Quantum-Assisted Memory-Efficient Training for Parameter-Intensive Wi-Fi-Based Human Activity Recognition

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

***Abstract*—Wi-Fi-based human activity recognition (HAR) has become an important part of integrated sensing and communications, paving the way for a range of context-aware services. However, most existing Wi-Fi-based HAR systems rely on deep learning (DL) models that are computationally and memory intensive in both training and inference, which poses significant challenges for real-world deployment. Conventional training requires simultaneous updates of millions of parameters, leading to prohibitive memory consumption. In this paper, we propose a novel quantum-assisted memory-efficient training framework (Q-MET) designed to improve efficiency in both training and inference. Q-MET utilizes a hybrid quantum classical neural network to indirectly generate parameters for HAR models, significantly reducing the trainable parameter count compared to direct optimization. To further support the deployment on resource-constrained devices, we integrate structured pruning during the training phase. Experimental results demonstrate that Q-MET achieves a 90% to 95% reduction in trainable parameters compared with conventional backpropagation-based DL training while maintaining or even exceeding classical classification accuracy. Additionally, Q-MET supports lightweight inference through structured pruning, achieving 75% to 85% model sparsity with less than 2% loss in classification accuracy. To the best of our knowledge, this work represents the first quantum-assisted approach to simultaneously tackle memory inefficiencies in both the training and inference stages of HAR systems.**

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, in part by the UK Government as part of the New Deal for Northern Ireland, administered through Innovate UK/UKRI, 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) CREATE program (grant number 596205-2025), and in part by the UK Engineering and Physical Sciences Research Council (EPSRC) through the EPSRC Hub on All Spectrum Connectivity under Grant EP/X040569/1 and Grant EP/Y037197/1.

To Truong An, Jie Zhang, Guolin Yin, and Simon L. Cotton are with the Centre for Wireless Innovation, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK (e-mail: ato01@qub.ac.uk, jie.zhang@qub.ac.uk, g.yin@qub.ac.uk, simon.cotton@qub.ac.uk).

Junqing Zhang is with the School of Computer Science and Informatics, University of Liverpool, Liverpool, L69 3DR, U.K. (email: Junqing.Zhang@liverpool.ac.uk).

Yanjiao Li is with the Institute of Engineering Technology, University of Science and Technology Beijing, Beijing 100083, China. (e-mail: yanjiaoli@ustb.edu.cn).

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, School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Belfast BT7 1NN, UK (e-mail: tduong@mun.ca).

Corresponding author is Trung Q. Duong.

***Index Terms*—Human activity recognition, model compression, quantum machine learning, training optimization, Wi-Fi sensing.**

I. Introduction

Over the last decade, the rapid proliferation of sensing technologies has revolutionized human activity recognition (HAR) across various fields, enabling applications ranging from smart home, elder care, and safety monitoring [1]–[6]. Among these technologies, Wi-Fi sensing has emerged as a particularly cost-effective solution, leveraging existing infrastructure to extract rich environmental information without the need for dedicated hardware sensors. Recognizing this potential, the IEEE 802.11bf Task Group was formed to work on an amendment to the IEEE 802.11 standard to specifically deliver support for Wi-Fi sensing [7]–[10]. This is a major step toward enabling reliable, scalable, and deployable Wi-Fi sensing systems for HAR and related applications.

Wi-Fi sensing typically relies on either received signal strength indicator (RSSI) or channel state information (CSI) [11]–[13]. Although RSSI is widely accessible, it is coarse-grained and susceptible to environment fluctuations, limiting its stability and discriminative capability [14]–[16]. In contrast, CSI captures fine-grained physical-layer properties such as amplitude and phase responses across subcarriers. This allows for robust characterization of subtle motions and environmental changes [17]–[20].

To exploit these high-dimensional features, deep learning (DL) models have become the standard for Wi-Fi-based HAR offering the ability to automatically learn discriminative representations from large-scale datasets [21]–[23]. Numerous studies have investigated the robustness of DL-based HAR models, focusing on their classification accuracy, resilience to channel variations, and inference efficiency [24]–[28]. However, the widespread adoption of DL in HAR faces a critical challenge, the significant memory requirements. While DL models are typically evaluated in environments with abundant resources, practical HAR deployment often targets resource-constrained edge platforms and commercial-off-the-shelf (COTS) devices.

In recent years, cloud computing has emerged as a promising approach for handling large-scale data processing and training DL models [29]–[31]. Due to the limited computational and energy resources of edge devices, the training of DL-based HAR models is often offloaded to cloud platforms

to leverage scalable resources, while inference is typically performed either locally on devices or at the edge. However, cloud-centric approaches introduce latency, privacy concerns, and reliance on network connectivity, drawbacks that are often unacceptable for real-time applications [32]. On-device learning and inference offer a solution to these issues but must operate within strict memory and power constraints [33].

