# A Quantum-Optimized Training Framework for Radio Frequency Fingerprint Identification

To Truong An, *Student Member, IEEE*, Guolin Yin, *Member, IEEE*, Junqing Zhang, *Senior Member, IEEE*, Yuan Ding, *Member, IEEE*, Trung Q. Duong, *IEEE Fellow*, and Simon L. Cotton, *IEEE Fellow*

***Abstract*—Radio frequency fingerprint identification (RFFI) offers a promising physical-layer method to authenticate devices based on unique hardware impairments. However, existing RFFI systems use deep learning (DL) models that are resource-intensive. Training is particularly demanding, requiring repeated updates to a large number of parameters. In this paper, we introduce a quantum-assisted training (QAST) framework to address training inefficiencies in RFFI. QAST integrates a quantum neural network (QNN) with a mapping network to generate parameters for a classical DL model. This indirect training strategy substantially reduces the number of trainable parameters and the overall memory requirements compared to direct training of the DL model. We achieve this by introducing a multimodal mapping network that effectively learns the QNN output. This network generates multiple classical parameters from a shared quantum representation, thereby reducing qubit requirements and lowering the risk of barren-plateau-related trainability degradation. We also propose a new embedding method that reduces the size of the embedding matrix and yields a 15% to 30% reduction in training time. The tailored QAST framework trains the RFFI model while requiring only 10% of the original number of parameters while maintaining comparable accuracy, thereby substantially reducing memory and computational overhead and enabling efficient training or retraining in resource-limited environments.**

***Index Terms*—Device authentication, model compression, quantum machine learning, radio frequency fingerprint identification (RFFI), training optimization.**

This work was supported in part by the Cyber AI Hub Doctoral Training Programme at CSIT, Queen's University Belfast, by the UK Government as part of the New Deal for Northern Ireland, administered through Innovate UK/UKRI. It was also supported by the UK Engineering and Physical Sciences Research Council (EPSRC), through the EPSRC Hub on All Spectrum Connectivity (EP/X040569/1 and EP/Y037197/1). The work of T. Q. Duong was supported in part by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109, in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant Program RGPIN-2025-04941, and in part by the NSERC CREATE program (Grant number 596205-2025). An earlier version of this paper was presented in part at the 2026 IEEE International Conference on Communications Workshops (ICC Workshops) [1].

To Truong An, Guolin Yin, Simon L Cotton are with the Centre for Wireless Innovation (CWI), School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, BT3 9DT, U.K. (e-mail: (ato01, g.yin, simon.cotton)@qub.ac.uk).

Junqing Zhang is with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, U.K. (e-mail: junqing.zhang@liverpool.ac.uk).

Yuan Ding is with the Institute of Sensors, Signals and Systems (ISSS), Heriot-Watt University, Edinburgh, U.K. (e-mail: yuan.ding@hw.ac.uk).

Trung Q. Duong is with the Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1C 5S7, Canada, and also with the Centre for Wireless Innovation (CWI), School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast, BT3 9DT, U.K. (e-mail: tduong@mun.ca).

## I. Introduction

Wireless communications are now integral to every aspect of modern life, and the need to ensure device security and authenticity has intensified. By their nature, wireless devices are vulnerable to a wide range of threats, including denial-of-service attacks, impersonation attacks, man-in-the-middle attacks, and unauthorized access [2]. Wireless networks are generally protected by cryptographic means. However, cryptographic solutions usually involve key management and can require significant computational power, which may not be suitable for edge devices or resource-constrained systems [3].

In this context, radio frequency fingerprint identification (RFFI) offers an alternative, non-cryptographic approach to physical layer-based authentication. The idea behind RFFI is to exploit the unique characteristics of radio frequency (RF) hardware front-end to identify a particular device [4]. Unique signatures are superimposed on transmitted RF signals as a result of subtle manufacturing imperfections found in hardware components such as oscillators, amplifiers, and modulators [5]. These attributes can then be exploited to differentiate among devices, even if they are from the same manufacturing batch [6]. RFFI has the potential to provide a powerful approach to security for various types of wireless technologies, including long-range (LoRa) [6], ZigBee [7], and Wi-Fi [8]. Recently, the robustness of the classification accuracy of RFFI models and their resistance to channel variations have been investigated, with the authors of [9]–[11], reporting high success rates for device recognition. These findings demonstrate that RFFI has significant potential as a device authentication solution.

Despite this promise, the practical deployment of RFFI faces a number of challenges, such as the intensive computation and memory required for training and retraining RFFI models. Modern RFFI systems need sophisticated deep learning (DL) models, such as convolutional neural networks (CNNs), multilayer perceptrons (MLPs), and recurrent neural networks (RNNs) to tackle the high-dimensional, dynamic, and noisy RF signals [12]–[14]. These models typically require millions of parameters [6], imposing substantial computational and memory costs. Moreover, during the training process, these parameters must be continually updated, further increasing the resource burden. This issue is discussed in detail in Section III-B.

To address the model complexity, several compression strategies have been explored. In [15], the authors demonstrated that structured pruning can reduce convolutional layers

by 27.2x, as well as deliver inference speed-ups of 11.5x on field-programmable gate arrays (FPGAs) and 3x on smartphones, while causing only a drop in accuracy of less than 1%. Similarly, the authors of [16] proposed unstructured pruning for an RFFI model, enabling the removal of up to 70% of weights post-training without compromising classification accuracy. However, these methods primarily target the inference phase and assume offline training. The full model must still be stored and trained over several epochs. In addition, prior work [17], [18] applied knowledge distillation (KD) to transfer knowledge from a large pretrained teacher to a compact student. However, KD requires a large pretrained teacher, which must itself first be trained at full size, and therefore leaves the issue of training inefficiency unresolved.

In this context, quantum machine learning (QML) offers much promise. Relying on underlying quantum principles, including superposition, entanglement, and quantum parallelism, QML can deal with high-dimensional and noisy RF signals in an effective way [19]–[22]. Both experimental results and theoretical analyses indicate that QML can enhance accuracy, reduce model complexity, and speed up convergence in certain applications [23]–[25]. However, QML is still in its early stage and faces several key challenges, particularly in encoding large-scale datasets due to qubit coherence limits and circuit complexity [26], [27]. Practical deployment is further hindered by the need for quantum hardware during inference, which is impractical given the current scarcity of scalable quantum computers. An emerging solution to these limitations is QuantumTrain (QT), initially introduced for image classification tasks [28], [29]. The QT framework incorporates a quantum parameter generator (QPG), which combines a quantum neural network (QNN) with a classical mapping network to generate trainable parameters for a DL model. This design bypasses the data encoding bottleneck by processing inputs entirely in the classical domain. The trained model remains purely classical, enabling inference on standard hardware while retaining the benefits of QML-assisted training. This approach improves training efficiency and significantly reduces the number of trainable parameters compared with conventional training methods.

In this paper, we propose a novel quantum-assisted training (QAST) framework for RFFI systems, building upon the QT approach by introducing an embedding strategy and a mapping network. Specifically, we tackle one of the most pressing bottlenecks in QT, which is excessive training time. We do this by introducing a new embedding method that reduces the size of the embedding matrix, thereby decreasing the computational load on the mapping network and reducing the training time of the QAST framework. Additionally, we introduce a multimodal mapping network (MultiMapNet) that efficiently converts quantum outputs into classical parameters through a grouping strategy. This design enhances the utilization of QNN outputs, reduces qubit requirements, mitigates the risk of barren-plateau-related trainability degradation, and enables stable, scalable training. Our main contributions are highlighted as follows:

- We identify and address a significant limitation of many existing RFFI systems that is often overlooked—their high computational and memory demands during training, which makes them difficult to implement, especially in resource-limited scenarios.
- We propose a new embedding strategy for QAST framework that employs sine and cosine functions restricted to one complete cycle to provide quantum state information corresponding to the output probabilities of the QNN. The new embedding function reduces the size of the embedding matrix from $[\mathcal{P} \times (Q+1)]$ to $[\mathcal{P} \times 3]$, independent of the qubit count $Q$, thereby reducing the computational load on the mapping network.
- We introduce a MultiMapNet that generates multiple classical parameters from a shared quantum representation and leverages multimodal learning to capture output probability distributions effectively. Instead of mapping each QNN output to a single parameter, MultiMapNet generates multiple parameters from a single QNN output feature. This design enhances the utilization of QNN outputs. By adjusting the group size and treating the number of qubits as a tunable hyperparameter, the approach reduces qubit overhead and thereby lowers the risk of barren-plateau-related trainability degradation in the QAST framework.
- We conduct extensive experiments to evaluate the performance of our proposed approach by using QAST to train an RFFI model to classify commodity Wi-Fi devices. We experimentally demonstrate that the proposed QAST framework enables the training of DL-based RFFI models with 90% fewer parameters than models trained using classical methods, without compromising accuracy relative to classical methods. The results also show that the proposed embedding method can reduce training time by more than 15% for 5–6 qubits and more than 30% for 7–10 qubits, compared with the existing embedding strategy utilized in the QT framework.

The rest of the paper is organized as follows: Section II introduces some quantum computing preliminaries. Section III presents the RFFI system and defines the research challenges. Section IV details the design of the QAST framework, while Section V analyzes its parameter efficiency. Section VI presents the experimental setup and highlights the key results. Finally, Section VII concludes the paper.

## II. Quantum Computing Preliminaries

### A. Qubits and Quantum States

A qubit is the basic unit of quantum computing. Classical bits exist exclusively in states of 0 or 1.Unlike classical bits, qubits can exist in superpositions of the basis states $|0\rangle$ and $|1\rangle$, which can be represented mathematically as

$$|\psi\rangle = a|0\rangle + b|1\rangle, \tag{1}$$

where $a$ and $b$ are complex amplitudes satisfying the normalization condition $|a|^2 + |b|^2 = 1$. Upon measurement, the qubit collapses to $|0\rangle$ or $|1\rangle$ with probabilities $|a|^2$ and $|b|^2$, respectively [30].

In quantum mechanics, measurement outcomes are inherently probabilistic. The probability of obtaining a specific outcome corresponding to a basis state $|i\rangle$ is given by [31]

$$p_i = |\langle i|\psi\rangle|^2, \tag{2}$$

where $|i\rangle$ is a computational basis state (e.g., $|00\cdots0\rangle$, $|00\cdots1\rangle$) and $|\psi\rangle$ is the quantum state before measurement. The inner product $\langle i|\psi\rangle$ yields the probability amplitude associated with observing the system in state $|i\rangle$. This amplitude is a complex number, so the probability is obtained by taking the squared magnitude (i.e., $|\langle i|\psi\rangle|^2$). The notation $|\cdot\rangle$, called a *ket*, denotes a quantum state as a column vector, while $\langle\cdot|$, called a *bra*, denotes its conjugate transpose, yielding a row vector whose elements are the complex conjugates of the corresponding ket components.

