Towards Quantum Efficient Training for Radio Frequency Fingerprint Identification

To Truong An*, Guolin Yin*, Junqing Zhang‡, Yuan Ding§, Trung Q. Duong*†, and Simon L. Cotton*
*Centre for Wireless Innovation (CWI), Queen's University Belfast, Belfast, BT3 9DT, U.K.
†Faculty of Engineering and Applied Science, Memorial University, St. John's, NL A1C 5S7, Canada.
‡School of Computer Science and Informatics, University of Liverpool, Liverpool, U.K.
§Institute of Sensors, Signals and Systems (ISSS), Heriot-Watt University, Edinburgh, U.K.
Emails: {ato01, g.yin, simon.cotton, trung.q.duong}@qub.ac.uk, junqing.zhang@liverpool.ac.uk, yuan.ding@hw.ac.uk.

***Abstract*—Radio frequency fingerprint identification (RFFI) is an emerging physical-layer method for authenticating devices through their unique hardware impairments. Deep learning (DL) is widely used to identify devices based on their signal transmissions. However, training DL models is resource-intensive because it requires repeated updates to numerous parameters, which becomes particularly problematic in resource-constrained environments. In this paper, we propose quantum-assisted training (QAST), a framework that addresses training inefficiencies in RFFI systems. QAST integrates a quantum neural network with a classical neural network to generate parameters for a DL model. Compared to traditional training methods, this indirect training strategy substantially decreases the number of trainable parameters, thereby mitigating the overall computational and resource demands. Experimental results show that QAST enables training an RFFI model with 90% fewer trainable parameters than traditional training approaches, while maintaining comparable classification accuracy.**

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

I. Introduction

Wireless communication technologies are integral to modern life, yet they remain vulnerable to threats, such as impersonation and unauthorized access [1]. While wireless networks are protected by cryptographic mechanisms, their reliance on key management and computational cost limit their applicability to resource-constrained devices [2]. As an alternative, radio frequency fingerprint identification (RFFI) is a physical-layer authentication method, exploiting the unique transmission characteristics of radio frequency (RF) hardware to identify a particular device [3]–[5]. RF signals have unique signatures due to subtle manufacturing defects found in hardware components such as oscillators, amplifiers, and modulators [6]. As a result, these properties can be exploited to differentiate devices even when they originate from the same manufacturing batch [3]. RFFI has been applied to various wireless technologies such as long-range (LoRa) [2], Wi-Fi [7], and ZigBee [8], demonstrating robust classification performance even under varying channel conditions.

While RFFI is a promising technology, the practical implementation of RFFI systems presents several challenges, the most significant being that deep learning (DL)-based RFFI models require substantial computational and memory resources. In particular, during training, the total memory footprint includes not only the model parameters but also the gradients, optimizer states, and intermediate activations, which must be retained for backpropagation [9]. This problem is further exacerbated in real-world deployments, where RFFI models need to be continuously refined based on newly collected data to support continual learning, as hardware characteristics can drift over time due to factors such as aging. With the rapid expansion of wireless networks, the growing number of connected devices further heightens this burden, creating a major obstacle to training and retraining RFFI models at scale. While model compression techniques such as pruning primarily aim to reduce model size during inference [10], they fail to address the fundamental bottleneck of *training inefficiency*, thereby underscoring the need for RFFI solutions that are both accurate and computationally efficient during training.

In this paper, we present a novel training framework for RFFI systems, called quantum-assisted training (QAST), which leverages output probabilities measured from a quantum neural network (QNN) to generate weights and biases for a classical DL model during training. Compared to directly training classical DL models, the QAST framework requires significantly fewer trainable parameters. Consequently, it can substantially reduce overall computational and resource demands. QAST is built upon the QuantumTrain (QT) framework, which was first introduced in the image classification domain [11]. We introduce a multimodal mapping network (MultiMapNet) that efficiently generates multiple classical parameters from a QNN output. This design reduces qubit requirements and mitigates the barren plateau (BP) problem. The main contributions of this work are summarized below:

- We propose QAST, a framework designed to enhance the training efficiency in RFFI systems. To the best of our knowledge, this is the first paper to directly address training inefficiency in RFFI.
- We propose a multimodal mapping network that generates multiple classical parameters from a shared QNN output and leverages multimodal learning to effectively capture the output probabilities of the QNN. By tuning the number of parameters mapped per QNN output, we can

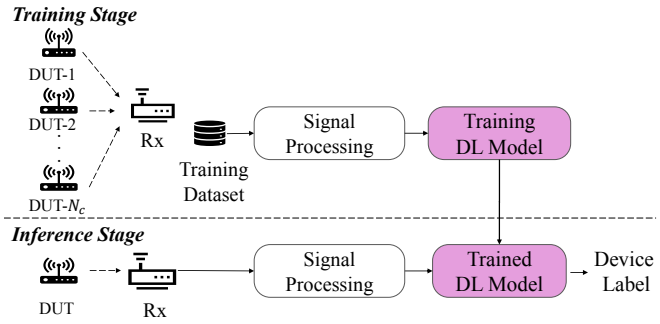
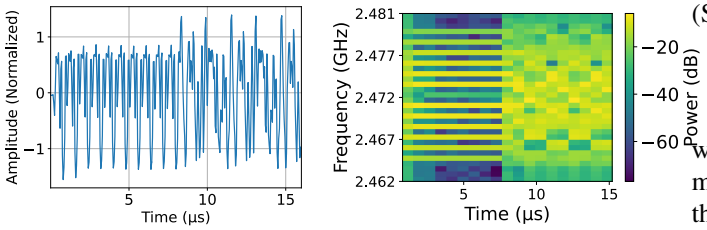


Fig. 1. The overall RFFI system model.



(a) (b)

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

control the number of qubits and mitigate the BP problem.

- We evaluated the performance of the proposed approach by utilizing the QAST to train a RFFI model to classify commodity Wi-Fi devices. In our experiments, we demonstrated that QAST can train a RFFI model requiring only 10% of the original number of parameters, while still achieving comparable classification accuracy.

The rest of the paper is organized as follows. Section II presents the RFFI system. Section III details the design of the QAST framework. Section IV shows the experimental setup. Section V presents results and analysis. Section VI concludes the paper.

II. DL-based RFFI System

As shown in Fig. 1, a DL-based RFFI system consists of two stages, training and inference. In the training stage, numerous labeled transmitted packets from N_c devices under test (DUTs) are collected to form the training dataset, which is used to train the DL model. After training, the DL model is deployed for inference to identify the transmitting device.

A. Signal Processing

During both training and inference, the received signals are preprocessed with carrier frequency offset (CFO) compensation, normalization, and signal representation before being input into the model. For Wi-Fi signals, we first apply CFO compensation to mitigate its effect on RFF feature extraction, followed by power normalization to ensure amplitude consistency across signals. An example of the normalized signal is shown in Fig. 2(a).

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**

After that, the normalized time-domain signal is transformed into spectrograms using the short-time Fourier transform (STFT), defined as

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

where $X(k,m)$ represents an element of the complex STFT matrix, with m indicating the window index and k denoting the frequency bin index. $w[n]$ denotes the window function of length N applied to each signal segment. H is the hop size, which determines how far the window moves forward each time. In this study, $N = 32$, and $H = 16$.

The absolute value of the complex matrix X is computed, which removes the phase information, preserving only the amplitude of X. The power spectrogram, computed in the logarithmic domain, is obtained from (1) as

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

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

B. Deep Learning Model

RFFI systems have employed a variety of DL architectures, including convolutional neural networks (CNNs), multilayer perceptrons (MLPs), and recurrent neural networks (RNNs). Our work is inspired by the CNN-based RFFI model proposed in [3]. This model was chosen because CNNs are well-suited to processing spectrograms. The CNN architecture is shown in Table I. The model comprises 368,511 trainable parameters.

III. Quantum-Assisted Training Framework

A. Overall Quantum-Assisted Training Framework

Our QAST is a hybrid quantum-classical machine learning approach designed to improve training efficiency in classical RFFI systems. As shown in Fig. 3, QAST utilizes a quantum parameter generator (QPG) to produce parameters for a CNN model. The QPG comprises three modules: a QNN, an embedding module, and a mapping network.

