, k \in \mathbb{Z}$, and small scale b) to the smashed data causes only a small fluctuation near the original location (red dot) of the qubit about the x-axis on the Bloch sphere



(a) Probability density function (PDF) of Laplacian noise centered at mean μ with varying scale parameter, b



(b) Rotation of qubit (red dot) around the x-axis of the Bloch sphere after adding Laplacian noise, η, with mean $\mu = 4k\pi$ and small scale, b

classical feature z by rotating the qubit about the x-axis by angle $\theta = z$. Adding noise η to the smashed data, hence, $\tilde{\theta} = z + \eta$, corresponds to applying

$$RX(\tilde{\theta}) \;=\; RX(z+\eta)\,. \tag{7}$$

By the 4π periodicity of $RX(\theta)$, centering our Laplace noise at any mean $\mu = 2k\pi$ ($k \in \mathbb{Z}$) ensures that as $b \to 0$,

$$RX(z+\eta) \;\approx\; RX(z) \quad \text{(up to a global phase)}, \tag{8}$$

so that the encoded quantum state is preserved. By computing the fidelity between the clean and noisy encoded quantum states, we formalize this below.

Expected fidelity preservation Using the matrix representation of our encoding RX-gate, the encoded quantum state, $|\psi(z)\rangle$, can be written as follows:

$$|\psi(z)\rangle = RX(z)|0\rangle = \cos\tfrac{z}{2}|0\rangle - i\sin\tfrac{z}{2}|1\rangle, \tag{9}$$

and with noise $\eta \sim \text{Laplace}(0, b)$ we have

$$|\psi(z+\eta)\rangle = \cos\tfrac{z+\eta}{2}|0\rangle - i\sin\tfrac{z+\eta}{2}|1\rangle. \tag{10}$$

The state fidelity is

$$F(\eta) = \left|\langle\psi(z)\mid\psi(z+\eta)\rangle\right|^2 = \cos^2\tfrac{\eta}{2} = \tfrac{1+\cos\eta}{2}. \tag{11}$$

Taking expectation over η and using the Laplace characteristic function yields

$$\mathbb{E}_\eta[F] = \frac{1}{2}\Big(1 + \mathbb{E}[\cos\eta]\Big) = \frac{1}{2}\Big(1 + \tfrac{1}{1+b^2}\Big). \tag{12}$$

In particular, for $b \ll 1$, we have $\mathbb{E}_\eta[F] \approx 1$. This implies that the clean and noisy encoded quantum state on the server-side quantum layer remains unchanged as $b \to 0$. Hence, keeping the scale parameter b small limits the noise to a minimal perturbation about the mean, as illustrated by the sharp Laplacian PDF in Fig. 11a. This low variability ensures consistent encoding that closely matches the clean quantum state, preserving the hybrid server model's utility.

In Fig. 12, we illustrate the defense setup at inference time: the client's smashed data $\mathsf{Z}_{\mathsf{inf}}$ is perturbed to $\tilde{\mathsf{Z}}_{\mathsf{inf}}$ via the Laplacian noise layer before entering the server's quantum circuit, thereby obscuring the input against reconstruction attacks while retaining classification accuracy.

We also include Algorithm 4 to outline the deployment of the Laplacian noise-based defense mechanism at inference time as a countermeasure against reconstruction attacks on the server side. The algorithm describes the process of obfuscating the smashed data $\mathsf{Z}_{\mathtt{inf}}$ before sending $\tilde{\mathsf{Z}}_{\mathtt{inf}}$ to the HQSL server. Depending on the selected mode, the server either performs classification using the hybrid quantum server model or attempts to reconstruct the original input $\mathsf{X}_{\mathtt{inf}}$.

6.2 Experiments on encoding gate-based noise defense

In this section, we consider the single-client single-server split learning setup and describe our experiments to test and tune our defense mechanism with varying Laplacian noise layer parameters by (i) comparing the differences between reconstructed ($\hat{\mathsf{X}}_{\mathsf{inf}}$) and original ($\mathsf{X}_{\mathsf{inf}}$) images in hybrid and classical settings and (ii) comparing the classification performances of HQSL and SL in the presence of the noise layer. From these experiments, we devise the optimal noise parameters that give HQSL a distinct advantage over SL. We conduct our experiments on Reconstruction Model 1 to tune our Laplacian noise parameters μ and b. In Appendix E, we present the reconstruction performances of the other two reconstruction models presented in Section 6.1.2 when we

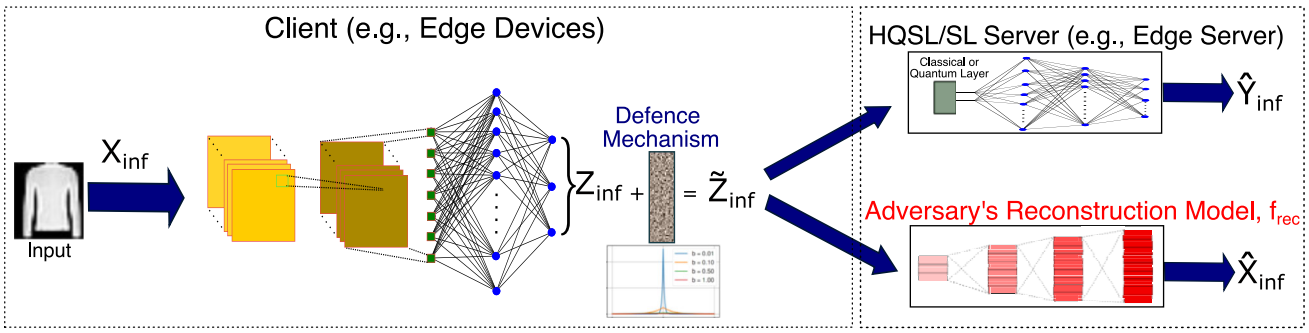


Fig. 12 At inference time, $\mathsf{X}_{\mathsf{inf}}$ is the input to the client model with the Laplacian noise layer as the defense mechanism. Laplacian noise is added to the smashed data, $\mathsf{Z}_{\mathsf{inf}}$, resulting in $\tilde{\mathsf{Z}}_{\mathsf{inf}}$. $\tilde{\mathsf{Z}}_{\mathsf{inf}}$ is communicated to the HQSL or SL server to carry out the classification task with outputs $\hat{\mathsf{Y}}_{\mathsf{inf}}$ as intended or fed to a reconstruction model to recreate private input data as $\hat{\mathsf{X}}_{\mathsf{inf}}$

Algorithm 3 HQSL with Laplacian noise defense deployed at inference time.

Input: Pre-trained Client Model f_c with weights $\tilde{\mathbf{W}}^C$, Server Model with classical layers f_s and weights $\tilde{\mathbf{W}}^S$, quantum layer parameters $\tilde{\mathbf{\Theta}}$, Adversary Reconstruction Model $f_{\mathtt{rec}}$, Input Data $\mathsf{X}_{\mathtt{inf}}$, Mode Selection $mode$ $(classification/attack)$
Output: Classification Output $\hat{\mathsf{Y}}_{\mathtt{inf}}$ (if classification mode) or Adversary's Reconstructed Input $\hat{\mathsf{X}}_{\mathtt{inf}}$ (if attack mode)
Notations: Client-side smashed data: $\mathsf{Z}_{\mathtt{inf}}$, quantum layer expectation output: $\mathbf{E}_{\mathtt{inf}}$

START OF INFERENCE
Client Side: `// Forward Pass`
Compute smashed data: $\mathsf{Z}_{\mathtt{inf}} = f_c(\mathsf{X}_{\mathtt{inf}}, \tilde{\mathbf{W}}^C)$
Sample Laplacian noise: $\eta \sim \text{Laplace}(\mu, b)$ `// Noise sampled from Laplace distribution with mean` μ `and scale` b
Add noise to smashed data: $\tilde{\mathsf{Z}}_{\mathtt{inf}} \leftarrow \mathsf{Z}_{\mathtt{inf}} + \eta$ `// Obfuscate smashed data`
Send $\tilde{\mathsf{Z}}_{\mathtt{inf}}$ to the server

Server Side: `// Either Classification or Adversary Attack`
if $mode = classification$ **then**
Compute expectation values: $\mathbf{E}_{\mathtt{inf}} = f_q(\tilde{\mathsf{Z}}_{\mathtt{inf}}, \tilde{\mathbf{\Theta}})$ `// Quantum Layer`
Compute classification output: $\hat{\mathsf{Y}}_{\mathtt{inf}} = f_s(\mathbf{E}_{\mathtt{inf}}, \tilde{\mathbf{W}}^S)$ `// Classical Layers`
return $\hat{\mathsf{Y}}_{\mathtt{inf}}$
else if $mode = attack$ **then**
Attempt to reconstruct input: $\hat{\mathsf{X}}_{\mathtt{inf}} \leftarrow f_{\mathtt{rec}}(\tilde{\mathsf{Z}}_{\mathtt{inf}})$ `// Adversary Reconstruction`
return $\hat{\mathsf{X}}_{\mathtt{inf}}$

Main Execution:
Load input data $\mathsf{X}_{\mathtt{inf}}$ and compute smashed data $\mathsf{Z}_{\mathtt{inf}}$
Compute $\tilde{\mathsf{Z}}_{inf} \leftarrow$ Sample and add Laplacian noise η to $\mathsf{Z}_{\mathtt{inf}}$
if $mode = classification$ **then** $\hat{\mathsf{Y}}_{\mathtt{inf}} \leftarrow$ Inference($\tilde{\mathsf{Z}}_{\mathtt{inf}}$)
else if $mode = attack$ **then** $\hat{\mathsf{X}}_{\mathtt{inf}} \leftarrow$ Adversary Reconstruction($\tilde{\mathsf{Z}}_{\mathtt{inf}}$)

END OF INFERENCE
return $\hat{\mathsf{Y}}_{\mathtt{inf}}$ (if classification mode) or $\hat{\mathsf{X}}_{\mathtt{inf}}$ (if attack mode)

apply our proposed noise defense mechanism with the configured parameter values μ and b.