### B. Quantum Gates and Parameterized Quantum Circuits

Quantum gates are the essential components of quantum circuits used to manipulate quantum states. Parameterized quantum gates are a subset of quantum gates that allow fine-tuning of quantum operations through adjustable parameters. These gates are primarily classified into two groups: single-qubit gates and multi-qubit gates. For example, a parameterized single-qubit gate is the $U_3$ gate, which has three real rotation parameters $\theta, \varphi, \lambda$. It is represented as a $2\times2$ unitary matrix, indicating that it acts on a single qubit (since gates acting on $n$ qubits are represented by $2^n \times 2^n$ matrices).

$$U_3(\theta,\varphi,\lambda) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{3}$$

Multi-qubit gates perform operations on two or more qubits and play a crucial role in establishing quantum correlations, especially entanglement among qubits. For example, the controlled-$U_3$ ($CU_3$) gate is employed to create entanglement between qubits. The $CU_3$ gate applies the $U_3$ operation to a target qubit conditioned on the state of the control qubit and is defined as follows [32]

$$CU_3(\theta,\varphi,\lambda)_{q_0,q_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\theta}{2}\right) & 0 & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & 1 & 0 \\ 0 & e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & 0 & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \tag{4}$$

where $q_0$, $q_1$ denote the control and target qubits, respectively.

Parameterized quantum circuits (PQCs) are typically composed of layers of parameterized single-qubit gates and multi-qubit gates. When a PQC is used as a QNN, the parameters of these gates are treated as trainable variables that are iteratively updated through classical optimization. This training process enables the circuit to minimize a task-specific loss, such as classification or regression problems.

## III. Conventional RFFI Approach and Problem Statement

### A. Conventional RFFI System

As shown in Fig. 1, there are $N_c$ devices under test (DUTs) to be classified by a receiver (Rx). A conventional DL-based

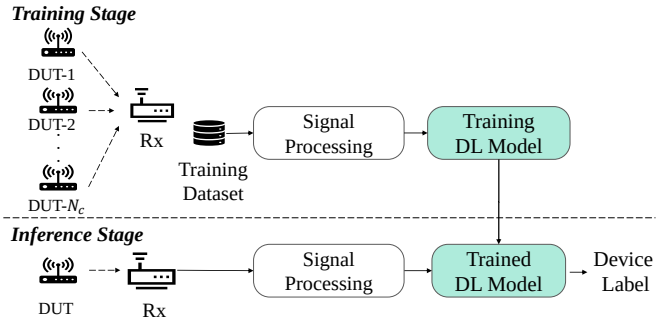


Fig. 1. The overall RFFI system model.

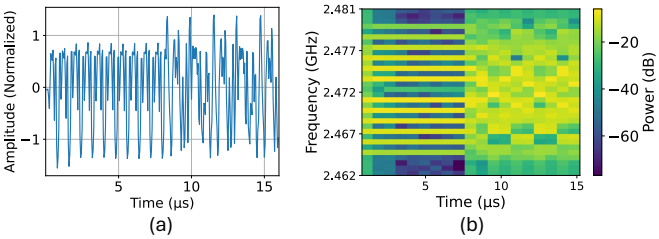


Fig. 2. Preamble part of a Wi-Fi packet. (a) In-phase part of the time-domain signal. (b) The power spectrogram.

RFFI system consists of two stages, i.e., training and inference. In the training stage, labeled packets from various known devices are collected to form a training dataset, which is subsequently used to train the DL model. Once training is complete, the DL model is employed in the inference stage to identify the transmitting device.

*1) Signal Processing:* In both the training and inference stages, the raw received signals undergo signal processing, including carrier frequency offset (CFO) compensation, normalization, and signal representation.

Specifically, for Wi-Fi signals, CFO compensation is first applied to reduce its impact on RFF feature extraction [13]. Next, power normalization is performed to ensure consistency across signals [13]. An example of the normalized signal is shown in Fig. 2(a).

Signal representation in RFFI converts the time-domain in-phase and quadrature (IQ) samples to a proper domain to reveal hardware features. In the literature, various signal representations have been employed for RFFI, including IQ samples, the magnitude of fast Fourier transform (FFT) results, and spectrograms derived from the short-time Fourier transform (STFT) [13]. In this work, we adopt the spectrogram as the signal representation because of its ability to capture both time and frequency domain information.

The normalized signal is transformed into the time–frequency spectrogram by applying a discrete-time STFT, which can be defined as

$$X(k,m) = \sum_{n=0}^{N-1} x[n+mH]\, w[n]\, e^{-j2\pi\frac{k}{N}n}, \tag{5}$$

where $X(k,m)$ denotes an element of the complex STFT matrix, $m = 1,2,\ldots,M$ is the window index, and $k = 0,1,\ldots,N-1$ is the frequency bin index. The function $w[n]$

TABLE I
ARCHITECTURE OF THE CLASSICAL CNN MODEL

| Layer Block | Output Shape | Parameters |
|---|---|---|
| Conv2d(1→8) + BN + MP | $[8, 17, 10]$ | 96 |
| Conv2d(8→16) + BN + MP | $[16, 9, 6]$ | 1,200 |
| Conv2d(16→32) + BN | $[32, 11, 8]$ | 4,704 |
| FC (2816 → 128) | 128 | 360,576 |
| Dropout (0.3) | 128 | 0 |
| FC (128 → 15) | 15 | 1,935 |
| **Total** | — | **368,511** |

represents the windowing function of length $N$ applied to each segment of the signal, and $H$ is the hop size, determining the number of samples by which the window shifts each time. In this study, the window length $N$ is set to 32 samples, and the hop size $H$ is set to 16 samples.

The absolute value of the complex matrix $X$ is computed, which effectively removes the phase information, preserving only the amplitude of $X$. The power spectrogram, computed in the logarithmic domain, is obtained from (5) as

$$X_{\mathrm{dB}} = 10 \log_{10} \left( |X|^2 \right), \tag{6}$$

which is used as the input to the RFFI model. An illustration of this representation is shown in Fig. 2(b).

*2) Deep Learning Model:* Various DL models have been utilized in RFFI systems, such as CNNs, RNNs, and MLPs. In this work, we are inspired by the CNN-based RFFI model proposed in [6] and use it as a case study, as CNNs are well-suited for processing spectrograms. The CNN architecture is shown in Table I. It is built with three sequential 2D convolutional layers, utilizing 3×3 kernel sizes and containing 8, 16, and 32 filters, respectively. Each convolutional stage incorporates batch normalization (BN) and employs the rectified linear unit (ReLU) activation function. To reduce spatial dimensions and extract dominant features, 2×2 max-pooling (MP) layers are inserted after the first and second convolutional layers. A dropout layer is included prior to the fully connected (FC) layers to mitigate overfitting. The final FC layer comprises 128 neurons and employs the Leaky ReLU activation function, accompanied by a dropout rate of 0.3.

### B. Problem Statement

RFFI models require significant computational and memory resources, particularly during the training stage. This issue is further exacerbated in real-world deployments, where RFFI models must be continually fine-tuned on newly collected data to support on-the-fly learning. Continuous training is essential because the hardware characteristics can drift over time due to factors such as aging, necessitating frequent model updates to maintain identification accuracy.

With the rapid expansion of wireless networks, the number of connected devices continues to rise, placing increasing demands on computational and memory resources. In RFFI systems, this trend directly affects the scalability of model training and retraining, as the growing number of devices produces larger and more diverse datasets that must be processed. Accommodating such datasets often necessitates the use of more complex DL models to achieve accurate device identification. However, training these models for RFFI is inherently resource-intensive, particularly in terms of memory consumption, as the total footprint includes memory for storing not only the model parameters, but also the associated gradients, optimizer states, and the intermediate activations that must be retained for backpropagation [33], [34]. As a result, traditional training methods are often impractical in environments with constrained computing and memory resources. This limitation highlights the pressing need for training-efficient models capable of supporting on-the-fly learning in RFFI systems without overburdening the underlying infrastructure.

## IV. QUANTUM-ASSISTED TRAINING FRAMEWORK

### A. Overall Quantum-Assisted Training Framework

Our QAST framework is a hybrid quantum-classical machine learning approach designed to enhance the training efficiency of classical neural networks by leveraging quantum computing. The overall structure of the QAST framework is illustrated in Fig. 3.

At its core, QAST employs a QPG to dynamically produce the parameters (weights and biases) of a target classical model, such as the CNN model commonly used in the RFFI. The QPG comprises three modules: a QNN, an embedding module, and a mapping network. The QNN, implemented as a PQC, generates a measurement probability vector from quantum computations. In the embedding module, the probability vector is augmented to indicate that the probabilities correspond to a unique computational basis state. Finally, the mapping network, a classical neural network, translates these augmented probabilities into the weights and biases of the model. Notably, QAST treats classical DL architectures generically, relying only on the total number of parameters $\mathcal{C}$ rather than their specific structure.

In this framework, the classical CNN neither stores intermediate layer activations nor trains its own weights. Instead, the QPG is responsible for producing the required parameters dynamically for each forward pass. As a result, only the QNN and mapping network are trained, thereby reducing trainable parameters compared to conventional approaches.

### B. Quantum Neural Network

In this work, a QNN utilizes the $U_3$ gate, a single-qubit gate that enables precise manipulation of quantum states [32], and the $CU_3$ gate, which is employed to create entanglement between qubits. The QNN structure is given in Fig. 3(a), which shows the layering of PQC ($U_3$ and $CU_3$ gates) over $Q$ qubits. Notably, the $CU_3$ gate is often designed with a circular layout in quantum circuit diagrams. This layout visually represents the conditional nature of the gate, where the control qubit influences the application of the $U_3$ gate to the target qubit [32]. As the $CU_3$ gate requires both a control and a target qubit, the minimum number of qubits required to construct the QNN is two. The $CU_3$ gate is crucial for constructing multi-qubit interactions within the QNN, enabling the generation of complex quantum states necessary for learning tasks. We

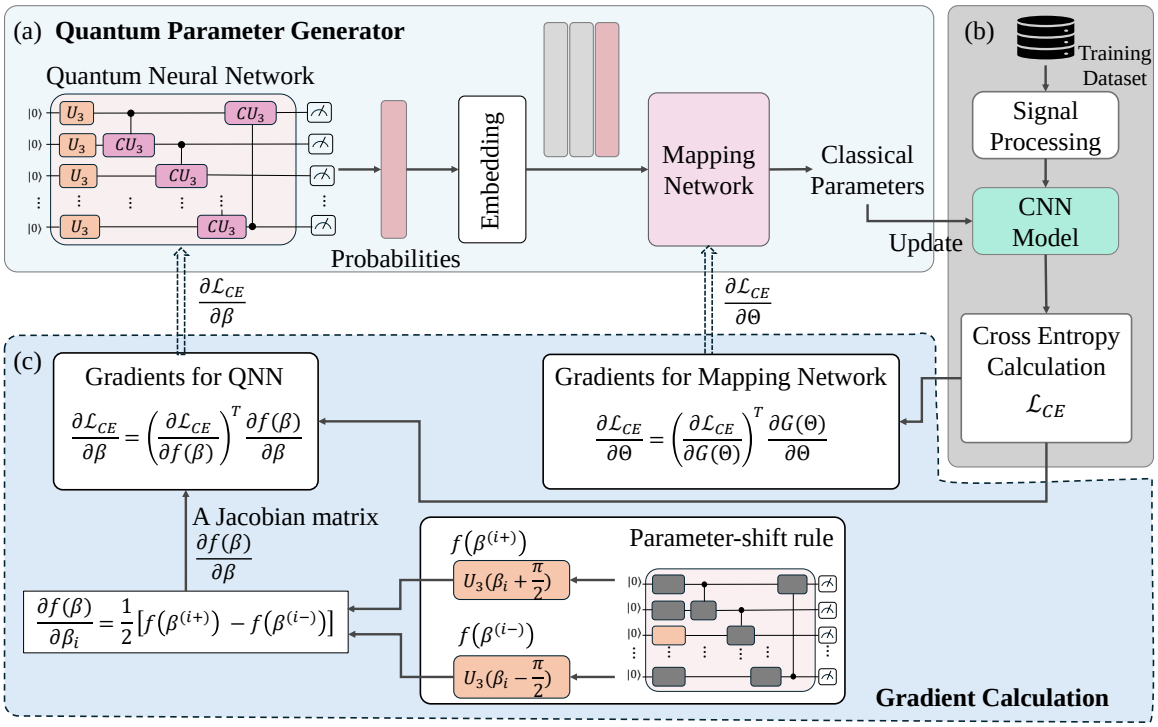


Fig. 3. The proposed quantum-assisted training framework. (a) Quantum parameter generator. (b) Classical CNN based RFFI model. (c) Gradient calculation.

represent the QNN as a function $f(\beta)$, where $\beta$ is the set of trainable parameters. The QNN output quantum state is expressed as

$$|\psi(\beta)\rangle = \left(\prod_i CU_3^{i,i+1} \prod_j U_3^j\right)|0\rangle^{\otimes Q}, \tag{7}$$

where $U_3^j$ denotes a single-qubit $U_3$ rotation applied to qubit $j$, $CU_3^{i,i+1}$ is a controlled-$U_3$ gate with control qubit $i$ and target qubit $i+1$, $\prod_i$ and $\prod_j$ denote ordered products of gates applied sequentially over qubit indices $i = 1, \ldots, Q-1$ and $j = 1, \ldots, Q$, respectively. With $Q$ qubits, the Hilbert space has dimension $2^Q$, hence $2^Q$ possible measurement outcomes.