Although training can be offloaded to cloud infrastructures, reducing training-related memory costs remains a critical challenge. Training DL models requires significant memory for storing model parameters, optimizer states, intermediate activations, and gradients [34]. Specifically, the backpropagation mechanism underlying every parameter update involves the storage of layer-by-layer activations to compute gradients. Consequently, training requires a large memory footprint, substantially greater than the memory required during inference [35]. As wireless networks increasingly host DL-based applications, from wireless sensing, physical-layer security, channel estimation, and related tasks [36]–[40], this training memory overhead becomes a prohibitive barrier to practical deployment [41], [42]. However, most existing studies prioritize inference optimization and leave the issue of inefficient training largely unaddressed.

For the purpose of reducing model complexity and deploying models in real-world applications, especially on resource-constrained devices, researchers have extensively studied model compression techniques, primarily targeting inference efficiency. One of the most widely used strategies is quantization, reducing precision in model parameters by converting high-precision representations, such as FP32 (32-bit floating point), to low-bit forms such as INT8 (8-bit integer) or INT4 (4-bit integer). This method significantly reduces both memory footprint and computational cost [43]. The other popular technique in model compression is pruning, which involves finding redundant or less-significant connections from the neural network [44]. The pruning algorithm keeps only the most important weights and pathways, which reduces computation with little or no impact on the performance.

Despite these advances, state-of-the-art HAR optimization focuses almost exclusively on the inference phase. These methods, such as quantization and pruning, typically require training full-scale, parameter heavy models before applying compression techniques, and optionally additional fine-tuning, which further increases the computational and memory costs during training [24], [32]. As a result, these approaches remain memory-intensive and computationally expensive during the training stage, rendering them unsuitable for memory-limited platforms.

To bridge this gap, we explore an emerging paradigm of quantum-assisted machine learning. The QuantumTrain (QT) framework utilizes a hybrid quantum-classical neural network to generate parameters for classical DL models. Because the quantum circuit acts as a highly expressive generator using significantly fewer trainable parameters than the target classical network, QT offers a pathway to massive reductions in training memory. While QT has been applied to image classification [45], large language models (LLMs) [46], and physical layer authentication [47], it has not yet been adapted to the specific constraints of HAR, nor does it inherently address inference efficiency.

In this paper, we propose a novel Quantum-assisted Memory-Efficient Training (Q-MET) framework. The primary objective of Q-MET is to enhance efficiency in both the training and inference phases of HAR system by introducing a pruning-enhanced QT approach. Unlike traditional approaches, where pruning is applied after the model has been fully trained, and often requiring additional fine-tuning, introducing extra computational overhead. Our method applies structured pruning directly within the QT training process. This design allows Q-MET to eliminate redundant parameters as it learns, avoiding the inefficiencies of post-training pruning. Ultimately, Q-MET produces a pruned and compact DL model that is immediately suitable for on-device inference, making it especially valuable for deployment in real-time, resource-limited HAR applications. By addressing inefficiencies in both training and inference, Q-MET provides a comprehensive solution for scaling DL-based HAR to practical, low-resource environments. Our main contributions are highlighted as follows

- We identify and address the open problem of training inefficiency in HAR systems, which poses significant challenges in real-world deployments. To the best of our knowledge, this is the first work to explicitly target training efficiency in this domain.
- We extend the QT framework to HAR by designing a hybrid quantum classical neural network, consisting of a quantum neural network and a classical mapping network, to generate the parameters of DL-based sensing models efficiently.
- We propose Q-MET, a novel memory-efficient end-to-end training framework for HAR. In Q-MET, structured pruning is applied directly within the QT process, enabling the model to remove redundant parameters during training, resulting in compact models that are directly deployable for on-device inference without post-training compression.
- We evaluate the performance of the proposed approach by applying Q-MET to train a DL-based HAR model on two public datasets: Widar3.0 [48] and UT-HAR [49]. We experimentally demonstrate that Q-MET enables the training of DL-based HAR models with up to 95% fewer trainable parameters compared with standard backpropagation-based training, substantially reducing memory requirements and enabling efficient training in cost-sensitive cloud environments, while preserving or slightly enhancing model performance. Furthermore, Q-MET enables lightweight inference by producing highly sparse models through structured pruning, achieving 75% to 85% sparsity without significant performance degradation, which makes the resulting models well suited for resource-constrained HAR deployments.