In this study, a QNN is developed using parameterized quantum circuits (PQC). The QNN layer utilizes the U_3 gate, a single-qubit gate that enables precise manipulation of quantum states [12]. The U_3 gate can be represented as follows

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

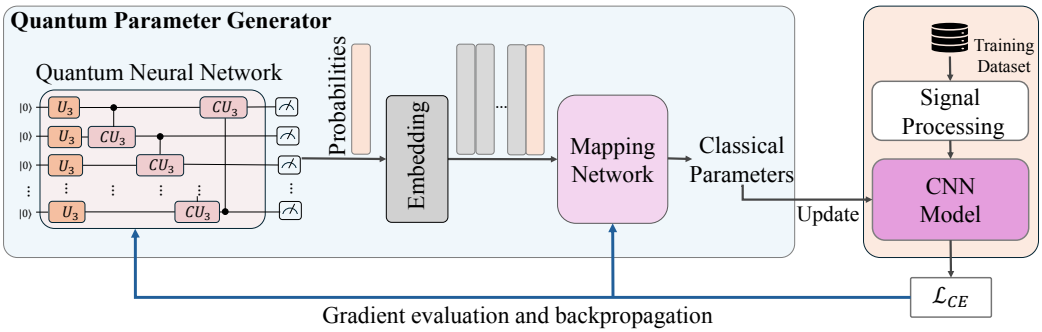


Fig. 3. The proposed quantum-assisted training framework.

where θ, φ, λ are real parameters that define the rotation.

Besides the U_3 gate, the controlled-U_3 (CU_3) gate is utilized to generate entanglement between qubits and is expressed as

$$CU_3 = I \otimes |0\rangle\langle 0| + U_3(\tilde{\theta}, \tilde{\varphi}, \tilde{\lambda}) \otimes |1\rangle\langle 1|. \tag{4}$$

Notably, the CU_3 gate is commonly designed with a circular layout in quantum circuit diagrams [12]. Fig. 3 presents a block diagram of the QNN architecture, showing the arrangement of PQC with U_3 and CU_3 gates across Q qubits. The QNN output quantum state is expressed as

$$|\psi(\boldsymbol{\beta})\rangle = \left(\prod_r CU_3^{r,r+1}(\tilde{\theta}, \tilde{\varphi}, \tilde{\lambda}) \prod_s U_3^s(\theta, \varphi, \lambda) \right) |0\rangle^{\otimes Q}, \tag{5}$$

where $\beta = \{(\tilde{\theta}, \tilde{\varphi}, \tilde{\lambda}), (\theta, \varphi, \lambda)\}$ denotes the trainable parameters of QNN. U_3^s represents a single-qubit U_3 rotation applied to qubit j, $CU_3^{r,r+1}$ is a controlled-U_3 gate with control qubit r and target qubit $r+1$. The operators $\prod_r$ and $\prod_s$ represent ordered products of gates applied sequentially over qubit indices $r = 1, \ldots, Q-1$ and $s = 1, \ldots, Q$, respectively.

The measurement outcome probabilities of the QNN, with Q qubits yielding $\mathcal{P} = 2^Q$ possible outcomes in the computational basis, can be collected into a probability vector

$$\Psi = \begin{bmatrix} |\langle \phi_1 | \psi(\beta) \rangle|^2 \\ \vdots \\ |\langle \phi_k | \psi(\beta) \rangle|^2 \\ \vdots \\ |\langle \phi_{\mathcal{P}} | \psi(\beta) \rangle|^2 \end{bmatrix}, \tag{6}$$

where $|\phi_k\rangle$ denotes the k-th computational basis state.