The measurement outcome probabilities of the QNN output state $|\psi(\beta)\rangle$ in the $\mathcal{P} = 2^Q$ computational basis can be collected into the probability vector

$$\Psi = \begin{bmatrix} |\langle\phi_1|\psi(\beta)\rangle|^2 \\ \vdots \\ |\langle\phi_k|\psi(\beta)\rangle|^2 \\ \vdots \\ |\langle\phi_\mathcal{P}|\psi(\beta)\rangle|^2 \end{bmatrix}, \tag{8}$$

where $|\phi_1\rangle$ denotes the first computational basis state ($|00\cdots0\rangle$), $k \in \{1, 2, \ldots, \mathcal{P}\}$ indexes the computational basis state, and $|\langle\phi_k|\psi(\beta)\rangle|^2$ is the probability of obtaining the outcome $|\phi_k\rangle$ upon measurement.

Each $U_3$ gate and each $CU_3$ gate has three independently trainable rotation parameters, as defined in (3) and (4), respectively. In a single block, one $U_3$ is applied to each of the $Q$ qubits and one $CU_3$ is applied per qubit in the circular entangling layer, yielding six trainable parameters per qubit. The number of trainable QNN parameters is therefore

$$N_\beta = 6Q, \tag{9}$$

for the single-block configuration ($n_{\text{blocks}} = 1$) used in this work. For $L$ blocks, the count generalizes to $N_\beta = 6QL$. While the $U_3$ and $CU_3$ gates share the same parameter count, their parameter values are distinct and independently trainable.

### C. Embedding

The primary objective of the embedding function is to enrich the measurement probabilities by providing information that indicates these probabilities correspond to a unique computational basis state. This enriched representation provides the mapping network with richer information, thereby enabling more effective learning.

*1) Existing Method:* In [28], [32], the authors constructed the embedding matrix by combining the measurement probability $|\langle\phi_k|\psi(\beta)\rangle|^2$ with the corresponding computational basis state $|\phi_k\rangle$. The embedding matrix can be expressed as

$$\mathcal{E}(\Psi) = \begin{bmatrix} |\phi_1\rangle & \| & |\langle\phi_1|\psi(\beta)\rangle|^2 \\ \vdots & \vdots & \vdots \\ |\phi_k\rangle & \| & |\langle\phi_k|\psi(\beta)\rangle|^2 \\ \vdots & \vdots & \vdots \\ |\phi_\mathcal{P}\rangle & \| & |\langle\phi_\mathcal{P}|\psi(\beta)\rangle|^2 \end{bmatrix}, \tag{10}$$

where the $\|$ represents the concatenation operation, and the matrix $\mathcal{E}(\Psi) \in [0,1]^{\mathcal{P}\times(Q+1)}$. For instance, the first row of

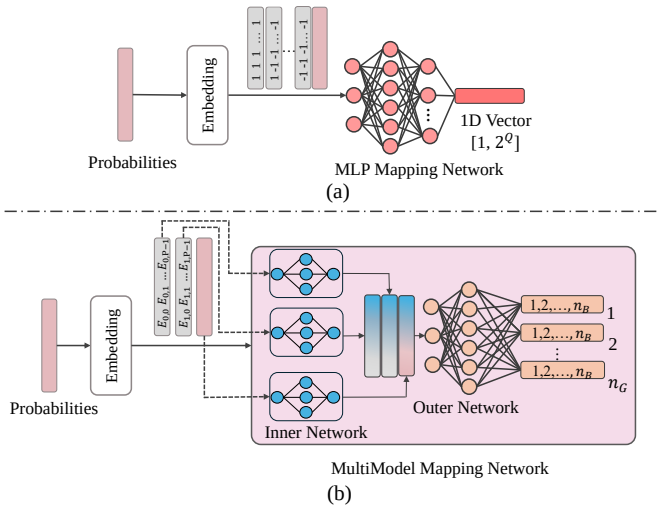


Fig. 4. Overview of the Embedding and Mapping Network in the Quantum-Assisted Training Framework. (a) Existing Architecture. (b) Proposed Architecture.

the embedding matrix $\mathcal{E}(\Psi)$, can be given as

$$\mathcal{E}(\Psi)_1 = |\phi_1\rangle \,\|\, |\langle\phi_1|\psi(\beta)\rangle|^2 \tag{11}$$

$$= [\underbrace{0,\, 0,\, \ldots,\, 0}_{Q \text{ elements}},\, |\langle\phi_1|\psi(\beta)\rangle|^2]. \tag{12}$$

Because classical DL parameters can be both positive and negative, the basis-state matrix is encoded such that each qubit's state is assigned a value of $-1$ for $|0\rangle$ and $+1$ for $|1\rangle$. Thus, the matrix $\mathcal{E}(\Psi) \in [-1, 1]^{\mathcal{P}\times(Q+1)}$. The embedding matrix is illustrated in Fig. 4(a). For example, the computational basis states of a two-qubit system are represented as

$$\Phi = \begin{bmatrix} -1 & -1 \\ -1 & +1 \\ +1 & -1 \\ +1 & +1 \end{bmatrix}. \tag{13}$$

However, the existing method presents a significant scalability challenge when training a very large DL model as the number of qubits $Q$ increases. The embedding matrix grows exponentially with $Q$, which significantly impacts computational efficiency. This not only increases memory usage but also leads to substantially longer processing times.

*2) Proposed Method:* To address these limitations, we propose a new embedding function that is inspired by the positional encoding used in the Transformer model [35], a widely adopted architecture in natural language processing. Rather than employing multiple frequencies, which can introduce unnecessary complexity, our method introduces a compact sinusoidal embedding based on sine and cosine functions within a single period. This approach ensures a unique representation for each computational basis state and generates outputs with both positive and negative values. The proposed embedding matrix can be mathematically described as

$$\mathcal{E}(\Psi) = [E_0, E_1, \Psi], \tag{14}$$

where $E_0$ and $E_1$ are embedding vectors, which provide a compact, unique tag for every measurement probabilities. Therefore, the triplet $[E_0, E_1, \Psi]$ preserves the identity of the corresponding computational basis state associated with each probability. Expanding (14) explicitly

$$\mathcal{E}(\Psi) = \begin{bmatrix} E_{0,0}, & E_{1,0}, & |\langle\phi_1|\psi(\beta)\rangle|^2 \\ \vdots & \vdots & \\ E_{0,k_e}, & E_{1,k_e}, & |\langle\phi_k|\psi(\beta)\rangle|^2 \\ \vdots & \vdots & \\ E_{0,\mathcal{P}-1}, & E_{1,\mathcal{P}-1}, & |\langle\phi_\mathcal{P}|\psi(\beta)\rangle|^2 \end{bmatrix}, \tag{15}$$

where $E_{0,k_e}$ and $E_{1,k_e}$ are the two sinusoidal embedding components, defined as

$$E_{\iota,k_e} = \sin\left(\frac{k_e}{\mathcal{P}}2\pi + \frac{\iota\pi}{2}\right), \tag{16}$$

where $\iota = 0, 1$ is the embedding index, and $k_e$ is the quantum state index with $k_e \in \{0, 1, 2, \ldots, \mathcal{P}-1\}$.

When $\iota = 0$, the embedding is a sine function

$$E_{0,k_e} = \sin\left(\frac{k_e}{\mathcal{P}}2\pi + 0\right) = \sin\left(\frac{k_e}{\mathcal{P}}2\pi\right). \tag{17}$$

When $\iota = 1$, the embedding is

$$E_{1,k_e} = \sin\left(\frac{k_e}{\mathcal{P}}2\pi + \frac{\pi}{2}\right) = \cos\left(\frac{k_e}{\mathcal{P}}2\pi\right). \tag{18}$$

which becomes a cosine function. Thus, the pair of sinusoidal embedding components for a given $k$ is

$$(E_{0,k_e}, E_{1,k_e}) = \left(\sin\left(\frac{k_e}{\mathcal{P}}2\pi\right),\, \cos\left(\frac{k_e}{\mathcal{P}}2\pi\right)\right). \tag{19}$$

Since sine and cosine functions range from $-1$ to 1, the embedding outputs contain positive and negative values.

We now prove that this embedding is unique for each computational basis state.

*Theorem 1 (Uniqueness of the Pair of Sinusoidal Embedding Components):* Let $Q \in \mathbb{N}$ be a positive integer, and let $k_e \in \{0, 1, \ldots, \mathcal{P}-1\}$. Then the pair of sinusoidal embedding components $(E_{0,k_e}, E_{1,k_e})$ is unique for each distinct $k$.

*Proof:* Suppose, for the sake of contradiction, that there exist $k_{e1} \neq k_{e2} \in \{0, 1, \ldots, \mathcal{P}-1\}$ such that

$$(E_{0,k_{e1}}, E_{1,k_{e1}}) = (E_{0,k_{e2}}, E_{1,k_{e2}}). \tag{20}$$

This implies

$$\sin(\xi_{k_{e1}}) = \sin(\xi_{k_{e2}}) \quad \text{and} \quad \cos(\xi_{k_{e1}}) = \cos(\xi_{k_{e2}}), \tag{21}$$

where $\xi_{k_e} = \frac{2\pi k_e}{\mathcal{P}} \in [0, 2\pi)$.

Using the cosine difference identity and the Pythagorean identity,

$$\begin{aligned} \cos(\xi_{k_{e1}} - \xi_{k_{e2}}) &= \cos\xi_{k_{e1}}\cos\xi_{k_{e2}} + \sin\xi_{k_{e1}}\sin\xi_{k_{e2}} \\ &= \cos\xi_{k_{e1}}\cos\xi_{k_{e1}} + \sin\xi_{k_{e1}}\sin\xi_{k_{e1}} \\ &= \cos^2\xi_{k_{e1}} + \sin^2\xi_{k_{e1}} = 1. \end{aligned} \tag{22}$$

Hence $\cos(\xi_{k_{e1}} - \xi_{k_{e2}}) = 1$, which means $\xi_{k_{e1}} - \xi_{k_{e2}} = 2\pi m$ for some $m \in \mathbb{Z}$. Moreover, since $\xi_{k_{e1}}, \xi_{k_{e2}} \in [0, 2\pi)$, so we have $m = 0$ and thus $\xi_{k_{e1}} = \xi_{k_{e2}}$. This contradicts the assumption that $k_{e1} \neq k_{e2}$.