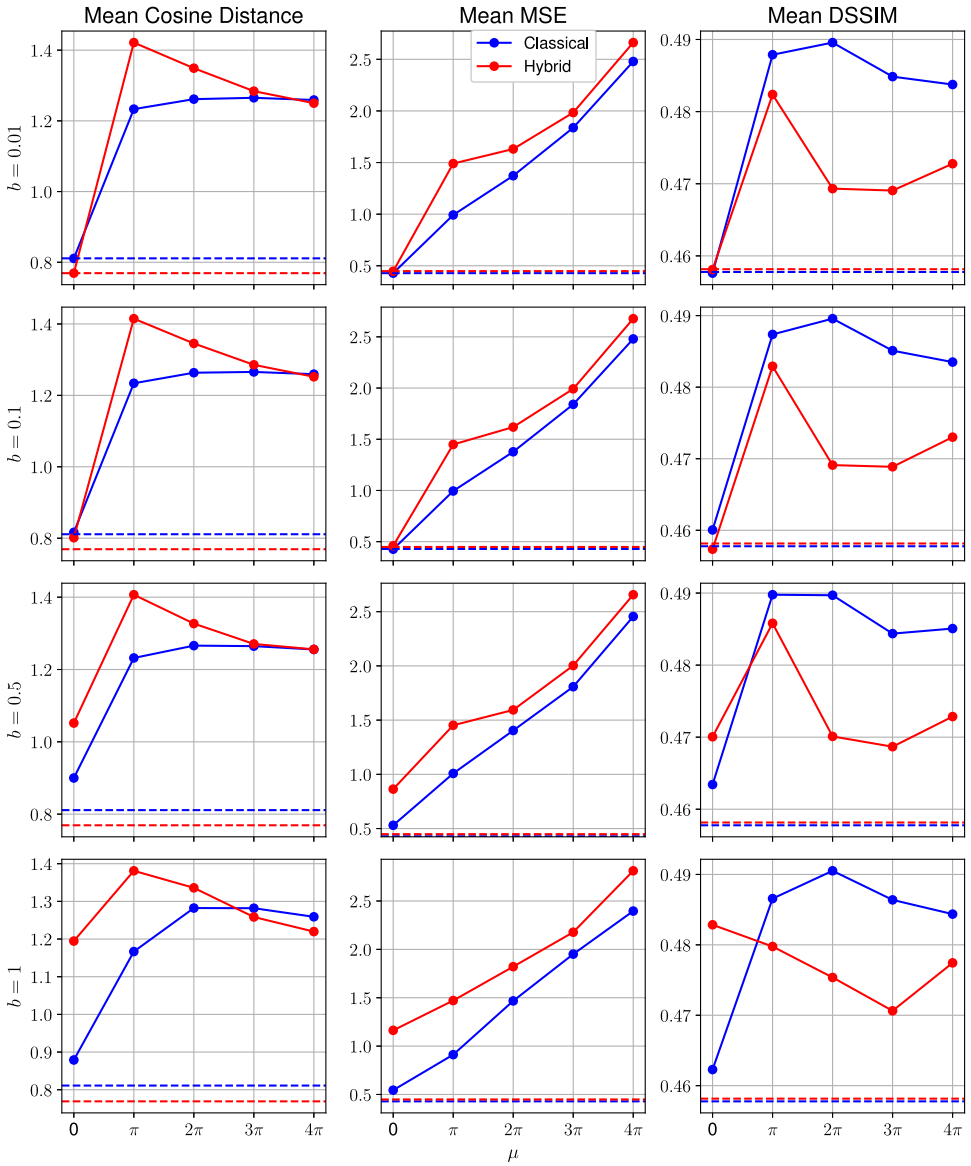
We split each dataset in the $\mathsf{X}_{\mathsf{train}}$: $\mathsf{X}_{\mathsf{test}}$ of 4: 1 ratio. We train the client and server models in both HQSL and SL settings using the training dataset, $\mathsf{X}_{\mathsf{train}}$. The training process takes place as described in Section 5.1. We further split the test dataset, $\mathsf{X}_{\mathsf{test}}$, in the ratio $\mathsf{X}_{\mathsf{rec}}$: $\mathsf{X}_{\mathsf{inf}}$ of 4: 1. $\mathsf{X}_{\mathsf{rec}}$ represents a subset of the auxiliary dataset, $\mathsf{X}_{\mathsf{rec}} \subset \mathcal{D}_{\mathsf{aux}}$, as described in our threat model (Section 6.1.1), to train the reconstruction model. Here, each dataset is split into three subsets such that they are disjoint from each other. We pass $\mathsf{X}_{\mathsf{rec}}$ to the trained client model to generate the smashed data, $\mathsf{Z}_{\mathsf{rec}}$. This step assumes that the client model represents the shadow model, $\mathsf{f}_{\mathsf{shadow}}$, described in our threat model. We generate the smashed data, $\mathsf{Z}_{\mathsf{rec}}$, from the auxiliary dataset and use the data pair ($\mathsf{Z}_{\mathsf{rec}}$, $\mathsf{X}_{\mathsf{rec}}$) to train the adversary's reconstruction model, $\mathsf{f}_{\mathsf{rec}}$.

We use the stochastic gradient descent, `SGD`, optimizer, with a learning rate of 10^{-3} and momentum of 0.9, to train the reconstruction model for 200 epochs using the MAE/`L1Loss` loss function. After training the adversary's reconstruction model, we introduce the Laplacian noise layer, with mean $\mu \in \{0, \pi, 2\pi, 3\pi, 4\pi\}$ and scale $b \in \{0.01, 0.1, 0.5, 1\}$, at the end of the previously trained client-side model. We consider this range of values for the mean μ because it represents the location of a qubit at different points along a full rotation around the x-axis of the Bloch sphere. Notably, due to the 4π-periodicity of quantum states under such rotations, angles beyond 4π do not provide additional insights while tuning our Laplacian noise defense layer, as they result in equivalent quantum states. For example, mean values $\mu = 6\pi$ and 8π would be redundant as they are equivalent to $\mu = 2\pi$ and 4π, respectively, as the state evolution repeats cyclically with a period of 4π. The cases of $\mu = 2\pi$ and 4π are particular relevant, as they are expected to cause the quantum states to be invariant in the presence of our noise layer given a small scale b as explained in Section 6.1.3.

Next, $\mathsf{X}_{\mathsf{inf}}$ represents a portion of the private data of a client at inference time, $\mathsf{X}_{\mathsf{inf}} \subset \mathcal{D}_{\mathsf{priv}}$, that we aim to reconstruct using the trained reconstruction model. However, due to the noise layer, the client model now outputs noisy smashed data, $\tilde{\mathsf{Z}}_{\mathsf{inf}}$. $\tilde{\mathsf{Z}}_{\mathsf{inf}}$ represents the input to the reconstruction model during the attack phase to recreate $\mathsf{X}_{\mathsf{inf}}$ as $\hat{\mathsf{X}}_{\mathsf{inf}}$.