In the embedding module, the probability vector is augmented by combining the measurement probabilities with the corresponding computational basis states. 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 embedding matrix satisfies $\mathcal{E}(\Psi) \in [-1, 1]^{\mathcal{P} \times (Q+1)}$. For instance, the first row of the embedding matrix $\mathcal{E}(\Psi)$, can be given as

$$\mathcal{E}(\Psi)_1 = |\phi_1\rangle \,\|\, |\langle \phi_1 | \psi(\beta) \rangle|^2 \tag{7}$$

$$= [\underbrace{-1, -1, \ldots, -1}_{Q \text{ elements}}, |\langle \phi_1 | \psi(\beta) \rangle|^2]. \tag{8}$$

Finally, the mapping network converts these embedded probabilities into the weights and biases of the target model. For example, consider a classical CNN-based RFFI model that comprises C_{CNN} trainable parameters, and $\theta_{\text{CNN}} = (\theta_1, \theta_2, \ldots, \theta_{C_{\text{CNN}}})$ denotes the trainable parameter vector. A classical neural network serves as a mapping model, denoted G with tunable parameters Θ. The classical parameters generated by the QAST framework can be expressed as

$$\theta = G_\Theta(\mathcal{E}(\Psi)), \tag{9}$$

where θ is the parameter vector generated by the QPG and has length C_{CNN}.

B. Proposed Mapping Network

1) Existing Mapping Network: In [12], [13], the authors employ an MLP to map each QNN output directly to a corresponding parameter of the target model. The architecture of the MLP mapping network is shown in Fig. 4(a). However, their one-to-one mapping strategy is inefficient, as it demands a large number of qubits to train a large DL model. For example, to train our CNN with $C_{\text{CNN}} = 368,511$ parameters, the QAST-MLP framework requires 19 qubits, as determined by

$$Q = \lceil \log_2 \; C_{\text{CNN}} \rceil. \tag{10}$$

One limitation of existing methods is that, as the number of qubits increases, QNNs suffer from the BP problem, where the gradient of the cost function vanishes exponentially [14], thereby severely hindering the training process. The BP effect will be examined in detail in Section IV-C2.

2) MultiMapNet: We propose MultiMapNet, a multimodal mapping network designed to address these limitations and improve the mapping network's ability to learn measured probabilities from a QNN. The key idea is to restructure the mapping network so that multiple classical parameters are generated from each 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 is $n_G = \lceil C_{\text{CNN}}/n_B \rceil$, where each group contains n_B parameters.

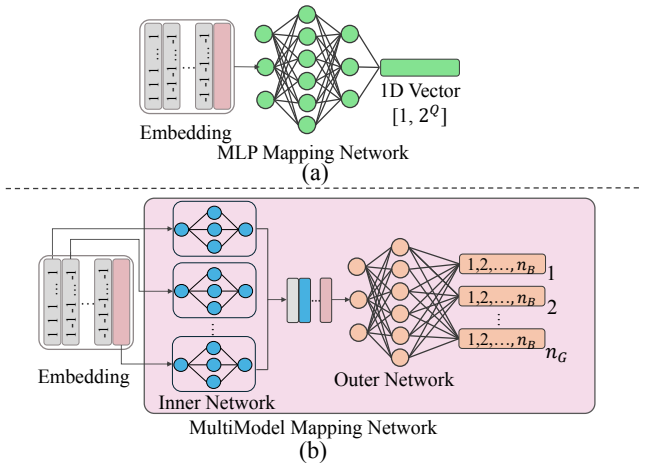


Fig. 4. Overview of the mapping network in the quantum-assisted training framework. (a) Existing architecture. (b) Proposed architecture.

TABLE II
DETAILED ARCHITECTURE OF THE *MULTIMAPNET* MODEL

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

†*The inner function is applied independently to each input variables.*

As illustrated in Fig. 4(b), MultiMapNet comprises two primary components. First, an inner network that independently analyzes each input variable, allowing the model to better capture QNN probability distributions by decoupling variable influences. Second, an outer network is a MLP model that processes the inner network's outputs, generating a $[n_G, n_B]$ matrix. This matrix is then reshaped into a one-dimensional vector of length $n_G \times n_B$. The configuration of the mapping network as reported in Table II. To center the data, the mean of this vector is subtracted from each of its elements, resulting in a vector with zero mean [12]. This centered vector represents the raw parameter values for the classical CNN. By restructuring the parameter mapping process, the required number of qubits is reduced from $Q = \lceil \log_2 C_{\text{CNN}} \rceil$ to $Q = \lceil \log_2 n_G \rceil$. This design enables active control of the number of qubits (by adjusting n_B). Such control is crucial in the QAST framework because it helps reduce the qubit requirements, thereby mitigating the BP problem.