Therefore, no two different angles in $[0, 2\pi)$ have the same sine and cosine values. Hence each $k$ yields a unique pair of sinusoidal embedding components. ■

The new embedding is illustrated in Fig. 4(b). The key benefit of this method is that it significantly reduces the computational load on the mapping network by decreasing the size of the embedding matrix from $\mathcal{E}(\Psi) \in [-1, 1]^{\mathcal{P}\times(Q+1)}$ to $\mathcal{E}(\Psi) \in [-1, 1]^{\mathcal{P}\times 3}$, regardless of the number of qubits $Q$. This substantial reduction in dimensionality eliminates a major bottleneck in existing approaches and results in a much faster and more scalable QAST framework, especially as the number of qubits increases. However, it introduces a minor trade-off in terms of accuracy. This slight reduction in accuracy, compared to the original high-dimensional embedding, will be elaborated in Section VI-C4.

### D. Mapping Network

A crucial step in the QAST framework is to map the embedded probabilities to the trainable parameters of the target DL model. For example, a classical DL-based RFFI model that comprises $\mathcal{C}_{\text{DL}}$ trainable parameters can be represented as a function $Z(\theta)$, with $\overrightarrow{\theta}_{DL} = (\theta_1, \theta_2, \ldots, \theta_{\mathcal{C}_{\text{DL}}})$ denotes the trainable parameter vector. To map the probabilities (obtained from the embedding matrix) to the target parameters, a classical neural network is used as a mapping model, denoted by $G$ with tunable parameters $\Theta$. The classical parameters generated by the QAST framework can be expressed as

$$\overrightarrow{\theta} = G_\Theta(\mathcal{E}(\Psi)). \tag{23}$$

*1) Existing Mapping Network:* In [28], [32], the authors employ an MLP as a mapping network, whose architecture is shown in Fig. 4(a). The number of qubits needed for $\mathcal{C}_{\text{DL}}$ parameters is defined as

$$Q = \lceil \log_2 \mathcal{C}_{\text{DL}} \rceil . \tag{24}$$

One of the limitations of the current MLP-based mapping network is its simultaneous processing of the entire embedding matrix, which can hinder the model's capacity to accurately learn the distribution of quantum measurement probabilities. Moreover, the existing mapping network utilizes a one-to-one mapping scheme, where each individual measurement probability is mapped to a single parameter of the classical DL model

$$G_\Theta^{MLP}(E_{0,k_e}, E_{1,k_e}, |\langle\phi_k|\psi(\beta)\rangle|^2) = \theta_k, \tag{25}$$

where $\theta_k$ denotes the $k$-th DL parameter.

For example, to train our CNN with $\mathcal{C}_{\text{CNN}} = 368{,}511$ parameters, the QAST-MLP framework requires 19 qubits, as determined by (24). While current quantum hardware can, in principle, accommodate 19 qubits, the memory footprint and execution time of quantum algorithms both scale exponentially with the number of qubits, making such an implementation impractical. Additionally, QNNs suffer from the phenomenon of barren plateaus, where the cost-function gradient vanishes exponentially as the network size increases [36], [37], which severely affects the training process.

*2) Proposed Mapping Network:* To address these challenges, we introduce MultiMapNet, a multimodal mapping network designed to improve scalability, reduce quantum resource demands, and enhance the model's ability to learn measurement probability distributions. The key idea is to restructure the mapping network so that multiple classical parameters are generated from a measurement probability, rather than mapping each measurement probability to a single parameter. This grouping strategy allows for more efficient use of quantum information and reduces the total number of qubits required. Let $n_B$ denote the number of classical parameters derived from a single quantum representation. The total number of groups, $n_G = \lceil \mathcal{C}_{\text{DL}}/n_B \rceil$, is then determined accordingly, where each group contains $n_B$ parameters.

The proposed MultiMapNet architecture consists of two main components: an inner network and an outer network, portrayed in Fig. 4(b). The inner network is motivated by the Kolmogorov-Arnold Theorem [38], which indicates that multivariate functions can be represented as sums of univariate functions. Following this principle, we design the inner function $g_{\text{in}}(\cdot)$ by assigning a dedicated inner network to each input variable, which are the embedding components $E_{0,k_e}$, $E_{1,k_e}$, and the probability $|\langle\phi_k|\psi(\beta)\rangle|^2$. Each is processed independently by a dedicated sub-network. The inner function is defined as

$$\begin{aligned} g_{\text{in}}\big(E_{0,k_e}, E_{1,k_e}, |\langle\phi_k|\psi(\beta)\rangle|^2\big) = \mathrm{K}_1(E_{0,k_e}) + \mathrm{K}_2(E_{1,k_e}) \\ + \mathrm{K}_3\big(|\langle\phi_k|\psi(\beta)\rangle|^2\big), \end{aligned} \tag{26}$$

where each $\mathrm{K}_i$ is a learned univariate function assigned for each variable. This independent processing disentangles the contributions of each variable, enabling more accurate modeling of QNN measurement probabilities.

The outer network is an MLP model $g_{\text{out}}(\cdot)$ that transforms the aggregated output of the inner network into a matrix of classical parameters with shape $[n_G, n_B]$, where each row corresponds to one group of parameters. This matrix is flattened into a vector of length $n_G \times n_B$. This final output serves as the raw parameter vector $\overrightarrow{\theta}$ for the classical DL.

$$G_\Theta^{MultiMapNet}(\mathcal{E}(\Psi)) = g_{\text{out}}(g_{\text{in}}(\mathcal{E}(\Psi))) = \overrightarrow{\theta}. \tag{27}$$

While MultiMapNet increases complexity, it effectively addresses the inefficiencies of one-to-one mapping in the QAST framework. By restructuring the parameter mapping process, the required number of qubits is reduced from $Q = \lceil \log_2 \mathcal{C}_{\text{DL}} \rceil$ to $Q = \lceil \log_2 n_G \rceil$. This design enables active control over the number of qubits by adjusting the number of classical parameters derived from a single quantum representation. Such control is crucial in the QAST framework because it helps reduce qubit overhead, thereby lowering the risk of barren-plateau-related trainability degradation, as discussed further in Section VI-C3. In addition, MultiMapNet enhances the capability of the model to learn quantum measurement distributions by independently processing each variable, thereby isolating its influence.

**Algorithm 1** QAST Framework

**Input:** Dataset $\mathcal{D}$; classical CNN $Z(\theta)$; QNN $f(\beta)$; mapping network $G_{\Theta}^{MultiMapNet}$; total parameter count $\mathcal{C}_{\text{CNN}}$; target number of qubits $Q$
**Output:** Classical parameters $\overrightarrow{\theta}$

$n_G \leftarrow 2^Q$
$n_B \leftarrow \left\lceil \frac{\mathcal{C}_{\text{CNN}}}{2^Q} \right\rceil$
**for** each epoch **do**
  **for** each mini-batch $(x, y) \in \mathcal{D}$ **do**
    Measure probabilities $\Psi$ from QNN $f(\beta)$
    $\mathcal{E}(\Psi) \leftarrow [E_0, E_1, \Psi]$
    $\overrightarrow{\theta} \leftarrow G_{\Theta}^{MultiMapNet}(\mathcal{E}(\Psi))$
    $\hat{y} \leftarrow Z(x; \overrightarrow{\theta})$
    Calculate $\mathcal{L}_{CE}$
    Update $\Theta$ and $\beta$ via backpropagation
  **end for**
**end for**
**return** $\overrightarrow{\theta}$

### *E. Training Flow for QAST Framework*

The training flow is outlined in Algorithm 1. It begins with initialization: the QNN is configured, qubits are selected, and the mapping network is set up. During the forward pass, the generated parameters are injected into the classical CNN model to produce predictions and compute the task loss. In the backward pass, gradients are computed and backpropagated to update the mapping network, QNN parameters, thereby refining the parameter generator and improving overall performance.

*1) Initialization and Qubit Selection:* To initialize the framework, we determine the desired number of qubits, which are related to the target level of parameter efficiency. Details on parameter efficiency are provided in Section V.

Using the number of qubits and the total number of parameters in the classical CNN model, we then construct the QNN and design the mapping network by specifying the parameters $n_B$ and $n_G$, which are given by

$$n_G = 2^Q, \quad n_B = \left\lceil \frac{\mathcal{C}_{\text{CNN}}}{2^Q} \right\rceil. \tag{28}$$

*2) Forward Pass and Parameter Update:* At each training iteration, the QPG generates the full set of CNN parameters. We then replace the CNN's parameters with $\overrightarrow{\theta}$ produced by the QPG. To do so, we use PyTorch's functional operators (e.g., `F.conv2d`, `F.linear`), which accept `weight` and `bias` as arguments and thus allow on-the-fly parameter injection into the corresponding layers, as detailed in Fig. 3(b). After the parameter update, the CNN processes an input $x$ from the training set to produce the prediction $\hat{y}$

$$\hat{y} = Z(x; \overrightarrow{\theta}). \tag{29}$$

For the multi-class classification task, the cross-entropy (CE) loss measures the discrepancy between predicted probabilities $\hat{y}$ and ground-truth labels $y$

$$\mathcal{L}_{CE} = -\frac{1}{N_{\text{Batch}}} \sum_{i=1}^{N_{\text{Batch}}} \sum_{c=1}^{N_c} y_{i,c} \log(\hat{y}_{i,c}), \tag{30}$$

where $N_{\text{Batch}}$ is the batch size.

*3) Backward Pass and Optimization:* After the forward pass, we obtain the CE loss from (30). Gradient calculation is applied to obtain the gradients for the mapping network and the QNN, as detailed in Fig. 3(c). Gradients for the classical mapping network parameters are given by

$$\frac{\partial \mathcal{L}_{CE}}{\partial \Theta} = \left( \frac{\partial \mathcal{L}_{CE}}{\partial G(\Theta)} \right)^{\top} \frac{\partial G(\Theta)}{\partial \Theta}, \tag{31}$$

where $\partial \mathcal{L}_{CE} / \partial G(\Theta)$ is obtained by applying the chain rule through the CNN and the mapping network $G(\Theta)$, while $\partial G(\Theta)/\partial \Theta$ denotes the Jacobian of $G(\Theta)$ with respect to $\Theta$.

For the quantum parameters $\beta = [\beta_1, \beta_2, \ldots, \beta_{N_\beta}]$ in the QNN, gradients are obtained using the parameter-shift rule [39], [40], given by

$$\frac{\partial f(\beta)}{\partial \beta_i} = \frac{1}{2} \left[ f\left(\boldsymbol{\beta}^{(i+)}\right) - f\left(\boldsymbol{\beta}^{(i-)}\right) \right], \tag{32}$$

where $\beta^{(i\pm)} = [\beta_1, \ldots, \beta_i \pm \frac{\pi}{2}, \ldots, \beta_n]$ denotes the parameter vector with only $\beta_i$ shifted by $\pm\frac{\pi}{2}$ while others unchanged.