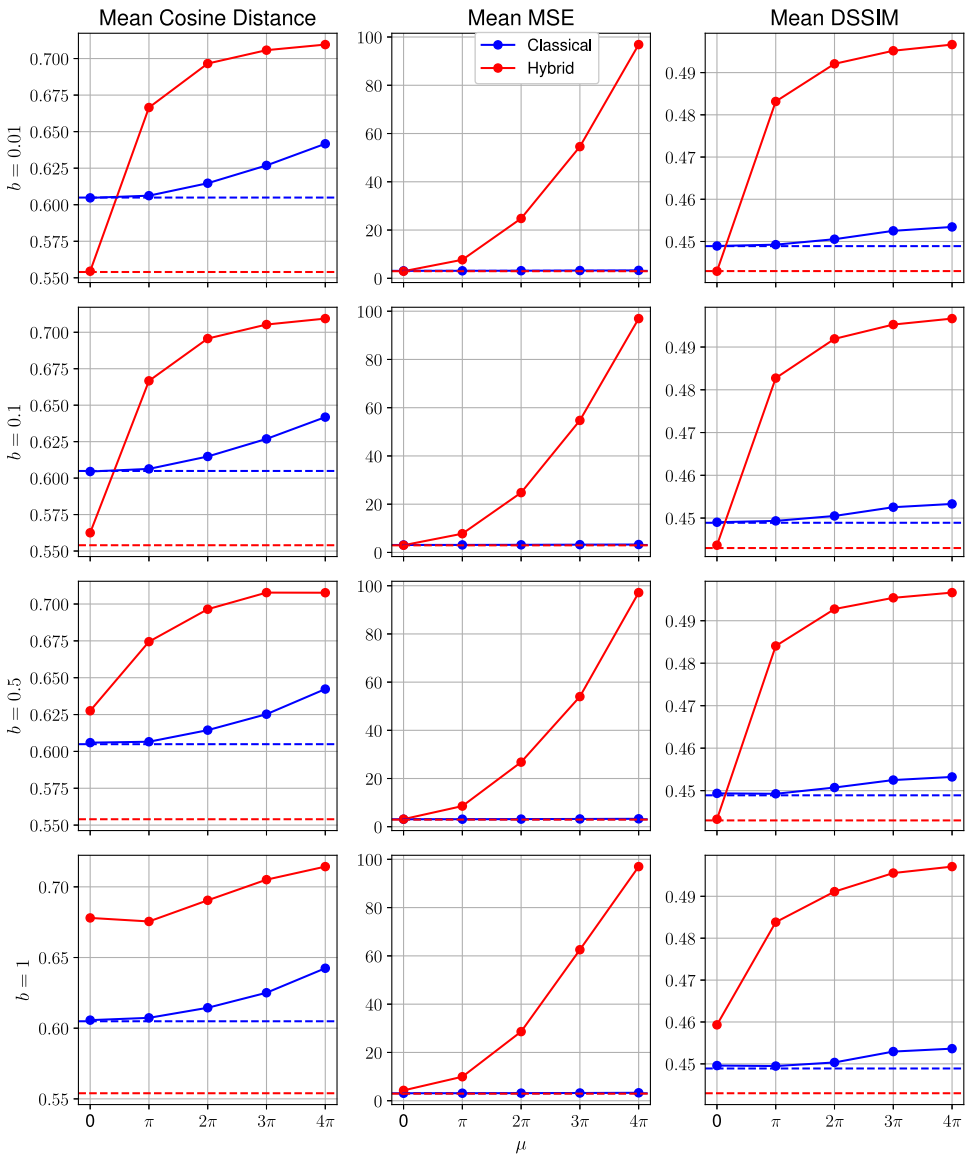
We evaluate the reconstruction performance by computing the difference between the original ($\mathsf{X}_{\mathsf{inf}}$) and reconstructed images ($\hat{\mathsf{X}}_{\mathsf{inf}}$). For the Fashion-MNIST and MNIST datasets, we use three image comparison metrics: cosine distance, mean square error (MSE), and structural dissimilarity index measure(DSSIM). For the spectrograms from the Speech Commands dataset, the 3 metrics we employ are cosine distance, DSSIM, and log-spectral distance (LSD). We used LSD for the Speech Commands spectrograms instead of MSE as LSD gave more significant results than MSE. We describe these metrics in more detail in Appendix D. To ensure a fair comparison, we apply a masking function to the original and reconstructed images. This masking function takes 2 images of the same dimensions and extracts the non-zero pixel values from both images at positions where at least one image has a non-zero pixel. These masked images are then used for further analysis and comparison using the image comparison metrics. The means of these metrics are taken over a batch of testing data. We present our findings in Section 6.3.1.

For evaluating the classification performance at inference time, we compare the test accuracies and F1-scores under HQSL and SL settings for five folds, now in the presence



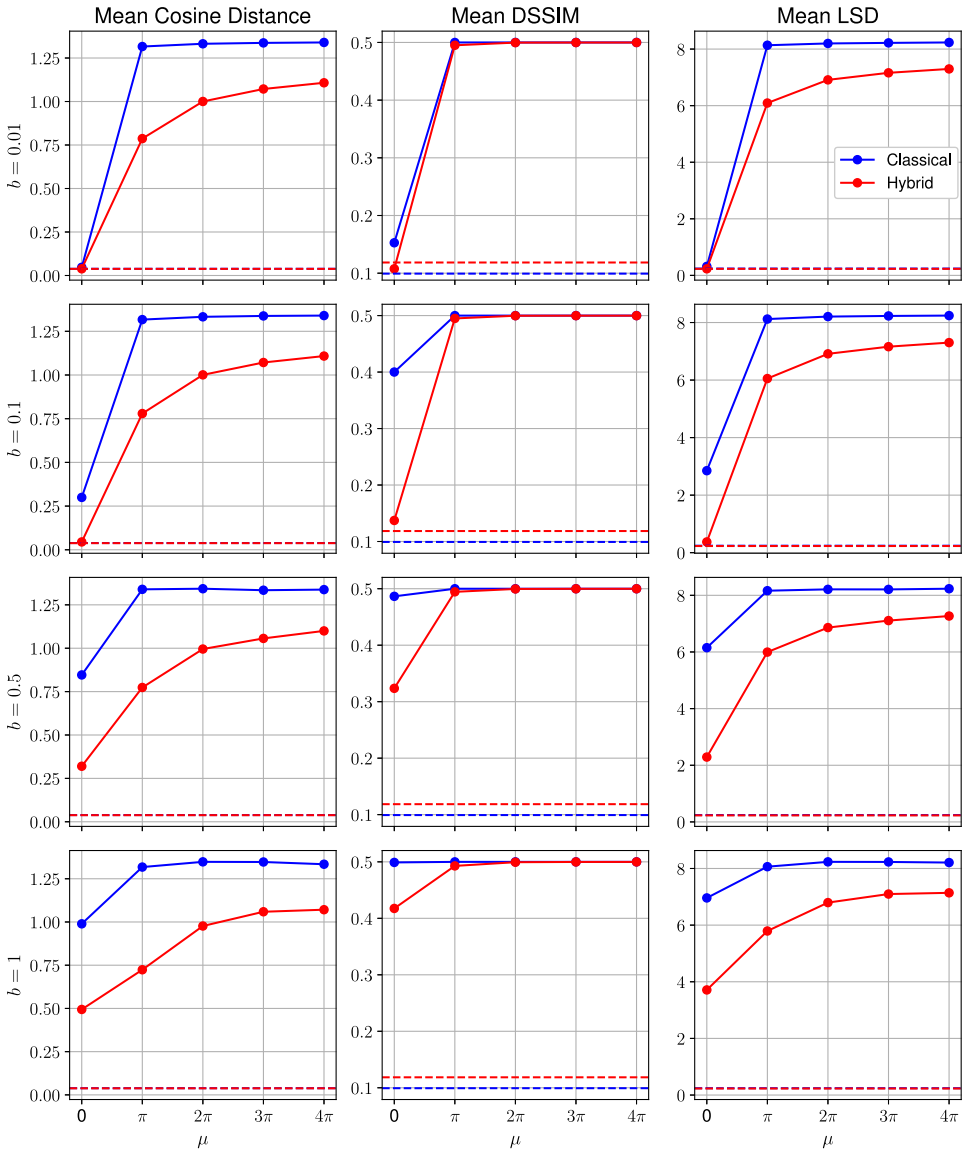
(a) Effect of varying noise parameters μ and b on FMNIST dataset

Fig. 13 Reconstruction performance with varying Laplacian noise layer parameters. The horizontal dashed lines represent the baseline values for each metric, with each line's color corresponding to the legend. Irrespective of the scale parameter value, b, at mean $\mu = 4\pi$, we obtain, in general, the largest deviations from the baseline. This finding is consistent for all datasets and metrics considered



(b) Effect of varying noise parameters μ and b on MNIST dataset

Fig. 13 continued



(c) Effect of varying noise parameters μ and b on Speech Commands spectrograms dataset

Fig. 13 continued

of the Laplacian noise layer with parameters μ and b. These results are discussed in Section 6.3.2.

6.3 Reconstruction attacks results and discussion

In this section, we present and discuss our findings from the experiments carried out to test our proposed encoding gate-based noise defense layer as a countermeasure against reconstruction attacks. We evaluate the performance of the reconstruction model by using the metrics described earlier in Section 6.3.1. This is done in both classical and hybrid settings and for different Laplacian noise parameter values, μ and b. Next, we analyze HQSL's inference-time accuracies and F1-scores across five folds, fixing the mean parameter μ while varying the scale parameter b, as detailed in Section 6.3.2. These findings provide insights into optimizing the noise parameters to enhance HQSL's defense against reconstruction attacks while maintaining robust inference-time classification performance compared to SL, despite the added noise layer.

6.3.1 Comparing the difference between original and reconstructed images under HQSL and SL settings at different noise levels

To evaluate the performance of the reconstruction model in classical and hybrid settings in the presence of Laplacian noise, we investigated the impact of noise mean, μ, and scale, b, in reconstructing Fashion-MNIST, MNIST, and spectrogram images. Our comparative analysis focused on 4 key image comparison metrics: Mean Cosine Distance, Mean MSE, Mean Structural Dissimilarity Index (Mean DSSIM), and Mean Log Spectral Distance (Mean LSD). The results for the FMNIST, MNIST, and Speech Commands spectrograms datasets are depicted in Fig. 13a, b, and c, respectively. The baseline values (represented by the horizontal dashed lines) correspond to the noise-free implementation of the reconstruction attack, i.e., when the smashed data transmitted to the reconstruction model are free from Laplacian noise. In both the classical and hybrid settings, the baseline results are closer to 0 than when the noise layer is applied, showing the reconstruction attacks are successful in recreating the private inputs. These results, hence, underscore the need for the defense mechanism that we investigate in this section.

The general trends from Fig. 13a and b were that as we increased the mean parameter, μ, the metric values were well above the baseline, irrespective of the noise scale parameter, b. Notably, on the MNIST dataset, the deviations from the baseline in the hybrid setting were more significant than in the classical setting for all 3 metrics. This shows that there was a larger difference between original and reconstructed images in the hybrid setting than in the classical setting. In the classical setting, the minimal deviations from the baseline indicate that a reconstruction attack can still be successful for the MNIST dataset, regardless of the noise parameters. This suggests that, at least for the MNIST dataset, the reconstruction model in the classical setting appears relatively insensitive to changes in the noise parameters, with its baseline performance being largely preserved regardless of the trialled noise configurations. We also highlight that at $\mu = 4\pi$, irrespective of the scale parameter value, b, the largest deviations were obtained, particularly in the hybrid setting, indicating that at this μ value, the additive noise was successfully hindering the reconstruction performance of the adversary's model. We also note that significant deviations were still obtained at $\mu = 2\pi$. These findings underscore that the mean values $\mu = 2\pi$ and 4π caused the performance of the adversary's reconstruction model to drop in the hybrid setting.