The remainder of the paper is organized as follows. Section II reviews related works on Wi-Fi-based HAR systems and the QT framework, and Section III introduces preliminaries on quantum computing and pruning. Section IV presents the on-device Wi-Fi-based HAR system, and Section V details the

TABLE I
SUMMARY OF NOTATION

Symbol	Definition
$U(\theta)$	Parameterized unitary of the PQC
$U_L(\theta_L)$	Parameterized unitary at the L-th layer
$\|\psi(\theta)\rangle$	Output state of the PQC
$p_\theta(z)$	Probability of measurement outcome z
z	Possible measurement outcome
$H(n)$	Channel frequency response (CFR)
$X(n)$	Transmitted signal in the frequency domain
$W(n)$	Additive white Gaussian noise (AWGN)
$\|H_k\|$	Amplitude of the k-th subcarrier
$\angle H_k$	Phase of the k-th subcarrier
θ_{PQC}	Trainable parameters of the PQC
N_q	Number of qubits
Ψ	Measurement probability vector of the PQC
$\|\phi_k\rangle$	k-th computational basis state
$\mathcal{P}$	Number of basis states, $\mathcal{P} = 2^{N_q}$
$\mathcal{S}(\Psi)$	Sinusoidal embedding of Ψ
S_0, S_1	Sinusoidal embedding vectors
M_{θ_m}	Mapping network with parameters θ_m
θ_m	Trainable parameters of the mapping network
Θ_m	Matrix of generated classical parameters
n_G	Number of parameter groups
n_B	Number of parameters per group
$\overrightarrow{\theta}_{\text{ResNet-18}}$	Generated parameter vector of ResNet-18
$\mathcal{Z}(\cdot)$	Forward function of ResNet-18
$\mathcal{L}_{CE}$	Cross-entropy loss
N_{Batch}	Batch size
N_c	Number of output classes
η	Learning rate
$C_{\text{ResNet-18}}$	Total number of parameters in ResNet-18
$C_{\text{Q-MET}}$	Total trainable parameters of Q-MET
$\Delta\mathcal{C}$	Parameter efficiency gain
score$(u; W)$	LAMP importance score of weight u
p	Pruning ratio
s	Sparsity level
T_e	Average training time per epoch

design of Q-MET. Section VI presents the experimental setup and highlights the key results, and Section VII concludes the paper. For convenience, the main symbols used throughout the paper are summarized in Table I.

II. Related Work

This section presents a review of prior research on Wi-Fi-based HAR systems, quantum machine learning and QT framework. Table II provides a comparative summary of the existing methods in relation to our proposed approach. As summarized in Table II, prior HAR compression methods [24], [32], [50] target inference efficiency only, whereas the QuantumTrain family [45], [46], [51]–[53] targets training efficiency only. In contrast, Q-MET is, to the best of our knowledge, the first framework to jointly address both training- and inference-stage memory efficiency. Moreover, Q-MET is compatible with existing compression techniques, as the structured pruning used in this work can be replaced by alternative pruning or quantization methods.

A. Wi-Fi-based Human Activity Recognition Systems

Wi-Fi-based HAR systems are capable of obtaining contextual information about human activities by analyzing changes of wireless signals [54], [55]. In recent years, DL-based methods have been widely used for Wi-Fi-based HAR. For instance, Li et al. [56] first developed a domain-independent motion change pattern of gestures, and then proposed a novel dual-task DL framework for gesture recognition. Gu et al. [57] proposed WiGRUNT based on a dual-attention mechanism to enhance cross-domain gesture recognition. Zhang et al. [58] presented a modified graph-based few-shot learning approach, which leveraged complex-valued convolutions in the frequency domain followed by channel-wise attention for Wi-Fi-based HAR. To alleviate the complexity and computational overhead of conventional DL models while enhancing the robustness and generalization of wireless sensing systems, lightweight neural networks and online learning methods have been increasingly adopted in HAR. Additionally, modified online sequential learning algorithms were developed to enhance Wi-Fi-based contactless localization, enabling robust and efficient domain adaptation without retraining existing models [59], [60].

Several studies have investigated the use of quantization and pruning techniques to enhance model efficiency in HAR systems. In [32], the authors demonstrated that the post-training-quantization could reduce up to 72% model size of DL-based HAR model, while achieving an accuracy degradation within 5%. Alternatively, channel pruning was employed in the design of the lightweight Wi-Fi CSI human sensing system (LWiHS) [24]. This approach achieved a reduction of over 50% in the number of parameters, with less than a 0.5% reduction in accuracy. While in [50], a hybrid pruning-quantization approach have achieved significant complexity reductions. First, post-training pruning was used to reduce the model's complexity, followed by quantizing the already pruned weights into the INT8 format. This hybrid approach provided a 27% reduction in both computational complexity and storage, with only a 2% drop in classification accuracy.