The gradient of the CE loss with respect to $\beta$ is then computed using the chain rule

$$\frac{\partial \mathcal{L}_{CE}}{\partial \beta} = \left( \frac{\partial \mathcal{L}_{CE}}{\partial f(\beta)} \right)^{\top} \frac{\partial f(\beta)}{\partial \beta}, \tag{33}$$

where $\partial \mathcal{L}_{CE} / \partial f(\beta)$ is obtained by applying the chain rule through the CNN and the QNN and $\partial f(\beta)/\partial \beta$ is the Jacobian matrix, obtained using (32).

These gradients are then backpropagated through the QAST framework, updating only the QNN and the mapping network within the QPG, while the classical CNN model remains non-trainable.

## V. Parameter Efficiency Analysis

As presented in Table II, the MultiMapNet model comprises a total of $370 + 6 \times n_B$ parameters. Therefore, the total number of parameters of the QAST framework is the sum of the parameters of the QNN and the MultiMapNet. The total number of parameters in the QAST framework is given by

$$\mathcal{C}_{\text{QAST}} = 6Q + (370 + 6 \times n_B). \tag{34}$$

Substituting $n_B = \lceil \mathcal{C}_{\text{CNN}} / 2^Q \rceil$ into the above equation yields

$$\mathcal{C}_{\text{QAST}} = 6Q + \left\lceil \frac{6\mathcal{C}_{\text{CNN}}}{2^Q} \right\rceil + 370. \tag{35}$$

The percentage improvement in parameter efficiency achieved by the QAST framework relative to the classical method is $\Delta\mathcal{C}\ (\%) = (1 - \mathcal{C}_{\text{QAST}}/\mathcal{C}_{\text{CNN}}) \times 100$. Based on (35), we can obtain the parameter efficiency as follows

$$\Delta\mathcal{C}(\%) = \left( 1 - \frac{6Q}{\mathcal{C}_{\text{CNN}}} - \frac{6}{2^Q} - \frac{370}{\mathcal{C}_{\text{CNN}}} \right) \times 100. \tag{36}$$

In the QAST framework, where $Q$ is significantly smaller than $\mathcal{C}_{\text{CNN}}$, both terms $6Q/\mathcal{C}_{\text{CNN}}$ and $370/\mathcal{C}_{\text{CNN}}$ become negligible. Therefore, the parameter efficiency can be approximated as

$$\Delta\mathcal{C}_{\text{approx}}(\%) \approx \left( 1 - \frac{6}{2^Q} \right) \times 100. \tag{37}$$

TABLE II
DETAILED ARCHITECTURE OF THE *MULTIMAPNET* MODEL

| Component | Layer Type | Output Shape | Parameters |
|---|---|---|---|
| **Inner Function**† | FC ($1 \rightarrow 6$) + BN | 6 | 24 |
| | FC ($6 \rightarrow 1$) | 1 | 7 |
| **Outer Function** | FC ($3 \rightarrow 8$) + BN | 8 | 48 |
| | FC ($8 \rightarrow 16$) | 16 | 144 |
| | FC ($16 \rightarrow 5$) | 5 | 85 |
| | FC ($5 \rightarrow n_B$) | $n_B$ | $6 \times n_B$ |
| **Total** | - | - | $370 + 6 \times n_B$ |

†*The inner function is applied independently to each of the 3 input variables (repeated 3 times)*

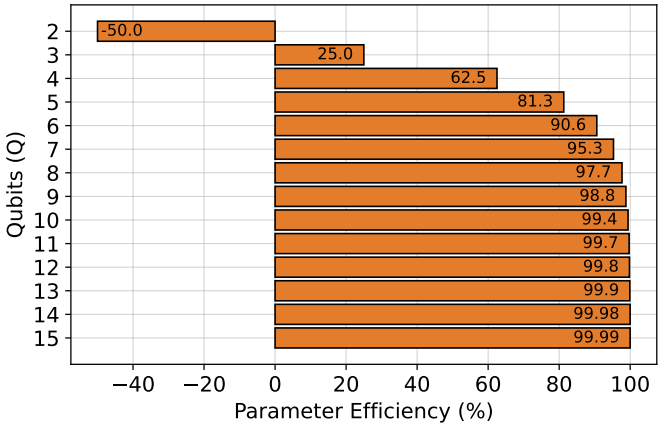


Fig. 5. Approximate parameter efficiency for qubit counts $Q$= 2 to 15.

This approximation reveals that the parameter efficiency improves exponentially with the number of qubits $Q$, demonstrating the efficiency of the QAST framework in terms of model compression. We evaluate the percentage reduction for $Q$ from 2 to 15, setting the minimum at $Q = 2$ because a $CU_3$ gate requires both a control and a target qubit; therefore, at least two qubits are needed to construct the QNN.

As illustrated in Fig. 5, the QAST framework starts to exhibit parameter efficiency over the baseline model when $Q \geq 3$. The parameter efficiency increases substantially with additional qubits, rising from 25% at $Q = 3$ to 62.5% at $Q = 4$, 81.3% at $Q = 5$, and reaching 90.6% at $Q = 6$. These results highlight the effectiveness of QAST, as more than 90% parameter efficiency can be achieved with only 6 qubits. However, beyond $Q = 10$ (where efficiency reaches 99.4%), further improvements become negligible, with only a 0.59% gain observed between $Q = 10$ and $Q = 15$.

## VI. EXPERIMENTAL EVALUATION

This section evaluates the QAST framework's ability to train RFFI models with significantly fewer parameters while maintaining performance comparable to that of conventional training. We conduct experiments using quantum circuit simulations implemented with PyTorch and TorchQuantum [41].

We assume an ideal, noise-free quantum system for precisely measuring quantum state amplitudes during the training of the QAST framework. The reported training time corresponds to ideal quantum-circuit simulation and therefore does not reflect the end-to-end execution latency that would be incurred on real quantum hardware. While current quantum hardware often yields lower accuracy for QNNs compared to noise-free simulations due to noise and limited qubit coherence [36], it is anticipated that rapid advancements in this area suggest that scalable quantum computing may become feasible within the next decade [42], [43]. Moreover, recent studies [39], [44], [45] have shown that QNNs can achieve performance in noise-free simulations comparable to that expected on real quantum devices. Notwithstanding, we emphasize that our focus in this study is to develop a QAST-based algorithm for RFFI and assess its performance compared to classical approaches.

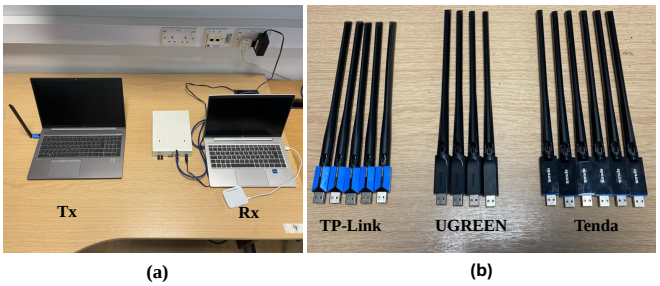


Fig. 6. The experiment setup and devices. (a) Transceiver setup. (b) Wi-Fi dongles.

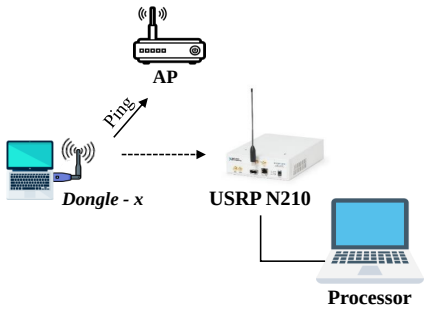


Fig. 7. The data collection setup.

### A. Experimental Setup

*1) Hardware and Collection Tools:* To assess the performance of the proposed QAST framework for the purpose of RFFI, we consider Wi-Fi device identification. In this work, as illustrated in Fig. 6, we utilized a USRP N210 software-defined radio (SDR) as the receiver and employed 15 Wi-Fi dongles as the DUTs. The Wi-Fi dongles were manufactured by three different vendors (TP-Link, UGREEN, and Tenda), and all use the RealTek RTL8821CU, which is a highly integrated single-chip solution. For data collection, the DUTs were connected to a Linux laptop, which sent ping packets to a Wi-Fi access point (AP) to generate a continuous packet stream. The antennas on the DUTs and receiver were verticaly polarized. On the receiver side, the USRP N210 was linked to another Linux laptop running PicoScenes [46] to decode the Wi-Fi packets and store signals to the dataset. A diagram of the data collection setup is shown in Fig. 7.

*2) Data Collection Scenarios:* To ensure that the reduction in trainable parameters facilitated by the QAST framework does not compromise the model's ability to generalize across physical layer variations, we conducted data collection for three commonly encountered communication scenarios to capture RFF characteristics under varying channel conditions: 1) static line-of-sight (LOS); 2) static non-line-of-sight (NLOS); and 3) mobile. The details of each scenario are as follows.

1) *Static LOS*: The DUTs and the USRP N210 receiver were placed on the same table in an office, 5 m apart, with a clear line of sight between transmitter and receiver. This setup provides baseline RFF characteristics with minimal channel distortion.
2) *Static NLOS*: The DUTs were placed inside an office while the USRP N210 receiver was positioned on a table in the corridor, approximately 3 m away, with a concrete wall separating them, simulating realistic indoor environments where obstacles are present.
3) *Mobile*: To investigate the impact of mobility on RFF signatures, the USRP N210 receiver was kept stationary while a person carried the DUTs and walked through the environment (both LOS and NLOS) at approximately 1 m/s during transmission.

We collected 4,500 Wi-Fi packets from each DUT for training (1,500 packets per channel condition), and an additional 1,500 packets per DUT for testing on a separate day (500 packets per scenario). All DUTs operated on channel 13 at 2.472 GHz with a 20 MHz bandwidth, transmitting at 100 packets per second.

*3) Training Hyperparameter Configuration:* The QAST framework and classical RFFI models were trained on a Dell Precision 3680 workstation (Intel® Core™ i7-14700 processor, 64 GB RAM, and 2 TB SSD), using the Adam optimizer, with an initial learning rate of 0.001 and a batch size of 256. A maximum of 500 training epochs was allowed. Early stopping was applied, stopping the training process if the validation loss did not improve for 30 consecutive epochs.

### B. Evaluation Metrics

*1) Accuracy:* Accuracy is a fundamental metric for evaluating the classification performance of the DL-based RFFI model. It measures the proportion of correctly classified instances relative to the total number of instances in the test set. The formula for accuracy is

$$Acc = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(y_i = \hat{y}_i)\,, \tag{38}$$

where $N$ is the total number of the test dataset, $y_i$ is the true label for the $i$-th sample, $\hat{y}_i$ is the predicted label, and $\mathbb{I}(y_i = \hat{y}_i)$ equals 1 if the prediction is correct and 0 otherwise.

To compare QAST with the classical approach, we define the accuracy loss

$$Acc_{Loss} = Acc_{baseline} - Acc_{QAST}, \tag{39}$$

where $Acc_{QAST}$, $Acc_{baseline}$ denote the accuracies of the DL-based RFFI model trained using the QAST framework and using the traditional approach, respectively.