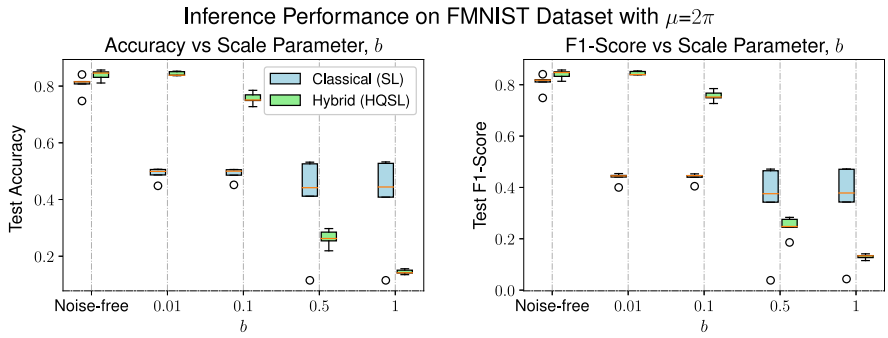
The results for the Speech Commands spectrograms shown in Fig. 13c indicated similar findings. The baseline results were very close to 0, showing that in the absence of the noise layer, the reconstructed spectrograms were very similar to the original spectrograms. As we increased the mean, μ up to 4π, the deviations increased. For this dataset, the mean cosine distance and mean LSD metrics showed that in the classical setting, the deviations from the baselines were more significant than in the hybrid setting. Still, large deviations above the baseline were obtained in the hybrid setting, signaling the increased difficulty of the adversary's model to recreate the original spectrograms. For all scale parameters, b, at $\mu = 2\pi$ and 4π, we obtained significant deviations in both the hybrid and classical settings.

Overall, this study demonstrates that the introduction of the Laplacian noise layer, with appropriately configured noise parameters, can obfuscate the smashed data, thereby hindering the performance of the reconstruction model, and in some cases, to a much greater extent in the hybrid setting. From these results, we establish that mean values of $\mu = 2\pi$ and 4π effectively increase the difference between the original and reconstructed images, hence supporting our analysis in Section 6.1.3. Therefore, we conclude here that *by tuning the Laplacian noise layer parameters considering the rotational properties of encoding gates, we can strengthen the resilience of HQSL against reconstruction attacks. In the classical setup, the reconstruction attack can still successfully recreate private input data.* Next, a necessary investigation is the impact of the perturbation caused by the scale parameter, b on the classification performance of HQSL and SL.

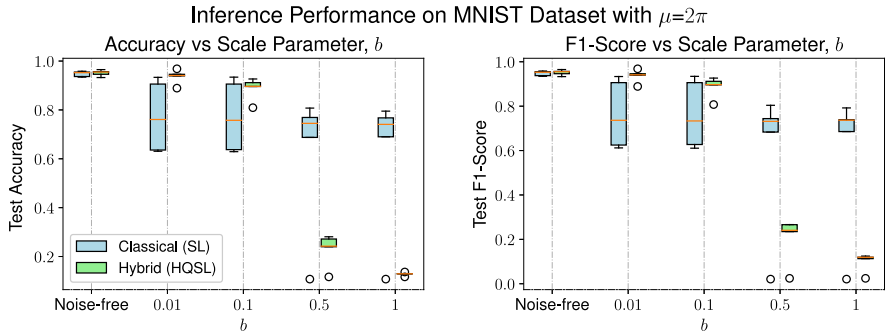
6.3.2 Inference time performance of HQSL and SL at different noise levels

From Section 6.3.1, we established that when $\mu = 2\pi$ and 4π, we obtain large differences between original and reconstructed images indicated by the deviations above the baseline results. In this section, we investigate the effect of varying scale parameters, b, on the inference performance of HQSL and SL given mean values $\mu = 2\pi$ and 4π. We measure the performance by comparing the accuracy and F1-score in classifying FMNIST, MNIST, and Speech Commands spectrogram images with varying scale parameters, b. We present these results using the box plots shown in Figs. 14 and 15, representing the five-fold test accuracies and F1-scores in the presence and absence of noise. The baseline performances are represented by the Noise-free box plots.

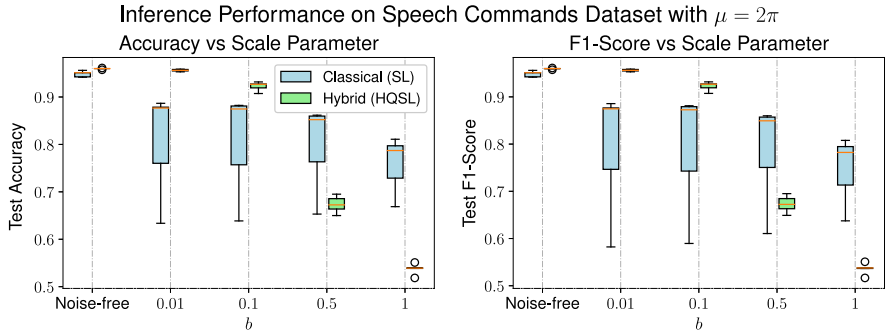
We make the following deductions, backed by the experiments on all 3 datasets. The noise-free box plots representing the baseline performances showed a high accuracy and F1-score with low fluctuations across all folds in both HQSL and SL. The heights of the box plots indicate the fluctuations or variance of the accuracies and F1-score across all folds. By varying the Laplacian noise scale, b, and keeping the mean fixed at $\mu = 2\pi$ (Fig. 14) and $\mu = 4\pi$ (Fig. 15), we demonstrated that a small enough scale parameter value,



(a) Fashion-MNIST: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 2\pi$



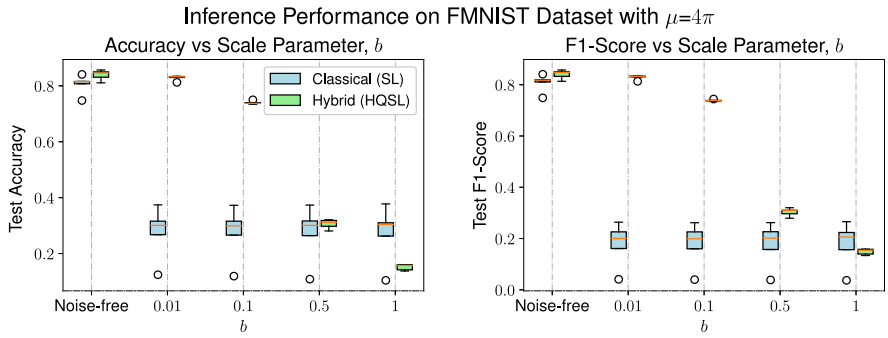
(b) MNIST: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 2\pi$



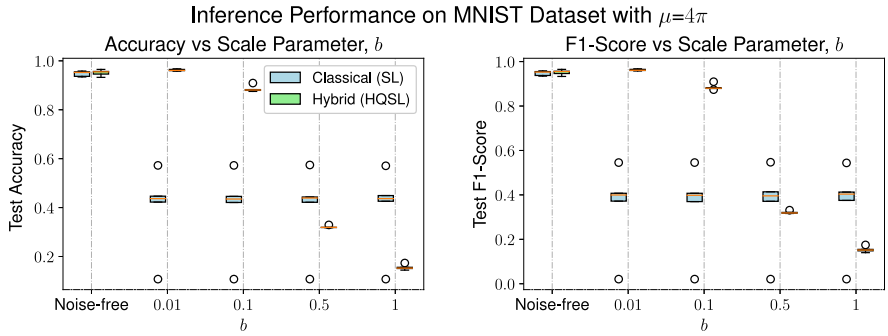
(c) Speech Commands spectrograms dataset: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 2\pi$

Fig. 14 Effect of noise with fixed mean $\mu = 2\pi$ and varying scale b on inference performance in classifying FMNIST, MNIST, and spectrograms from Speech Commands datasets. HQSL (green) maintains a higher accuracy and F1-score with lower variance than SL (blue) in the presence of Laplacian noise with $\mu = 2\pi$ and $b = 0.01, 0.1$

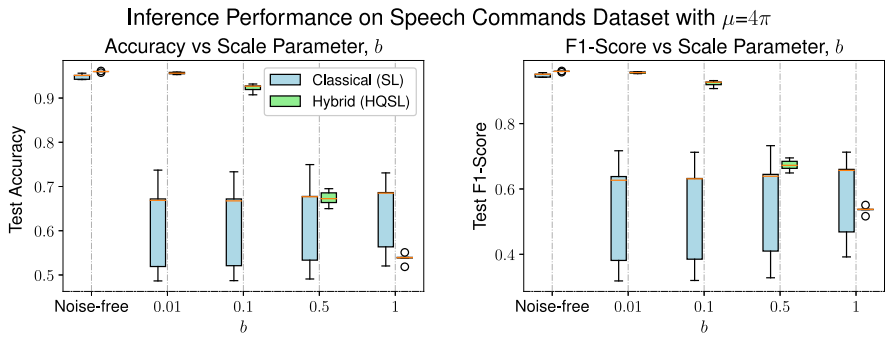
e.g., $b = 0.01$, caused the performance of SL at inference time to drop and show high fluctuations in accuracy and F1-score. In contrast, HQSL maintained a high performance with much less fluctuations – very similar to the noise-free performance. At both $\mu = 2\pi$ and $\mu = 4\pi$, when $b = 0.1$, HQSL still maintained a superior performance to SL but was slightly worse than when $b = 0.01$. This observation shows that when the scale parameter value increased, the performance of HQSL started getting affected by the Laplacian noise layer. As b increased above 0.1, HQSL's performance at inference time dropped even more. Hence, this experiment indicates that the scale value b should be kept close to 0. We, therefore, choose $b = 0.01$ as the desirable scale parameter value for the noise layer as it allows HQSL to maintain a high performance, similar to its noise-free baseline. These experimental findings support our analysis in Section 6.1.3