C. Training Flow for QAST Framework

At each training iteration, the QPG generates the full set of CNN parameters. We then replace the CNN's parameters with θ produced by the QPG. After the parameter update, the CNN processes an input from the training set to produce the prediction, which is evaluated by a cross-entropy (CE) loss function. The CE loss measures the divergence between predicted class probabilities $\hat{y}$ and actual labels y, which is mathematically expressed as

$$\mathcal{L}_{CE} = -\frac{1}{N_{\text{Batch}}} \sum_{i=1}^{N_{\text{Batch}}} \sum_{c=1}^{N_c} y_{i,c} \log(\hat{y}_{i,c}), \tag{11}$$

where N_{Batch} is the batch size, N_c the number of classes.

As the CNN parameters were generated by QNN and the mapping network, the parameter dependency is $\theta = \theta(\beta, \Theta)$. During training, the QNN and mapping network parameters are optimized to reduce the CE loss. By applying the chain rule and parameter-shift rule, the gradient of loss with respect to joint parameters is given by [13]

$$\nabla_{\beta,\Theta} \mathcal{L}_{\text{CE}} = \left(\frac{\partial \theta}{\partial(\beta, \Theta)} \right)^{\top} \nabla_{\theta} \mathcal{L}_{\text{CE}}, \tag{12}$$

where $\frac{\partial \theta}{\partial(\beta,\Theta)}$ is the jacobian, describing how the generated parameters θ change when adjusting the QNN and mapping network parameters. $\nabla_{\theta} \mathcal{L}_{\text{CE}}$ is the gradient of the loss with respect to classical weights of CNN model. A detailed description of the backpropagation can be found in [13].

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

IV. Experimental Evaluation

A. Data Collection and Experimental Setup

In this work, as illustrated in Fig. 5, we utilized a USRP N210 software-defined radio 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 vertical polarized. On the receiver side, the USRP N210 was linked to another Linux laptop running PicoScenes [15] to decode the Wi-Fi packets and store signals in the dataset. A diagram of the data collection setup is shown in Fig. 6.

B. Training Hyperparameter Configuration

All experiments were conducted 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 were allowed. Early stopping was applied, stopping the training process if the validation loss did not improve for 30 consecutive epochs.

C. Experimental Results

We report the average classification accuracy and standard deviation obtained from ten consecutive runs to ensure statistical reliability and reproducibility.

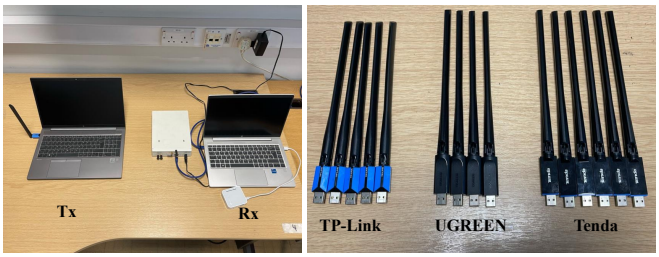


Fig. 5. The experiment setup and devices. (a) Transceiver setup. (b) Wi-Fi dongles.

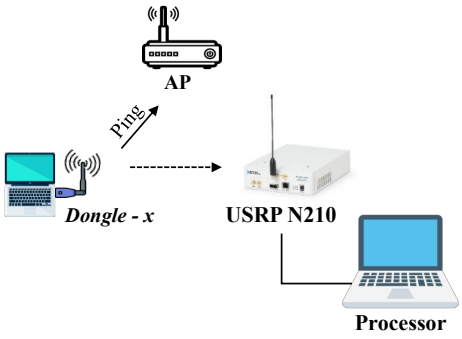


Fig. 6. The data collection setup.

1) Performance of the RFFI Model and QAST Framework: We compare the CNN-based RFFI model trained with the classical approach (denoted CNN) and with the QAST framework (denoted QAST), using overall accuracy (Acc) as the primary metric. The accuracy loss (Acc_{Loss}) is defined as the difference between the CNN and QAST accuracies.