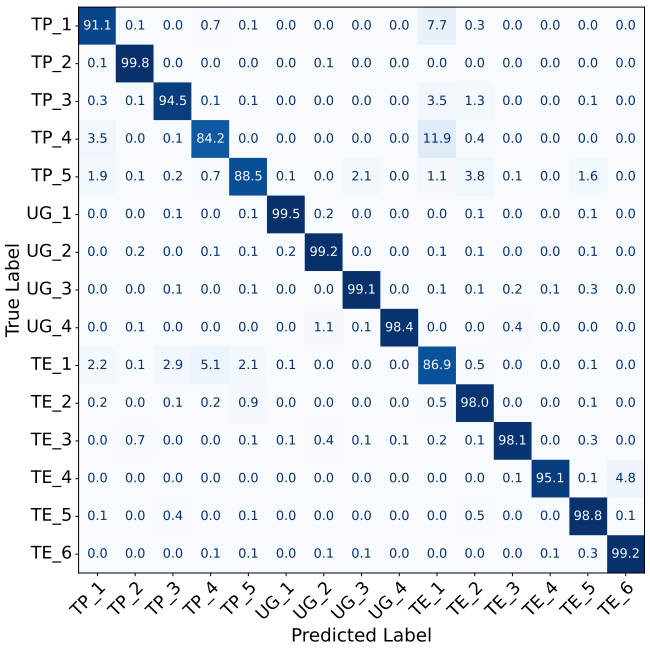


Fig. 8. The confusion matrix of the RFFI model.

*2) Training Time:* Computational efficiency is evaluated using the average training time per epoch, $T_e$ (seconds)

$$T_e = \frac{\text{Total training time}}{\text{Number of epochs}}, \tag{40}$$

where the total training time is measured until model convergence. The relative reduction in training time compared to the baseline is defined as

$$\Delta T_e(\%) = \frac{T_e^{baseline} - T_e^{\text{QAST}}}{T_e^{baseline}} \times 100. \tag{41}$$

To assess the processing efficiency of the proposed embedding, we compute the percentage reduction in training time per epoch relative to the baseline

$$\text{Time Reduction}(\%) = \frac{T_{e,\,E1} - T_{e,\,E2}}{T_{e,\,E1}} \times 100, \tag{42}$$

where $T_{e,\,E1}$ and $T_{e,\,E2}$ denote the training times per epoch for the baseline and proposed embeddings, respectively.

### C. Experimental Results

In this section, we present the average classification accuracy and standard deviation from ten independent consecutive runs to ensure statistical reliability and reproducibility. These metrics reflect the central tendency and variability of the model's performance.

*1) Performance of the RFFI Model and QAST Framework:* A CNN-based RFFI model trained in a classical manner achieves a high level of classification accuracy, reaching a test accuracy of 95.8%. Fig. 8 shows the confusion matrix for the test dataset.

Additionally, we evaluate the CNN-based RFFI model performance when trained using both the traditional method and the QAST framework. In this work, we aim to improve the

TABLE III
COMPARISON OF QAST MODELS OF DIFFERENT $Q$ WITH THE CLASSICAL CNN

| Model | $\Delta\mathcal{C}$ (%) | $Acc$ (%) | $Acc_{Loss}$ (%) | $T_e$ | $\Delta T_e$ (%) |
|---|---|---|---|---|---|
| CNN | – | 95.8 ± 0.3 | – | 1.28 | – |
| QAST-3 | 24.89 | 94.9 ± 1.1 | 0.9 | 1.21 | 5.47 |
| QAST-4 | 62.39 | 95.3 ± 0.5 | 0.5 | 1.30 | −1.56 |
| QAST-5 | 81.14 | 95.5 ± 0.6 | 0.3 | 1.34 | −4.69 |
| QAST-6 | 90.51 | 95.4 ± 0.4 | 0.4 | 1.40 | −9.38 |
| QAST-7 | 95.20 | 94.9 ± 0.6 | 0.9 | 1.70 | −32.81 |
| QAST-8 | 97.54 | 93.5 ± 0.7 | 2.3 | 1.73 | −35.16 |
| QAST-9 | 98.71 | 91.0 ± 1.7 | 4.8 | 1.71 | −33.59 |
| QAST-10 | 99.30 | 85.6 ± 2.8 | 10.2 | 1.95 | −52.34 |

efficiency of the training stage in the RFFI system. Guided by the parameter efficiency bounds established in Section V, we evaluate the QAST framework's performance by varying the number of qubits from 3 to 10 (where efficiency gains plateau), referred to as QAST-3 through QAST-10. The results are shown in Table III. The parameter efficiency achieved by the QAST framework closely aligns with the approximated values calculated using (37). For instance, the parameter efficiency of QAST-3 is 24.89%, while the estimated value based on (37) is 25%. It can be observed that the parameter efficiency increases monotonically with the number of qubits, from 24.89% (QAST-3) to 99.3% (QAST-10).

As shown in Table III, the classification accuracy of the QAST model exhibits a distinct unimodal relationship with qubit scaling, peaking at $Q = 5$ before significant degradation. Accuracy rises from 94.9% (±1.1%) in QAST-3 to a peak of 95.5% (±0.6%) in QAST-5, indicating that adding qubits up to this point enhances the QAST ability to train the classical CNN effectively. However, beyond QAST-5, accuracy declines, reaching 85.6% (±2.8%) in QAST-10. This decline in accuracy is mainly due to the mapping network becoming very small when the parameter efficiency exceeds 95%, leading to underfitting. Additionally, the barren plateau issue, where gradients vanish exponentially with increased qubit count, may further impede effective training.

Regarding training efficiency, most of the QAST models require longer training times per epoch compared to the classical CNN baseline, except for QAST-3. The training times per epoch for QAST-4 to QAST-6 are approximately 1.30 to 1.40 seconds, which is 1.5%–10% higher than those of the conventional training method. As the number of qubits increases, the training time per epoch increases from 1.21 seconds (QAST-3) to 1.95 seconds (QAST-10). The longest duration observed is for QAST-10, which takes 1.95 seconds per epoch, $52\%$ longer than that of the CNN baseline. The training-time cost arises because the current QAST framework is implemented on classical computers by leveraging the open-source TorchQuantum library to run quantum-circuit simulations, which require significant computation, hence impacting the response time. This pattern suggests an optimal qubit count of 6, at which the QAST framework trains the CNN using only 9.49% of the original parameters while achieving classification accuracy close to the classical CNN baseline.

It is worth clarifying the sense in which QAST improves training efficiency. Because the per-epoch cost above includes the TorchQuantum simulation, we make no claim of a quantum speedup: for the CNN, QAST-6 is in fact about 9% slower per epoch than the classical baseline, and its benefit lies in the substantial reduction in trainable parameters rather than in training time. The training-time advantage instead becomes apparent for larger models, as shown later for the MLP-based RFFI model in Section VI-C5, where the classical baseline must update 4.47M weights directly while QAST generates them from a compact mapping network, reducing per-epoch time by approximately 64%–76% even with the simulation included. Real quantum hardware is accordingly regarded as a potential future improvement to the framework rather than a requirement for its operation.

TABLE IV
CLASSIFICATION ACCURACY OF QAST-6 AND THE CLASSICAL CNN UNDER DIFFERENT CHANNEL SCENARIOS.

| Method | Static LOS | Static NLOS | Mobile | Overall |
|---|---|---|---|---|
| CNN | $97.7\% \pm 0.3$ | $92.9\% \pm 0.8$ | $96.9\% \pm 0.1$ | $95.8\% \pm 0.3$ |
| QAST-6 | $97.4\% \pm 0.5$ | $92.4\% \pm 1.0$ | $96.4\% \pm 0.4$ | $95.4\% \pm 0.4$ |

To assess the behavior of QAST under varying environmental conditions, we use QAST-6, which the preceding analysis identified as the optimal configuration. Table IV reports the per-condition classification accuracy of the classical baseline and QAST-6 across the Static LOS, Static NLOS, and mobile test sets, all collected on a separate day from training. QAST-6 closely matches the classical baseline across all three conditions, indicating that the substantial reduction in trainable parameters achieved by QAST does not degrade the behavior of the model under these cross-day and cross-channel shifts relative to conventional training.

*2) Comparison with a Classical Hypernetwork:* To determine whether the parameter-efficiency gains of QAST stem from the quantum-assisted design or simply from parameter reduction, we compare QAST-6 with the static hypernetwork [47], [48]. This baseline is chosen because it is a well-established hypernetwork formulation, it was originally designed to generate the convolutional kernels of CNNs and therefore matches our CNN-based RFFI model. The hypernetwork is configured to generate all parameters of the CNN-based RFFI model, with its embedding size tuned so that its trainable parameter count (32,960) matches that of QAST-6 (34,954). As shown in Table V, the hypernetwork method achieves an average classification accuracy of 95.3% compared to 95.4% for QAST-6, indicating that both methods achieve comparable classification accuracy. However, there is a clear difference in terms of training costs, with QAST-6 training 4.6 times faster than hypernetwork. This speed advantage stems from the design of QAST, where the QNN produces all CNN parameters from a single shared quantum state in one forward pass, whereas the hypernetwork must invoke its generator once per kernel slice. With $Q$ qubits, the quantum state yields a $2^Q$-component measurement distribution from only $6Q$ trainable parameters, and this compact representation allows QAST to generate all CNN parameters in a single pass, which underlies its shorter training time relative to the hypernetwork.

*3) Effect of the Proposed MultiMapNet:* In this experiment, we compare the proposed MultiMapNet with MLP-based

TABLE V
COMPARISON OF QAST-6 WITH STATIC CLASSICAL HYPERNETWORK AND THE CLASSICAL CNN BASELINE.

| Method | Params | Acc. (%) | $T_e$ |
|---|---|---|---|
| CNN | 368,511 | $95.8 \pm 0.3$ | 1.28 |
| Hypernetwork | 32,960 | $95.3 \pm 0.6$ | 6.40 |
| QAST-6 (proposed) | 34,954 | $95.4 \pm 0.4$ | 1.40 |

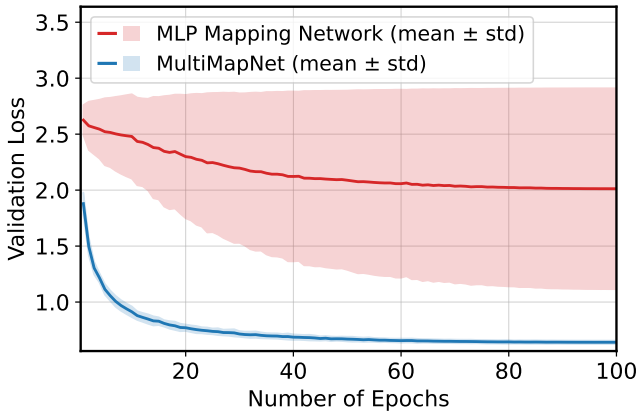


Fig. 9. Validation loss (mean and standard deviation across 10 runs) of QAST models with the MLP mapping network and MultiMapNet.

mapping networks to evaluate its performance. To make a fair comparison, both QAST models leverage the same embedding method as proposed in [28], [32]. It should be emphasized that there is no difference between the two models except for the mapping network structure. This enables us to separate the effect of the mapping network on the overall performance of the QAST framework.

Fig. 9 compares the convergence behavior of QAST when using an MLP-based mapping network and the proposed MultiMapNet. The QAST-MLP framework exhibits unstable training, as indicated by the large variance across runs. In particular, 6 out of 10 runs failed to converge, with the validation loss remaining flat throughout training. Training a CNN-based RFFI model with 368,511 parameters under the QAST-MLP framework requires 19 qubits. This unstable behavior at high qubit count is consistent with barren-plateau-related trainability degradation, in which gradient magnitudes vanish exponentially with the number of qubits [49]. In contrast, MultiMapNet enables proactive control of the qubit count as a hyperparameter, allowing the framework to operate at only 6 qubits and thereby avoid the high-qubit one-to-one regime. As shown in Fig. 9, the QAST using MultiMapNet achieved smooth and consistent convergence with minimal variance across runs.