(a) Fashion-MNIST: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 4\pi$



(b) MNIST: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 4\pi$



(c) Speech Commands spectrograms dataset: 5-fold test accuracy and F1-score versus scale parameter, b, with mean fixed at $\mu = 4\pi$

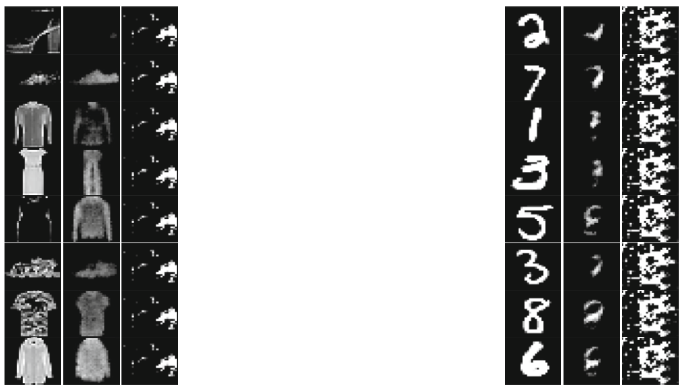
Fig. 15 Effect of noise with fixed mean $\mu = 4\pi$ and varying scale b on inference performance in classifying FMNIST, MNIST, and spectrograms from Speech Commands datasets. HQSL (green) maintains a higher accuracy and F1-score with lower fluctuations than SL (blue) in the presence of Laplacian noise with $\mu = 4\pi$ and $b = 0.01, 0.1$

that the scale parameter value, b, should be configured to a small value to limit the perturbations caused by the noise layer, hence maintaining the HQSL server model's utility.

Hence, we conclude here that, *by tuning the Laplacian noise layer parameters considering the rotational properties of encoding gates, HQSL's classification performance can match the baseline performance, showing robustness to the noise layer. In contrast, SL is less robust to the tuned noise layer.*

Reconstruction performance when HQSL has an advantage

In Fig. 16, we present some visual observations of the reconstruction attack model performance when we employ our



(a) Reconstruction results on FMNIST images (b) Reconstruction results on MNIST images

Fig. 16 Left to right: Original, reconstructed Images in classical and hybrid settings, respectively, with Laplacian noise layer with parameters $\mu = 4\pi$ and $b = 0.01$. For both FMNIST and MNIST images, reconstruction is almost impossible in the hybrid setting, whereas some visual similarities are observed between reconstructed and original images in the classical setting

proposed Laplacian noise layer defense tuned considering the rotational properties of quantum encoding gates (here we test using the parameters $\mu = 4\pi$ and $b = 0.01$).

Figure 16 illustrates that the reconstructed images show more significant differences from the original images in the hybrid setting than in the classical setting. Hence, we can directly confirm that our designed noise layer effectively impedes the reconstruction model in recreating the raw images. This is also depicted in the results shown in Fig. 13. Furthermore, in HQSL, the noise layer causes the smashed data to be handled properly by the quantum layer on the server-side model, hence maintaining a similar classification performance to the noise-free performance, as shown in Figs. 14 and 15. *This is due to the configured mean values of* $\mu = 2\pi$ and 4π, *which causes the quantum state encoded by the RX-gates in the quantum layer to be nearly invariant to the additive Laplacian noise, given the scale parameter is small* ($b = 0.01$). This means that, for the quantum layer, the obfuscated classical intermediate features correspond to quantum states that are close to those encoded from clean classical smashed data.

By leveraging the rotational properties of the encoding gates in the quantum layer of HQSL, we can tune the noise layer parameters to obtain clear advantages in terms of enhancing resistance in split learning against reconstruction attacks from data privacy leakage. The advantages of our proposed noise layer in HQSL are that it (i) impedes the reconstruction attack model's performance in reconstructing raw data and (ii) enables HQSL to maintain a high performance despite its presence. HQSL addresses the privacy-utility tradeoff associated with the use of noise layers to enhance resistance against data privacy leakage in SL.

7 Conclusion

This paper addresses the problem of applying split learning (SL) concepts to pure quantum machine learning (QML) models for resource-constrained clients lacking quantum computing resources. We propose a novel hybrid quantum split learning, HQSL, architecture that enables resource-limited clients in the classical domain to train their models collaboratively with a hybrid quantum server for classification tasks. Specifically, HQSL consists of a hybrid quantum neural network, HQNN, split into two parts: the client side, consisting of a classical neural network, and the server side, comprising an HQNN with the first layer as a quantum layer consisting of a 2-qubit quantum circuit. For the quantum circuit, we introduce a qubit-efficient data-loading technique that allows us to keep the number of qubits and circuit depth small in anticipation of their implementation on near-term quantum devices. Experimental results demonstrated the feasibility and ability of HQSL to improve classification accuracy and F1-score compared to its classical equivalent. Performance evaluations on actual quantum devices and noisy simulators provide initial evidence supporting HQSL's noise resilience and practicality for deployment on near-term, noisy quantum (NISQ) devices. Our experiments also demonstrated the scalability of HQSL with an increasing number of clients as HQSL maintains high performance during classification as we sequentially increase the number of clients.

In this work, we also address the overarching problem of data privacy leakage associated with SL, which can lead to clients' private input data reconstruction attacks on the server side. We propose a Laplacian noise layer defense mechanism designed based on the rotational properties of the encoding

gates present within the quantum layer of HQSL. By tuning the parameters of our proposed Laplacian noise layer defense mechanism, we showed that HQSL is more robust to the privacy-utility tradeoff generally associated with the use of noise to enhance SL's resistance against reconstruction attacks in SL.

Overall, HQSL enables resource-constrained clients to train ML models collaboratively with a server equipped with quantum computing resources, resulting in improved classification performance. The quantum circuit designed in this work underscores a qubit-efficient data-loading mechanism that can be extended beyond the scope of HQSL to address qubit count limitations in near-term devices. Furthermore, HQSL supports multiple clients in the classical domain for collaborative HQNN model training. The defense mechanism proposed in this work enhances HQSL's resistance against data privacy leakage and the risks of reconstruction attacks.

Future work Although our current HQSL prototype uses a relatively small and shallow quantum circuit, it lays the groundwork for more ambitious and feasible HQNN+SL architectures. As fault-tolerant devices become available, one could scale HQSL with richer variational blocks or multiple quantum layers on the server side, enabling even lightly resourced clients to benefit from potential quantum advantages. Future studies could also explore alternative split topologies, e.g., U-shaped splits without label sharing or vertically partitioned data, and benchmark HQSL against federated learning (FL) and quantum federated learning (QFL) baselines to provide a more comprehensive performance context. Moreover, while our experiments demonstrate that the rotationally tuned Laplacian noise layer effectively preserves classification accuracy and impedes reconstruction attacks, a rigorous theoretical analysis, such as formal differential-privacy guarantees or robustness against adaptive adversaries, remains an open area for future research. In addition, future work could investigate defense techniques against a broader range of attack scenarios, beyond reconstruction attacks, e.g., model inversion, membership inference, and quantum-specific attacks, to deepen insights into HQSL's security vulnerabilities and further strengthen its robustness.

Appendix A: Quantum computing background

In this section, we provide further background information on quantum computing supplementing our work. We also discuss the current generation of quantum computers to highlight the regime this paper is relevant to.

Qubits Qubits are the basic unit of information of quantum computing (Nielsen and Chuang 2010). A qubit can be represented mathematically using Dirac notation as $|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$, where $|\psi\rangle$ is the quantum state, $\alpha_0, \alpha_1 \in \mathbb{C}$ represent the complex probability amplitudes, and $|0\rangle$ and $|1\rangle$ denote the computational basis states representing a quantum state. This superposition allows for the simultaneous representation of both $|0\rangle$ and $|1\rangle$ states until measured, following the normalization condition $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

Superposition and entanglement A unique property in quantum mechanics is superposition. Superposition allows a quantum state to exist simultaneously in multiple basis states. In quantum computing, this often involves a qubit being in a combination of $|0\rangle$ and $|1\rangle$ states, but superposition can include many more states in more complex systems. This property allows calculations to be performed on multiple states at the same time. Entanglement is another property of qubits that occurs when two or more particles become correlated in such a way that their states are no longer independent and cannot be described individually. Instead, their states are intrinsically connected, regardless of the spatial separation between them.

Noisy intermediate-scale quantum (NISQ) computers As predicted by Preskill (2018), we have now entered the NISQ era, where quantum hardware has evolved from a few qubits (e.g., the 2019 IBM Falcon) to modular technology boasting 3 x 133 qubits (e.g., the 2024 IBM Heron). However, this generation of hardware is still significantly limited by its size and error rate. It would hence be unrealistic to expect NISQ devices to revolutionize the world on its own; rather, it should be seen as a stepping stone towards developing more advanced quantum technologies in the future. With proper error mitigation and correction techniques, we would be able to run small quantum circuits with desired accuracy for practical applications. *Precisely, in this work, we operate in this regime, where small quantum circuits can have practical applications in QML.*

Appendix B: Datasets and data processing details

In this section, we provide the details of the datasets we use during our experiments, including how we process them.