To assess computational efficiency, we use parameter efficiency (PE), defined as $\Delta C\ (\%) = \left(1 - C_{\text{QAST}}/C_{\text{CNN}}\right) \times 100$, where C_{QAST} is the number of parameters in the QAST framework and C_{CNN} is that of the baseline. We also report the average training time per epoch (T_e) in seconds. To quantify the impact on training speed, we calculate the relative change in training time as

$$\Delta T_e\,(\%) = \frac{T_e^{\text{CNN}} - T_e^{\text{QAST}}}{T_e^{\text{CNN}}} \times 100, \tag{13}$$

This section evaluates the QAST framework's ability to train RFFI models with significantly fewer parameters while maintaining performance comparable to traditional training approaches. We evaluate its performance by varying the number of qubits from 3 to 10, denoted as QAST-3 through QAST-10. The QAST starts to exhibit improved parameter efficiency over the baseline model when $Q \geq 3$. For example, QAST-3 achieves a PE of 24.9%. Notably, PE increases significantly with the number of qubits, reaching up to 99.3% with QAST-10.

As illustrated in Table III, the QAST model's classification accuracy exhibits a distinct unimodal relationship with qubit scaling, peaking at $Q = 5$ before a significant decline.

TABLE III
COMPREHENSIVE COMPARISON OF QAST MODELS OF DIFFERENT Q WITH THE CLASSICAL CNN[1]

Model Name	ΔC (%)	Acc	Acc_{Loss}	T_e	ΔT_e (%)
CNN	0	95.8 ± 0.3	0.0	1.28	–
QAST-3	24.89	95.0 ± 0.5	0.8	1.33	−5.47
QAST-4	62.39	94.9 ± 1.0	0.9	1.50	−17.18
QAST-5	81.14	95.6 ± 0.7	0.2	1.67	−30.47
QAST-6	90.51	95.5 ± 0.4	0.3	1.68	−31.25
QAST-7	95.20	94.8 ± 0.9	1.0	2.32	−81.25
QAST-8	97.54	94.1 ± 0.4	1.7	2.24	−75.00
QAST-9	98.71	91.9 ± 1.2	3.9	2.83	−121.09
QAST-10	99.30	89.5 ± 2.4	6.3	2.92	−121.13

Accuracy improves from 95.0% ± 0.5% in QAST-3 to a peak of approximately 95.5% in QAST-5 and QAST-6, indicating that increasing the number of qubits up to this point enhances QAST's ability to train the classical CNN effectively. Moreover, PE also improves substantially, rising from 24.9% in QAST-3 to 90.5% in QAST-6.

However, beyond QAST-6, the improvement in PE exhibits diminishing returns, increasing by less than 9% from QAST-6 to QAST-10, while classification accuracy drops rapidly from 95.6% ± 0.7% in QAST-5 to 85.6% ± 2.4% in QAST-10. This decline is mainly due to the fact that the mapping network becoming very small when PE exceeds 95%, leading to underfitting. Additionally, the BP issue, in which gradients vanish exponentially as the qubit count increases, may further impede effective training. This pattern suggests an optimal qubit count of $Q = 6$, 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 around 95.5%.

Regarding training efficiency, most of the QAST models require longer training times per epoch compared to the classical CNN baseline. The training times per epoch for QAST-3 to QAST-6, for example, which take around 1.33 to 1.68 seconds per epoch, are approximately 5% to 30% higher than those for the classical CNN model. As the number of qubits increases, the amount of time it takes to train per epoch increases from 1.33 seconds (QAST-3) to 2.92 seconds (QAST-10). The longest observation is for QAST-10, which takes 2.92 seconds per epoch, 121% longer than that of the CNN baseline. The additional cost in training time is incurred because the current QAST is constructed on classical computers using TorchQuantum, which requires significant computation, hence impacting the response time. When quantum computing matures, the training time of the QAST framework will likely be dramatically reduced when executing on actual quantum hardware.

[1] ΔT_e (%) column quantifies the relative change in training time per epoch, where negative values indicate an increase in time compared to the CNN baseline.