*4) Effect of the Proposed Embedding:* One of the most important advantages of the proposed embedding is that it can dramatically reduce the computational load of the mapping network. The embedding speeds up the processing time of the mapping network by reducing the complexity of the embedding matrix, which is the input of the mapping network.

To allow a fair assessment of the effect of the embedding scheme, we compared two types of QAST models. Both models used the same MultiMapNet architecture as the mapping network, and the only difference between the two models was the embedding method employed. We varied the qubit count from 2 to 10 (QAST-2 to QAST-10) rather than starting at 3, because at 2 qubits, the existing embedding (E1) and our proposed embedding (E2) yield embedding matrices of the same size, providing a more diagnostic assessment of the embedding method. Table VI summarises the classification accuracy of E2 compared with E1 within the QAST framework for qubit counts ranging from 2 to 10.

TABLE VI
COMPARISON OF ACCURACY BETWEEN QAST MODELS USING EXISTING EMBEDDING (E1) AND PROPOSED EMBEDDING (E2)

| Model | $Acc_{E1}$ (%) | $Acc_{E2}$ (%) | $Acc_{Loss}$ (%) |
|---|---|---|---|
| QAST-2 | $86.1 \pm 10.1$ | $92.4 \pm 4.9$ | $-6.3$ |
| QAST-3 | $95.0 \pm 0.5$ | $94.9 \pm 1.1$ | 0.1 |
| QAST-4 | $94.9 \pm 1.0$ | $95.3 \pm 0.5$ | $-0.4$ |
| QAST-5 | $95.6 \pm 0.7$ | $95.5 \pm 0.6$ | 0.1 |
| QAST-6 | $95.5 \pm 0.4$ | $95.4 \pm 0.4$ | 0.1 |
| QAST-7 | $94.8 \pm 0.9$ | $94.9 \pm 0.6$ | $-0.1$ |
| QAST-8 | $94.1 \pm 0.4$ | $93.7 \pm 0.7$ | 0.4 |
| QAST-9 | $91.9 \pm 1.2$ | $91.0 \pm 1.7$ | 0.9 |
| QAST-10 | $89.5 \pm 2.4$ | $85.6 \pm 2.8$ | 3.9 |

Both embedding methods show comparable performance overall. In QAST-2, where both E1 and E2 use two vectors for embedding, the model using our proposed method achieves higher classification accuracy, 92.4% ($\pm$ 4.9%), which is a 6.3% improvement over the model using E1. Additionally, the model trained using E1-based QAST shows a larger standard deviation of 10%, which is twice as high as that of E2-based QAST. From QAST-3 to QAST-8, the difference in classification accuracy between the two embedding methods is minimal, ranging from approximately 0.1% to 0.4%. However, in higher-qubit models such as QAST-10, E1 achieves slightly better performance, with a classification accuracy about 4% higher than that of E2.

The training times of different QAST models using E1 and E2 are shown in Fig. 10. It is evident that the proposed embedding consistently reduces training time across all models compared to the existing embedding, with reductions ranging from 1.6% to 39.6%. For QAST-2, since the number of vectors used for embedding is similar, the training time per epoch for both embedding methods is nearly the same, at around 1.2 seconds. However, as the qubit count increases to 3 (QAST-3) and 4 (QAST-4), the training times per epoch for E1 rise to 1.33 seconds and 1.50 seconds, respectively, while E2 is 1.21 seconds and 1.29 seconds, achieving reductions of 9.0% and 14.0%. The most dramatic improvements are seen for models on larger numbers of qubits. For QAST-9, E1 requires 2.8 seconds, reduced by E2 to 1.7 seconds (a 39.6% decrease). Similarly, QAST-10 shows E1 at 2.92 seconds and E2 at 1.95 seconds (a 33.2% reduction). Moreover, as the number of qubits increases, the time reduction increases remarkably, from 1.6% for 2 qubits to 33.2% for 10 qubits. This trend can be observed in the line chart shown in Fig. 10. A notable observation is that the training time of E1 increases by 143.3%,

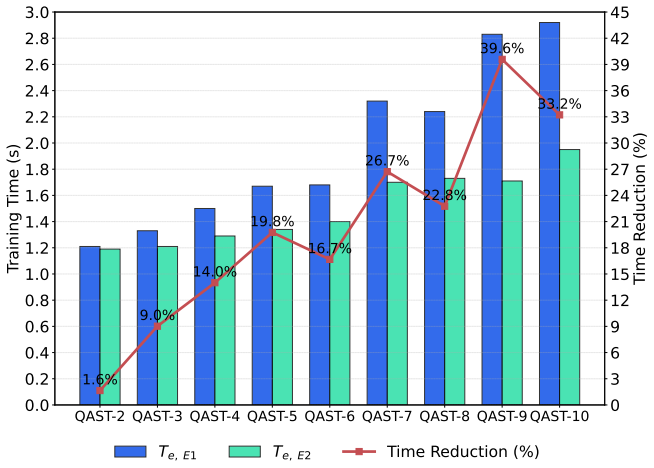


Fig. 10. Training time comparison of QAST models using existing embedding (E1) and proposed embedding (E2).

TABLE VII
COMPARISON OF QAST MODELS OF DIFFERENT $Q$ WITH THE MLP MODEL

| Model | $\Delta\mathcal{C}$ (%) | $Acc$ (%) | $Acc_{Loss}$ (%) | $T_e$ | $\Delta T_e$ (%) |
|---|---|---|---|---|---|
| MLP | – | 92.72 ± 0.9 | – | 6.92 | – |
| QAST-3 | 24.99 | 91.86 ± 0.9 | −0.86 | 1.65 | 76.16 |
| QAST-4 | 62.49 | 92.54 ± 0.6 | −0.18 | 1.72 | 75.14 |
| QAST-5 | 81.25 | 93.48 ± 0.8 | 0.76 | 1.77 | 74.42 |
| QAST-6 | 90.61 | 93.31 ± 0.7 | 0.59 | 1.80 | 73.99 |
| QAST-7 | 95.30 | 94.23 ± 0.5 | 1.51 | 2.08 | 69.94 |
| QAST-8 | 97.64 | 93.77 ± 0.9 | 1.05 | 2.11 | 69.51 |
| QAST-9 | 98.81 | 94.52 ± 0.5 | 1.80 | 2.00 | 71.10 |
| QAST-10 | 99.40 | 93.82 ± 0.4 | 1.10 | 2.47 | 64.31 |

rising from 1.2 seconds at 2 qubits to 2.92 seconds at 10 qubits, while the training time of E2 increases by only 63.5%.

*5) Scalability and Generalizability:* To investigate the scalability, we utilized QAST to train a significantly larger RFFI model, namely an MLP model proposed in [6]. The MLP model comprises three FC layers, each containing 1024 neurons. BN is utilized after each FC layer to enhance training stability, and the ReLU function is used as the activation function. The final layer of the MLP model is an FC layer containing 128 neurons and employs the Leaky ReLU activation function, accompanied by a dropout rate of 0.3. The MLP model has 4,466,703 parameters, which is approximately 12 times more parameters than the CNN model used in earlier experiments. The results of this analysis are shown in Table VII.

Overall, the MLP model trained using the QAST framework outperforms the baseline MLP in classification accuracy. However, QAST-3 and QAST-4 are lower than the baseline by 0.86% and 0.18%, respectively. The classification accuracy of the QAST model rises from 91.86% (± 0.9%) at QAST-3, peaks at 94.52% (± 0.5%) at QAST-9, followed by a slight decline to 93.82% (± 0.4%) at QAST-10. These results demonstrate the scalability of the QAST framework, which can train DL-based RFFI models with a large number of parameters. Training the MLP with QAST achieves a significantly shorter training time than training the MLP using a conventional approach, reducing the training time by approximately 64% to 76%. This occurs because the classical MLP uses substantially more parameters, resulting in longer training time. Applying QAST to the MLP significantly reduces the number of required parameters and, consequently, shortens training time.

More generally, because QAST depends only on the total parameter count of the target model rather than on its specific architecture, it is applicable beyond CNNs. The MLP-based RFFI experiment reported here confirms this for MLP architectures, and the underlying QuantumTrain paradigm has also been applied to RNN models in prior work [50], indicating that the framework generalizes across the DL architectures commonly used in RFFI.

*6) Hardware non-idealities and scalability:* The practical advantage of any hybrid quantum-classical model ultimately depends on hardware non-idealities such as state-preparation error and gate decoherence, and on the control technologies required to mitigate them. QAST is favorable in these respects for three reasons. First, it uses only a small number of qubits and a shallow circuit. Second, it does not encode classical data into the quantum state, thereby avoiding a major source of state-preparation overhead. Third, the quantum circuit is used only during offline training, and the required qubit count grows only logarithmically with model size, keeping the demands placed on quantum hardware modest. Realizing these benefits on physical hardware will nonetheless depend on continued advances in qubit control, calibration, and error mitigation, which remain a key challenge for the broader adoption of hybrid quantum-classical learning.

## VII. CONCLUSION

In this study, we introduced the QAST framework to enhance the training efficiency of RFFI systems based on a hybrid of quantum-classical machine learning. In detail, we employed the QNN in conjunction with MultiMapNet to generate parameters for a classical RFFI model. Furthermore, we propose a sinusoidal embedding that substantially reduces the size of the embedding matrix, thereby mitigating training-time overhead within the QAST framework. Experimental results show that QAST was able to train the CNN-based RFFI model and greatly reduced the number of trainable parameters by over 90% while preserving classification accuracy comparable to conventional training methods. In addition, the proposed embedding method reduces training time by approximately 15% to 30%. Extensive experiments on QAST variants (QAST-3 to QAST-10) showed that QAST-6 offered the best trade-off, achieving near-classical accuracy and high parameter efficiency with only a slight training time increase. Lastly, we evaluated the scalability of QAST by training an MLP-based RFFI model whose size is nearly 12 times larger than that of a CNN-based RFFI model. The results show that its classification accuracy and computational efficiency outperform those of the classical baselines. We also emphasize that QAST is complementary to existing lightweight techniques rather than a replacement for them. Because QAST operates at the training stage, it can be used to train a full RFFI model that is then pruned or quantized for efficient inference, combining training-stage and inference-stage efficiency in a single pipeline. The quantum parameter-generation principle underlying QAST has

further been shown to integrate with low-rank fine-tuning methods such as LoRA, reducing the number of trainable parameters still further [29]. In future work, we will combine QAST with classical model compression methods such as pruning and quantization, and evaluate QAST-trained RFFI models under explicit security and robustness conditions, including open-set unknown-device rejection and resistance to spoofing, replay, and adversarial perturbations. A further direction is to evaluate QAST on noisy quantum simulators and real quantum hardware, to characterize its realistic quantum execution costs, including shot complexity, noise sensitivity, circuit depth, the number of circuit evaluations, quantum–classical communication overhead, and end-to-end latency.

## References


[1] T. T. An, G. Yin, J. Zhang, Y. Ding, T. Q. Duong, and S. L. Cotton, "Towards quantum efficient training for radio frequency fingerprint identification," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Glasgow, UK, 2026, in press.