Botnet DGA dataset This dataset was obtained from Suryotrisongko (2022b). Botnets consist of compromised devices controlled by a central entity known as the botmaster, and

they engage in malicious activities such as launching DDoS[3] attacks or spreading malware (Zargar et al. 2013). The domain generation algorithm (DGA) is a technique employed by botnets to evade detection and maintain communication with their command-and-control (C&C) infrastructure. For our experiments, we extract 1000 equally distributed samples from the benchmarked dataset, obtained from Suryotrisongko (2022a), consisting of 7 features and 1 binary label (*malicious* or *non-malicious*).

Breast Cancer Wisconsin (Diagnostic) dataset A commonly used dataset is the University of California Irvine Machine Learning Breast Cancer Wisconsin (Diagnostic) dataset (Zwitter and Soklic 1988). This dataset consists of 569 instances of breast cancer samples with 30 attributes. We can classify each sample in this dataset as *benign* or *malignant*. Principal component analysis (PCA) is carried out to reduce the number of attributes or features from 30 to only 7 to prevent a high number of trainable parameters.

MNIST dataset This widely used dataset consists of 70,000 28 x 28 grayscale images of handwritten digits (0, …, 9) and their respective labels. They were obtained from Deng (2012) and made available for our use using the `torchvision` library. For our experiments, we utilized 6000 normalized random samples using a set random seed.

Fashion-MNIST dataset This is another image dataset consisting of 70,000 examples of clothing items and 10 labels. Each sample is a 28 x 28 grayscale image associated with a label from the 10 classes. We obtained the Fashion-MNIST (or FMNIST) dataset from Xiao et al. (2017), extracted 6000 random samples using the same set random seed as in the MNIST case, and normalized the images.

Speech Commands dataset This is an audio dataset consisting of 105,829 utterances of 35 commonly spoken words from 2168 speakers (Warden 2018). Each sample is stored as a `.wav` file with a length of a maximum of 1 s. This dataset is obtained from the `torchaudio` library. We extract the utterances corresponding to the words "yes" and "no" and generate their respective spectrograms resized into single channel 28 x 28 images. We transform the spectrograms as such to be able to use a similar HQSL model as variant 2 as described in Section 4.2.2. Hence, in this work, the classification of the Speech Commands dataset is a binary image classification task, where the images are of the same shape as MNIST and FMNIST datasets.

[3] Distributed denial-of-service (DDoS) is an example of unauthorized impairment of electronic communication.

Appendix C: Using a heuristic method to select quantum circuit

In this section, we describe the heuristic approach we have employed to select the quantum circuit that we use to build the quantum layer for its introduction in HQSL. We adopt this approach to find the optimal quantum circuit by making informed decisions and reaching a solution that satisfies some metric. We state the metric as follows: *Which quantum circuit from our proposed set of quantum circuits can give rise to an HQSL model that outperforms SL in terms of test accuracy and F1-score for all datasets?* For the classical counterpart (SL) of each of these circuits, we follow the same rule as introduced earlier, i.e., we use a hidden layer where the number of input nodes is equal to the size of the input data dimension, the number of output nodes is the same as the number of qubits (or measurement outputs) in the circuit, and the `ReLU` activation at the end of the dense layer.

Two important considerations for our set of quantum circuits are the circuit depth and width (number of qubits). We want to have the smallest circuit that satisfies the above-mentioned metric. The compact size of the circuit is advantageous to ensure a reasonable runtime for our simulations using the `default-qubit` simulator by PennyLane. Moreover, this makes it more likely to be implemented on real devices in the near-term. In short, the larger the number of qubits and the deeper the circuit, the longer the runtime of HQSL and the harder it is to implement on actual quantum computers.

First, we investigated 2-dimensional inputs to a quantum circuit with 2 qubits and entanglement, where we uploaded each feature 3 times per qubit. This is circuit 1. In circuit 2, we scaled the number of qubits to 4 to accommodate 4-dimensional inputs and updated the entanglement strategy accordingly. In circuit 3, we kept the 4 qubits layout but adopted a traditional VQC configuration instead with phase-encoding to convert 4-dimensional classical data to quantum states that were operated on by 3 parameterized rotation gates each and entangling gates. We then considered assessing the performance without entangling gates in circuit 4, which was similar to circuit 1 but lacked entanglement. In circuit 5, we increased the depth of the circuit. For all the circuits we described above (1–5) when used in HQSL, we were not able to obtain an accuracy and F1-score that outperformed their equivalent SL model on all datasets. Hence, our set metric was not satisfied for these circuits. We then adapted the work done by Pérez-Salinas et al. (2020) to develop circuit 6 as described in Section 4.2.1. Our experiments showed that circuit 6 satisfies the metric we have set and, hence, concludes this heuristic approach.

Table 4 Table of circuits trialled and brief description of the circuits

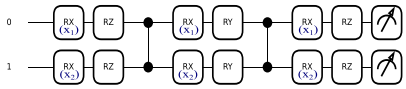
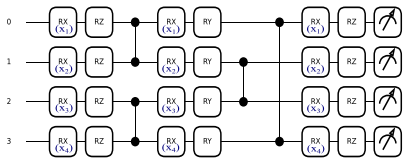
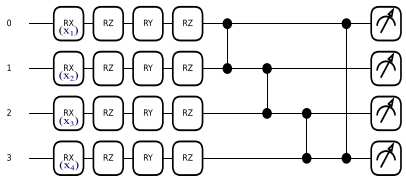
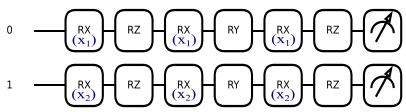
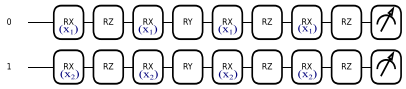
ID	Quantum circuit	Description
1		2-qubit circuit with CZ-entanglement operating on 2 classical input features (x_1, x_2), each uploaded 3 times at the 3 R_x-gates on each qubit
2		4-qubit circuit with CZ-entanglement organized as shown operating on 4 classical input features (x_1, x_2, x_3, x_4), each uploaded at 3 different points at the RX-gates on each of the 4 qubits
3		4-qubit circuit with single-upload of each of the 4 classical input features (x_1, x_2, x_3, x_4) at the encoding layer, followed by the RZ-RY-RZ parameterized gates on each qubit. This is then followed by circular entanglement using CZ-gates. This circuit is an example of a traditional variational quantum circuit
4		2-qubit circuit without entanglement operating on 2 classical input features (x_1, x_2), each uploaded 3 times at the RX-gates on each qubit
5		2-qubit circuit without entanglement operating on 2 classical input features (x_1, x_2), each uploaded 4 times at the RX-gates on each qubit
6		2-qubit circuit with CZ-entanglement operating on 3 classical input features (x_1, x_2, x_3), each uploaded once on each qubit at the RX-gates

Table 5 Summary of accuracy and F1-score for HQSL model using each quantum circuit from Table 4 compared against their equivalent SL models

Datasets			Circuit 1	Circuit 2	Circuit 3	Circuit 4	Circuit 5	Circuit 6
Botnet DGA	Hybrid	Mean Accuracy	0.921	0.922	0.930	0.918	0.920	0.920
		Mean F1-score	0.893	0.895	0.908	0.891	0.890	0.894
	Classical	Mean accuracy	0.928	0.933	0.923	0.928	0.929	0.919
		Mean F1-score	0.904	0.909	0.897	0.904	0.905	0.890
Breast cancer	Hybrid	Mean accuracy	0.958	0.967	0.961	0.970	0.972	0.970
		Mean F1-score	0.944	0.956	0.949	0.960	0.962	0.961
	Classical	Mean accuracy	0.968	0.963	0.968	0.968	0.970	0.965
		Mean F1-score	0.959	0.951	0.958	0.959	0.960	0.953
FMNIST	Hybrid	Mean accuracy	0.830	0.843	0.840	0.835		0.839
		Mean F1-score	0.831	0.837	0.842	0.838		0.840
	Classical	Mean accuracy	0.809	0.838	0.846	0.809		0.805
		Mean F1-score	0.811	0.840	0.846	0.811		0.808
MNIST	Hybrid	Mean accuracy	0.953		0.965	0.952		0.951
		Mean F1-score	0.953		0.965	0.952		0.951
	Classical	Mean accuracy	0.952		0.964	0.952		0.948
		Mean F1-score	0.952		0.964	0.952		0.948
Speech Commands	Hybrid	Mean accuracy						0.960
		Mean F1-score						0.960
	Classical	Mean accuracy						0.949
		Mean F1-score						0.948

Only HQSL with circuit 6 in its quantum layer outperforms SL on all datasets considered

We display all the circuits we have discussed above in Table 4. A brief description of the quantum circuits is also provided.

We train and test HQSL on all datasets and compare the testing accuracy and F1-score against their classical counterparts. We present these results in Table 5. The greyed-out areas demonstrate that these experiments were not carried out as the circuits they corresponded to were already rejected as they failed the set metric.