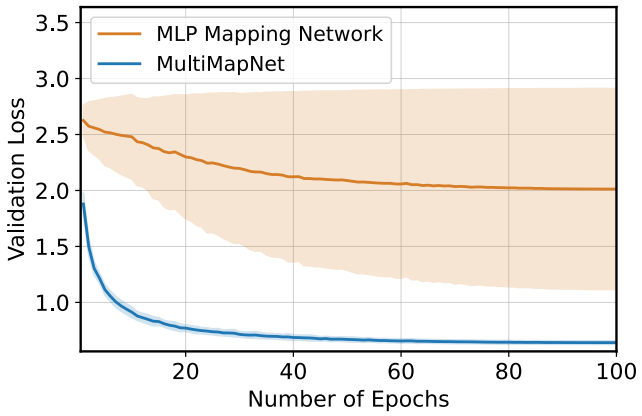


Fig. 7. Comparison of validation loss (mean and standard deviation across 10 runs) for QAST models using MLP Mapping Network and MultiMapNet.

2) Evaluation of the Proposed MultiMapNet: In this subsection, we evaluate the performance of the proposed mapping network by comparing MultiMapNet with MLP-based mapping networks. To make a fair comparison, both QAST models use the same embedding method as proposed in [12], [13]. Fig. 7 compares the convergence behavior of QAST when using either an MLP-based mapping network or the proposed MultiMapNet. The QAST-MLP framework exhibits unstable training, as indicated by the large variance across runs. This instability arises from the large qubit count. To train a CNN-based RFFI model, the QAST-MLP framework requires 19 qubits, as shown in (10). At such a high qubit count, the BP effect becomes more pronounced, making optimization extremely difficult.

In contrast, MultiMapNet enables proactive control of the qubit count as a hyperparameter, thereby improving scalability and avoiding the need for an excessive number of qubits. In this experiment, we used QAST with 6 qubits. As shown in Fig. 7, the mean validation loss of MultiMapNet is significantly lower than those of the MLP mapping network, demonstrating that the proposed MultiMapNet outperforms existing mapping networks. Moreover, the small standard deviation of the validation loss indicates that MultiMapNet effectively mitigates the BP problem.

V. Conclusion

In this paper, we proposed QAST to address the issue of *training inefficiency* in RFFI systems. Specifically, we utilized the QNN in conjunction with MultiMapNet to generate parameters for classical RFFI models. Moreover, we introduced a multimodal mapping network, which significantly outperforms existing mapping networks and avoids the BP problem. 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 PE. Experimental results demonstrate that QAST-6 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.

Acknowledgements

This work was supported in part by the Cyber AI Hub Doctoral Training Program at the Centre for Secure Information Technologies (CSIT), Queen's University Belfast, supported by the UK Government as part of the New Deal for Northern Ireland, administered through Innovate UK/UKRI. It was also supported by the Canada Excellence Research Chair (CERC) Program CERC-2022-00109 and the UK Engineering and Physical Sciences Research Council (EPSRC), through the EPSRC Hub on All Spectrum Connectivity (EP/X040569/1 and EP/Y037197/1).

References

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

[2] 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, 2022.

[3] 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, 2021.

[4] 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 100th Veh. Technol. Conf. (VTC-Fall)*, 2024, pp. 1–6.

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

[6] 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, 2025.

[7] R. Kong and H. Chen, "DeepCRF: Deep learning-enhanced CSI-based RF fingerprinting for channel-resilient WiFi device identification," *IEEE Trans. Inf. Forensics Secur.*, vol. 20, pp. 264–278, 2025.

[8] L. Peng, J. Zhang, M. Liu, and A. Hu, "Deep learning-based RF fingerprint identification using differential constellation trace figure," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 1091–1095, 2020.

[9] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion-parameter models," in *Proc. Int. Conf. High Performance Comput., Netw., Storage Anal. (SC)*, 2020, pp. 1–16.

[10] 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, 2022.

[11] C.-Y. Liu, E.-J. Kuo, C.-H. A. Lin, S. Chen, J. G. Young, Y.-J. Chang, and M.-H. Hsieh, "Training classical neural networks by quantum machine learning," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, 2024, pp. 34–38.

[12] 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)*, 2024, pp. 317–322.

[13] 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 Machine Intelligence*, vol. 7, no. 2, p. 80, 2025.

[14] 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, 2022.

[15] 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, 2022.