[2] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 1st Quart. 2016.

[3] Z. Cai, Y. Wang, G. Gui, and J. Sha, "Toward robust radio frequency fingerprint identification via adaptive semantic augmentation," *IEEE Trans. Inf. Forensics Security*, vol. 20, pp. 1037–1048, Jan. 2025.

[4] J. Zhang, F. Ardizzon, M. Piana, G. Shen, and S. Tomasin, "Physical layer-based device fingerprinting for wireless security: From theory to practice," *IEEE Trans. Inf. Forensics Security*, vol. 20, pp. 5296–5325, May 2025.

[5] G. Shen, J. Zhang, and A. Marshall, "Deep learning-powered radio frequency fingerprint identification: Methodology and case study," *IEEE Commun. Mag.*, vol. 61, no. 9, pp. 170–176, Sep. 2023.

[6] G. Shen, J. Zhang, A. Marshall, L. Peng, and X. Wang, "Radio frequency fingerprint identification for LoRa using deep learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2604–2616, Aug. 2021.

[7] P. C. Ng, J. She, and R. Ran, "Compressive RF fingerprint acquisition and broadcasting for dense BLE networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 4, pp. 1196–1210, Apr. 2022.

[8] A. Al-Shawabka, F. Restuccia, S. D'Oro, T. Jian, B. Costa Rendon, N. Soltani, J. Dy, S. Ioannidis, K. Chowdhury, and T. Melodia, "Exposing the fingerprint: Dissecting the impact of the wireless channel on radio fingerprinting," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Toronto, ON, Canada, Jul. 2020, pp. 646–655.

[9] G. Shen, J. Zhang, A. Marshall, and J. R. Cavallaro, "Towards scalable and channel-robust radio frequency fingerprint identification for LoRa," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 774–787, Feb. 2022.

[10] K. Li, J. Bao, X. Xie, J. Hong, and C. Hua, "Receiver-agnostic radio frequency fingerprint identification for zero-trust wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 6, pp. 1981–1997, Jun. 2025.

[11] L. Xie, L. Peng, J. Zhang, A. Gao, H. Fu, and J. Shi, "Channel2channel: Towards robust radio frequency fingerprint extraction and identification," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 11, pp. 3737–3751, Nov. 2025.

[12] G. Shen, J. Zhang, A. Marshall, M. Valkama, and J. R. Cavallaro, "Toward length-versatile and noise-robust radio frequency fingerprint identification," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2355–2367, Apr. 2023.

[13] Y. Zeng, Y. Gong, J. Liu, S. Lin, Z. Han, R. Cao, K. Huang, and K. B. Letaief, "Multi-channel attentive feature fusion for radio frequency fingerprinting," *IEEE Trans. Wireless Commun.*, vol. 23, no. 5, pp. 4243–4254, May 2024.

[14] Y. Lei, D. Li, M. Shao, S. Hong, and H. Sun, "CoSwinVIT: A vision transformer for enhanced uniform spectrum response in specific emitter identification," *IEEE Trans. Inf. Forensics Security*, vol. 21, pp. 1654–1668, 2026.

[15] T. Jian, Y. Gong, Z. Zhan, R. Shi, N. Soltani, Z. Wang, J. Dy, K. Chowdhury, Y. Wang, and S. Ioannidis, "Radio frequency fingerprinting on the edge," *IEEE Trans. Mobile Comput.*, vol. 21, no. 11, pp. 4078–4093, Nov. 2022.

[16] E. Bothereau, A. Chillet, R. Gerzaguet, M. Gautier, and O. Berder, "Investigating sparse neural networks for radio frequency fingerprint identification," in *Proc. IEEE Veh. Technol. Conf. (VTC-Fall)*, Washington, DC, USA, Oct. 2024, pp. 1–6.

[17] Y. Lin, H. Zha, Y. Tu, S. Zhang, W. Yan, and C. Xu, "GLR-SEI: Green and low resource specific emitter identification based on complex networks and fisher pruning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 5, pp. 3239–3250, Oct. 2024.

[18] N. Gao, Y. Liu, Q. Zhang, X. Li, and S. Jin, "Let RFF do the talking: Large language model enabled lightweight RFFI for 6G edge intelligence," *Sci. China Inf. Sci.*, vol. 68, no. 7, p. 170308, Jun. 2025.

[19] M. Henderson, S. Shakya, S. Pradhan, and T. Cook, "Quanvolutional neural networks: Powering image recognition with quantum circuits," *Quantum Mach. Intell.*, vol. 2, no. 1, p. 2, Jun. 2020.

[20] S. N. Paing, J. W. Setiawan, M. A. Ullah, F. Zaman, T. Q. Duong, O. A. Dobre, and H. Shin, "Counterfactual quantum byzantine consensus for human-centric metaverse," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 4, pp. 905–918, Apr. 2024.

[21] T. T. An, S. L. Cotton, J. Zhang, Y. Ding, and T. Q. Duong, "LoRa radio frequency fingerprinting identification using a hybrid quantum-classical neural network," in *Proc. IEEE Veh. Technol. Conf. (VTC-Fall)*, Washington, DC, USA, Oct. 2024, pp. 1–6.

[22] M. Shohibul Ulum, U. Khalid, J. William Setiawan, T. Q. Duong, M. Z. Win, and H. Shin, "Variational anonymous quantum sensing," *IEEE J. Sel. Areas Commun.*, vol. 42, no. 9, pp. 2275–2291, Sep. 2024.

[23] B.-N. D. Tran, M. Fahim, B. D. E. McNiven, M. Guizani, H. Shin, and T. Q. Duong, "Quantum LSTM model for estimation of energy expenditure in human aging using wearable IoT healthcare technology," *IEEE Internet Things J.*, vol. 12, no. 22, pp. 46 051–46 064, Nov. 2025.

[24] B. Hazarika, K. Singh, O. A. Dobre, C.-P. Li, and T. Q. Duong, "Quantum-enhanced federated learning for metaverse-empowered vehicular networks," *IEEE Trans. Commun.*, vol. 73, no. 4, pp. 2348–2363, Apr. 2025.

[25] W. Yu, L. Yin, C. Zhang, Y. Chen, and A. X. Liu, "Application of quantum recurrent neural network in low-resource language text classification," *IEEE Trans. Quantum Eng.*, vol. 5, pp. 1–13, Mar. 2024.

[26] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, "Challenges and opportunities in quantum machine learning," *Nat. Comput. Sci.*, vol. 2, no. 9, pp. 567–576, Sep. 2022.

[27] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep. 2017.

[28] C.-Y. Liu, E.-J. Kuo, C.-H. A. Lin, J. G. Young, Y.-J. Chang, M.-H. Hsieh, and H.-S. Goan, "Quantum-Train: Rethinking hybrid quantum-classical machine learning in the model compression perspective," *Quantum Mach. Intell.*, vol. 7, no. 2, p. 80, Dec. 2025.

[29] C.-Y. Liu, C.-H. H. Yang, H.-S. Goan, and M.-H. Hsieh, "A quantum circuit-based compression perspective for parameter-efficient learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Singapore, Apr. 2025.

[30] V. P. Pham, D. V. Huynh, E. Ak, L. D. Nguyen, B. Canberk, O. A. Dobre, and T. Q. Duong, "Joint optimal design for speed and routing in maritime logistics for green supply chain: A quantum approximate optimization algorithm approach," *IEEE Internet Things J.*, vol. 12, no. 19, pp. 39 556–39 571, Oct. 2025.

[31] E. F. Combarro and S. González-Castillo, *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms*. Birmingham, U.K.: Packt Publishing, Mar. 2023.

[32] C.-Y. Liu, C.-H. A. Lin, C.-H. H. Yang, K.-C. Chen, and M.-H. Hsieh, "QTRL: Toward practical quantum reinforcement learning via quantum-train," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, vol. 2, Montreal, QC, Canada, Sep. 2024, pp. 317–322.

[33] H. Cai, C. Gan, L. Zhu, and S. Han, "TinyTL: Reduce memory, not parameters for efficient on-device learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2020.

[34] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion parameter models," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal. (SC)*, Atlanta, GA, USA, Nov. 2020.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 30, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[36] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, "Noise-induced barren plateaus in variational quantum algorithms," *Nat. Commun.*, vol. 12, no. 1, p. 6961, Nov. 2021.

[37] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, "Connecting ansatz expressibility to gradient magnitudes and barren plateaus," *PRX Quantum*, vol. 3, no. 1, p. 010313, Jan. 2022.

[38] A. N. Kolmogorov, "On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller

number of variables," *Dokl. Akad. Nauk SSSR*, vol. 108, no. 2, pp. 179–182, 1956.

[39] H. Wang, Z. Li, J. Gu, Y. Ding, D. Z. Pan, and S. Han, "QOC: Quantum on-chip training with parameter shift and gradient pruning," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2022, pp. 655–660.

[40] D. Wierichs, J. Izaac, C. Wang, and C. Y.-Y. Lin, "General parameter-shift rules for quantum gradients," *Quantum*, vol. 6, p. 677, Mar. 2022.

[41] H. Wang, Y. Ding, J. Gu, Y. Lin, D. Z. Pan, F. T. Chong, and S. Han, "QuantumNAS: Noise-adaptive search for robust quantum circuits," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Seoul, South Korea, Apr. 2022, pp. 692–708.

[42] T. Proctor, K. Young, A. D. Baczewski, and R. Blume-Kohout, "Benchmarking quantum computers," *Nat. Rev. Phys.*, vol. 7, no. 2, pp. 105–118, Feb. 2025.

[43] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, "Measuring the capabilities of quantum computers," *Nat. Phys.*, vol. 18, no. 1, pp. 75–79, Jan. 2022.

[44] Y. Ueno, S. Imamura, Y. Tomida, T. Tanimoto, M. Tanaka, Y. Tabuchi, K. Inoue, and H. Nakamura, "C3-VQA: Cryogenic counter-based coprocessor for variational quantum algorithms," *IEEE Trans. Quantum Eng.*, vol. 6, pp. 1–17, Jan. 2025.

[45] H. Wang, J. Gu, Y. Ding, Z. Li, F. T. Chong, D. Z. Pan, and S. Han, "QuantumNAT: Quantum noise-aware training with noise injection, quantization and normalization," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jul. 2022, pp. 1–6.

[46] Z. Jiang, T. H. Luan, X. Ren, D. Lv, H. Hao, J. Wang, K. Zhao, W. Xi, Y. Xu, and R. Li, "Eliminating the barriers: Demystifying Wi-Fi baseband design and introducing the PicoScenes Wi-Fi sensing platform," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4476–4496, Mar. 2022.

[47] D. Ha, A. M. Dai, and Q. V. Le, "HyperNetworks," arXiv:1609.09106, Sep. 2016.

[48] ——, "HyperNetworks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Toulon, France, Apr. 2017.

[49] M. Larocca, S. Thanasilp, S. Wang, K. Sharma, J. Biamonte, P. J. Coles, L. Cincio, J. R. McClean, Z. Holmes, and M. Cerezo, "Barren plateaus in variational quantum computing," *Nat. Rev. Phys.*, vol. 7, pp. 174–189, Mar. 2025.

[50] C.-H. A. Lin, C.-Y. Liu, and K.-C. Chen, "Quantum-train long short-term memory: Application on flood prediction problem," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, vol. 02, Montreal, QC, Canada, Sep. 2024, pp. 268–273.