As shown in Table 5, only circuit 6 gave an outperformance in terms of both mean accuracy and F1-score over the equivalent SL model for all datasets studied. Circuit 6 is the one that makes efficient use of the 2 qubits in the circuit, whereby we load each of the 3 classical input features onto each qubit at 3 different RX-gates. Hence, we chose this qubit-efficient data-loading quantum circuit due to its superior performance compared to its classical analogue, and it is also easy to simulate without significant computational complexity.

Appendix D: Metrics to compare original and reconstructed images

We describe the metrics that we use to compare the difference between the client's original input and reconstructed images.

Metric 1. Cosine distance, $\mathbf{D_c}$ D_c is a metric used to complement cosine similarity. The latter is a measure of similarity between two non-zero vectors and is defined as the cosine of the angle between the vectors. While cosine similarity metric has been used in multiple contexts, such as face verification (Nguyen and Bai 2011) and text analysis (Lahitani et al. 2016), we use cosine distance to indicate the difference between two vectors instead of their similarity. Hence, the larger the cosine distance is, the larger the difference between two images, **A** and **B**.

$$\text{Cosine Similarity, } S_c(\mathbf{A},\mathbf{B}) := \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||\,||\mathbf{B}||}, -1 \le S_c \le 1 \tag{13}$$

$$\text{Cosine Distance, } D_c(\mathbf{A},\mathbf{B}) = 1 - S_c(\mathbf{A},\mathbf{B}), 0 \le D_c \le 2 \tag{14}$$

Metric 2. Mean square error, MSE MSE is a commonly used metric to compare the difference between 2 images. We use it to compare the difference between the reconstructed images and the original images. The greater the MSE is, the greater the difference between the 2 images. For two images **A** and **B** of size M×N, the MSE is calculated as

$$\text{MSE} = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(\mathbf{A}(i,j) - \mathbf{B}(i,j))^2 \tag{15}$$

where $A(i,j)$ and $B(i,j)$ are the pixel values of the original and reconstructed images, respectively, at position (i,j), and M and N are the dimensions of the images.

Metric 3. Structural dissimilarity index, DSSIM DSSIM can be considered as the complement of the structural similarity index, SSIM. The SSIM metric is widely used to quantify the quality of images and measure the similarity between 2 images (Wang 2004). The SSIM index is a real number between -1 and 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect anti-correlation. For our purpose, we treat values less than 0 as showing no similarity with the original image. Hence, we re-define SSIM as $\text{SSIM} = max(\text{SSIM}, 0)$ such that now, $0 < \text{SSIM} < 1$. While we do not delve into the mathematical properties of SSIM, we define DSSIM as another measure of the difference between 2 images.

$$\text{DSSIM}(\mathbf{A},\mathbf{B}) = \frac{(1 - \text{SSIM}(\mathbf{A},\mathbf{B}))}{2}, 0 \le DSSIM \le 0.5 \tag{16}$$

Given our redefinition of SSIM, similar to the previous two metrics, the larger the DSSIM value is, i.e., the closer it is to 0.5, the larger the difference between the reconstructed and original images.

Metric 4. Log spectral distance, LSD LSD is another metric that has been used in Nugraha et al. (2020) that can be used to compare the original and reconstructed spectrograms. LSD has been used in speech recognition comparing the power spectra of two speech signals (Erell and Weintraub 1990). We apply this to the original and reconstructed spectrograms from the Speech Commands dataset. The LSD between 2 images **A** and **B** of size M×N can be computed as

$$\text{LSD} = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[\log_{10}\left(\frac{\mathbf{A}(i,j)}{\mathbf{B}(i,j)}\right)\right]^2} \tag{17}$$

Appendix E: Additional experiments with reconstruction attack models 2 and 3

By carrying out experiments on Reconstruction Model 1, we have seen that when the noise parameters are tuned to $\mu = 2\pi, 4\pi$ and $b = 0.01$, our HQSL model is more

Table 6 Comparison of Mean Cosine Distance, MSE, and MDSSIM values between raw Fashion-MNIST (FMNIST) and MNIST images and reconstructed images for reconstruction models 2 and 3, with noise parameters $\mu = 4\pi, b = 0.01$

	Reconstruction Models	Mean Cosine Distance		MSE		MDSSIM	
		Classical	Hybrid	Classical	Hybrid	Classical	Hybrid
FMNIST	Model 2	1.211 (0.539)	**1.253(0.559)**	1.341(0.253)	**16.972(0.248)**	0.484(0.443)	**0.496(0.455)**
	Model 3	1.000(0.998)	**1.377(0.638)**	0.302(0.302)	**8.475(0.805)**	0.489(0.489)	**0.499(0.499)**
MNIST	Model 2	0.992(0.674)	**1.379(0.675)**	1.003(0.587)	**11.694(0.598)**	0.480(0.457)	**0.494(0.462)**
	Model 3	1.000(1.000)	**1.219(0.814)**	0.699(0.699)	**16.923(0.649)**	0.483(0.474)	**0.499 (0.499)**

The larger the metric (represented by bold entries), the worse the reconstruction performance. The values in the bracket represent the baseline results, corresponding to noise-free reconstruction performance

resilient to reconstruction attack by the adversary's model. In this section, we show that similar resilience is shown against other reconstruction models obtained from the literature. We described these models in Section 6.1.2. We configure our noise layer with the parameters $\mu = 4\pi$ and $b = 0.01$ in testing the reconstruction models against our encoding gate-based defense mechanism. These models form the decoder part of an auto-encoder and were obtained from Li et al. 2018—Reconstruction Model 2 and Balle et al. 2022—Reconstruction Model 3. To demonstrate the reconstruction performance in the hybrid and classical settings, we again use the metrics described in Appendix D. The results are presented in Tables 6—Fashion-MNIST and MNIST reconstructions and 7—Speech Commands spectrogram reconstruction.

In general, the reconstruction performance is worse in the hybrid settings, showing reconstruction becomes harder with the introduction of Laplacian noise with $\mu = 4\pi$ and $b = 0.01$. Significant deviations from the baselines are obtained in most cases. When coupled with the inference performance comparison from Fig. 15, we show a clear advantage of HQSL compared to SL. Specifically, in the presence of our proposed noise defense mechanism based on the rotational properties of the encoding gates in the quantum layer, we can simultaneously impede the reconstruction performance in the hybrid setting and maintain a high classification performance by HQSL. In contrast, SL's performance during classification suffers from the defense method we employ.

Appendix F. Speech Commands dataset result for single client-single server experiments

In this section, we present the raw data collected for the Speech Commands dataset describing the accuracies and F1-scores obtained during the single client-single server experiments. The results presented in Table 8 complement the single client-single server results in Fig. 7 presented in Section 5.1. The data show that the extremely low standard deviations (std) in the accuracies and F1-scores for HQSL demonstrate that our HQSL model's performance shows high consistency, corresponding to high stability on this dataset. These results also explain the narrow box plots obtained for the Speech Commands dataset using our proposed HQSL model, displayed in Fig. 7.

Table 7 Comparison of Mean Cosine Distance, MDSSIM, and Mean Log Spectral Distance (Mean LSD) values between original and reconstructed spectrogram images from Speech Commands dataset for reconstruction models 2 and 3, with noise parameters $\mu = 4\pi, b = 0.01$

	Reconstruction Models	Mean Cosine Distance		MDSSIM		Mean LSD	
		Classical	Hybrid	Classical	Hybrid	Classical	Hybrid
Speech	Model 2	**0.333(0.024)**	0.179(0.014)	**0.479(0.149)**	0.403(0.094)	**5.544(1.820)**	2.932(1.379)
Commands	Model 3	0.791(0.031)	**0.955(0.006)**	0.497(0.190)	**0.499(0.075)**	9.303(2.152)	**9.629(0.597)**

The larger the metric (represented by bold entries), the worse the reconstruction performance. The bracketed values represent the baseline metric values, corresponding to noise-free performance

Table 8 Raw data for Speech Commands dataset for single client-single server experiments complementing Fig. 7

	Fold	F1-score	Accuracy	Loss
Classical (SL)	1	0.941733	0.941766	0.120393
	2	0.951114	0.951158	0.105335
	3	0.942385	0.942392	0.114520
	4	0.951102	0.951158	0.105450
	5	0.956093	0.956168	0.101465
	mean	0.948485	0.948528	–
	std	0.006214	0.006237	–
Hybrid (HQSL)	1	0.957421	0.957420	0.086337
	2	0.959869	0.959925	0.082089
	3	0.961175	0.961177	0.082415
	4	0.959919	0.959925	0.086670
	5	0.959918	0.959925	0.084178
	mean	0.959660	0.959674	–
	std	0.001368	0.001372	–

The low standard deviations (std) relative to the means for HQSL case demonstrate high consistency in the accuracy and F1-score results

Author Contributions All authors contributed to the research and writing. H.C. did the implementation and testing. All authors reviewed and edited the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. No funding was received to assist with the preparation of this manuscript.

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare no competing interests.

References

Abuadbba S, Kim K, Kim M, Thapa C, Camtepe SA, Gao Y, Kim H, Nepal S (2020) Can we use split learning on 1D CNN models for privacy preserving training? In: Proceedings of the 15th ACM asia conference on computer and communications security. ACM, Taipei Taiwan, pp 305–318. https://doi.org/10.1145/3320269.3384740. https://dl.acm.org/doi/10.1145/3320269.3384740

Alam M, Kundu S, Topaloglu RO, Ghosh S (2021) Quantum-classical hybrid machine learning for image classification (iccad special session paper). In: 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE, pp 1–7

Balle B, Cherubin G, Hayes J (2022) Reconstructing training data with informed adversaries. In: 2022 IEEE Symposium on Security and Privacy (SP). IEEE, pp 1138–1156

Deng L (2012) The MNIST database of handwritten digit images for machine learning research [Best of the Web]. IEEE Signal Process Mag 29(6):141–142. https://doi.org/10.1109/MSP.2012.2211477

Dougherty S, Kumar A, Hou J, Tourani R, Tabakhi AM (2023) A stealthy inference attack on split learning with a split-fuse defensive measure. In: 2023 IEEE Conference on Communications and Network Security (CNS). IEEE, Orlando, FL, USA, pp 1–9. https://doi.org/10.1109/CNS59707.2023.10288661. https://ieeexplore.ieee.org/document/10288661/

Erell A, Weintraub M (1990) Estimation using log-spectral-distance criterion for noise-robust speech recognition. In: International conference on acoustics, speech, and signal processing, pp 853–8562. https://doi.org/10.1109/ICASSP.1990.115972

Gawron G, Stubbings P (2022) Feature space hijacking attacks against differentially private split learning. arXiv. arXiv:2201.04018 [cs]. http://arxiv.org/abs/2201.04018

Javadi-Abhari A, Treinish M, Krsulich K, Wood CJ, Lishman J, Gacon J, Martiel S, Nation PD, Bishop LS, Cross AW, Johnson BR, Gambetta JM (2024) Quantum computing with Qiskit. https://doi.org/10.48550/arXiv.2405.08810

Jia Z-A, Yi B, Zhai R, Wu Y-C, Guo G-C, Guo G-P (2019) Quantum neural network states: a brief review of methods and applications. Adv Quantum Technol 2(7–8):1800077

Joshi P, Thapa C, Camtepe S, Hasanuzzaman M, Scully T, Afli H (2022) Performance and information leakage in splitfed learning and multi-head split learning in healthcare data and beyond. Methods Protoc 5(4):60. https://doi.org/10.3390/mps5040060

Lahitani AR, Permanasari AE, Setiawan NA (2016) Cosine similarity to determine similarity measure: study case in online essay assessment. In: 2016 4th International conference on cyber and IT service management, pp 1–6. https://doi.org/10.1109/CITSM.2016.7577578

Li Z, Liu X, Xu N, Du J (2015) Experimental realization of a quantum support vector machine. Phys Rev Lett 114:140504. https://doi.org/10.1103/PhysRevLett.114.140504

Li F, Qiao H, Zhang B (2018) Discriminatively boosted image clustering with fully convolutional auto-encoders. Pattern Recognit 83:161–173. https://doi.org/10.1016/j.patcog.2018.05.019

Li O, Sun J, Yang X, Gao W, Zhang H, Xie J, Smith V, Wang C (2022) Label leakage and protection in two-party split learning

Liang Y, Peng W, Zheng Z-J, Silvén O, Zhao G (2021) A hybrid quantum-classical neural network with deep residual learning. Neural Netw 143:133–147

Liu W, Wu Q, Zha Y (2023) A block-ring connected topology of parameterized quantum circuits

Liu W, Zhang Y, Deng Z, Zhao J, Tong L (2021a) A hybrid quantum-classical conditional generative adversarial network algorithm for human-centered paradigm in cloud. EURASIP J Wirel Commun Netw 2021(1):37. https://doi.org/10.1186/s13638-021-01898-3. Accessed 06 Feb 2025

Liu J, Lim KH, Wood KL, Huang W, Guo C, Huang H-L (2021b) Hybrid quantum-classical convolutional neural networks. Sci China Phys Mech Astron 64(9):290311

Lloyd S, Mohseni M, Rebentrost P (2014) Quantum principal component analysis. Nat Phys 10(9):631–633

McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H (2018) Barren plateaus in quantum neural network training landscapes. Nat Commun 9(1):4812

Mireshghallah F, Taram M, Ramrakhyani P, Jalali A, Tullsen D, Esmaeilzadeh H (2020) Shredder: learning noise distributions to protect inference privacy. In: Proceedings of the twenty-fifth international conference on architectural support for programming languages and operating systems. ACM, Lausanne Switzerland, pp 3–18. https://doi.org/10.1145/3373376.3378522. https://dl.acm.org/doi/10.1145/3373376.3378522

Na H, Oh Y, Lee W, Choi D (2024) Systematic evaluation of robustness against model inversion attacks on split learning. In: Kim H, Youn J (eds) Information security applications. Springer, Singapore, pp 107–118

Nguyen HV, Bai L (2011) Cosine similarity metric learning for face verification. In: Kimmel R, Klette R, Sugimoto A (eds) Computer Vision - ACCV 2010. Springer, Berlin, Heidelberg, pp 709–720

Nielsen MA, Chuang IL (2010) Quantum computation and quantum information: 10th Anniversary Edition. Cambridge University Press, Cambridge, UK. https://doi.org/10.1017/CBO9780511976667

Nugraha AA, Sekiguchi K, Yoshii K (2020) A flow-based deep latent variable model for speech spectrogram modeling and enhancement. IEEE/ACM Trans Audio Speech Lang Process 28:1104–1117. https://doi.org/10.1109/TASLP.2020.2979603

Park S, Baek H, Kim J (2023) Quantum split learning for privacy-preserving information management. In: Proceedings of the 32nd ACM International conference on information and knowledge management. CIKM '23. Association for Computing Machinery, New York, NY, USA, pp 4239–4243. https://doi.org/10.1145/3583780.3615144

Pasquini D, Ateniese G, Bernaschi M (2021) Unleashing the tiger: inference attacks on split learning. In: Proceedings of the 2021 ACM SIGSAC conference on computer and communications security. ACM, Virtual Event Republic of Korea, pp 2113–2129. https://doi.org/10.1145/3460120.3485259. https://dl.acm.org/doi/10.1145/3460120.3485259

Pérez-Salinas A, Cervera-Lierta A, Gil-Fuster E, Latorre JI (2020) Data re-uploading for a universal quantum classifier. Quantum 4:226. https://doi.org/10.22331/q-2020-02-06-226. arXiv:1907.02085 [quant-ph]

Preskill J (2018) Quantum computing in the NISQ era and beyond. Quantum 2:79. https://doi.org/10.22331/q-2018-08-06-79

Sarathy R, Muralidhar K (2011) Evaluating Laplace noise addition to satisfy differential privacy for numeric data. Trans Data Privacy 4(1):1–17

Schetakis N, Aghamalyan D, Griffin P, Boguslavsky M (2022) Review of some existing QML frameworks and novel hybrid classical-quantum neural networks realising binary classification for the noisy datasets. Sci Rep 12(1):11927

Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N (2019) Evaluating analytic gradients on quantum hardware. Phys Rev A 99(3):032331. arXiv:1811.11184 [quant-ph]. https://doi.org/10.1103/PhysRevA.99.032331

Shokri R, Shmatikov V (2015) Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. ACM, Denver Colorado USA, pp 1310–1321. https://doi.org/10.1145/2810103.2813687. https://dl.acm.org/doi/10.1145/2810103.2813687

Shokri R, Stronati M, Song C, Shmatikov V (2017) Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). IEEE, San Jose, CA, USA, pp 3–18. https://doi.org/10.1109/SP.2017.41. http://ieeexplore.ieee.org/document/7958568/

Suryotrisongko H (2022a) Botnet DGA Detection. https://doi.org/10.24433/CO.4005597.v2

Suryotrisongko H (2022b) Botnet DGA Detection. https://ieee-dataport.org/open-access/botnet-dga-dataset

Thapa C, Mahawaga Arachchige PC, Camtepe S, Sun L (2022) SplitFed: when federated learning meets split learning. Proc AAAI Conf Artif Intell 36(8):8485–8493. https://doi.org/10.1609/aaai.v36i8.20825. Accessed 02 Aug 2024

Titcombe T, Hall AJ, Papadopoulos P, Romanini D (2021) Practical defences against model inversion attacks for split neural networks. arXiv. arXiv:2104.05743 [cs]. http://arxiv.org/abs/2104.05743

Tsang SL, West MT, Erfani SM, Usman M (2023) Hybrid quantum-classical generative adversarial network for high resolution image generation. IEEE Trans Quantum Eng 4:1–19. https://doi.org/10.1109/TQE.2023.3319319. arXiv:2212.11614 [quant-ph]. Accessed 07 Feb 2025

Vepakomma P, Gupta O, Swedish T, Raskar R (2018) Split learning for health: distributed deep learning without sharing raw patient data. arXiv. arXiv:1812.00564 [cs, stat]. http://arxiv.org/abs/1812.00564

Vepakomma P, Singh A, Gupta O, Raskar R (2020) NoPeek: information leakage reduction to share activations in distributed deep learning. arXiv. arXiv:2008.09161 [cs, stat]. http://arxiv.org/abs/2008.09161

Wang Z (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612

Warden P (2018) Speech commands: a dataset for limited-vocabulary speech recognition

Wu Q, Liu W, Huang Y, Liu H, Xiao H, Li Z (2024) A degressive quantum convolutional neural network for quantum state classification and code recognition. iScience 27(4):109394. https://doi.org/10.1016/j.isci.2024.109394. Accessed 06 Feb 2025

Xiao H, Rasul K, Vollgraf R (2017) Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms

Zargar ST, Joshi J, Tipper D (2013) A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. IEEE Commun Surv Tutor 15(4):2046–2069. https://doi.org/10.1109/SURV.2013.031413.00127

Zwitter M, Soklic M (1988) Breast cancer. UCI Machine Learning Repository. https://doi.org/10.24432/C51P4M