B. Quantum Machine Learning and QuantumTrain Framework

Quantum machine learning (QML) has demonstrated great potential in enhancing learning efficiency and handling complex, high-dimensional data [61]–[63]. QML takes advantage of key quantum properties, such as superposition and entanglement, to perform computations across many basis states in parallel. This parallelism opens up new possibilities for improving computational efficiency, especially when working with complex or high-dimensional data [64]–[66]. In QML model, data is commonly fed into quantum circuits through encoding schemes like gate-angle encoding, where data values control the rotation of quantum gates, or amplitude encoding, which maps information directly into the amplitudes of a quantum state [67]. Practical applications of QML have already emerged in several fields, including channel estimation [68], healthcare monitoring [69], natural language processing [70], and resource optimization [71]. However, QML is still in its early stages and faces several key challenges. One of the most pressing issues is the difficulty of encoding large datasets into quantum circuits, which is constrained by the limited number of available qubits and the shallow circuit depths allowed by

the short coherence times of current quantum hardware [72]. In addition to these encoding limitations, many QML models require quantum hardware not only during training but also at the inference stage. This reliance can lead to inefficiencies, especially in time-sensitive applications such as real-time decision-making in autonomous systems.

To address these challenges, a novel technique called QT has been proposed, which is employed to train classical neural network models. The core idea of the QT framework is to employ a quantum neural network (QNN) alongside a mapping mechanism to generate the weights of a classical neural network model. This approach offers significant parameter reduction on a polylogarithmic scale, eliminates data encoding issues, and enables inference on purely classical computers [73]. The authors in [45] explored the QT framework for image classification. By using the QT framework to train convolutional neural networks (CNNs), the model achieves a test accuracy of 80.21% on the MNIST dataset while utilizing only 10.8% of the original parameter count. Similarly, the author in [74] also leveraged the QT framework to train the CNN for the task of deepfake audio detection. The other studies [46], [52], [53], [75] have explored the potential of the QT framework in training various classical DL models, including reinforcement learning (RL), long short-term memory (LSTM), LLMs and federated learning (FL). These studies further demonstrate that the QT framework can be effectively applied across diverse classical DL architectures.

To enhance the scalability of the QT framework, the authors in [72] improved the existing mapping network by replacing the conventional multi-layer perceptron (MLP) with a tensor network-based mapping model. While this approach offers more efficient parameter representation and lowers computational demands, it also introduces a trade-off in the form of significantly increased training time. Meanwhile, the study in [51] addresses the training time problem of the QT framework. The authors proposed a novel embedding function that employs sine and cosine functions restricted to a single complete cycle, enabling the encoding of quantum state information corresponding to the QNN's output probabilities. By reducing the input dimensionality of the mapping network, this method lowers its computational load and consequently accelerates the training process of the QT framework.

III. Preliminaries on Quantum Machine Learning and Pruning Techniques

A. Quantum Machine Learning

QML refers to the application of quantum computing to machine learning tasks, utilizing quantum mechanical properties such as superposition and entanglement to achieve computational advantages [76]. By exploiting the high dimension Hilbert space, QML offers the potential to process complex data more efficiently than classical models [77], [78]. Among the various QML architectures, the parameterized quantum circuits (PQCs) have received significant attention.

A PQC consists of a sequence of quantum gates controlled by a set of tunable parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$. These parameters are optimized to minimize a task-specific cost function, often using classical optimization algorithms in a hybrid quantum-classical loop. A PQC can be mathematically represented as

$$U(\theta) = U_L(\theta_L) \ldots U_2(\theta_2) U_1(\theta_1), \tag{1}$$

where $U_L(\theta_L)$ denotes the parameterized unitary operation applied at the L-th layer of the circuit.

After applying the parameterized unitary $U(\theta)$ to an initial quantum state, which is usually set as zero state $|0\rangle^{\otimes n}$, with n is the number of qubits, the PQC outputs a quantum state $|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}$. Upon measurement in the computational basis, this state collapses to a probability vector. This probability can be used to compute expectation values or feed into classical post-processing steps. The probability of obtaining the outcome z is

$$p_\theta(z) = |\langle z|\psi(\theta)\rangle|^2, \tag{2}$$

where z denotes the possible measurement outcomes. For a quantum system of n qubits, there are 2^n possible outcomes. $|\psi(\theta)\rangle$ represents the output state of PQC before measurement. The inner product $\langle z \mid \psi(\theta)\rangle$ gives the complex probability amplitude of observing the outcome z when measuring the state $|\psi(\theta)\rangle$. Finally, the squared magnitude $|\langle z \mid \psi(\theta)\rangle|^2$ yields the actual probability of obtaining the bit string z upon measurement [79], [80].

B. Pruning

In recent years, the size of DL models has grown exponentially [81], [82], leading to increased computational cost and memory usage. These challenges hinder deployment on edge devices and embedded systems with limited on-chip resources, where memory, latency, and energy efficiency are critical constraints [44], [83]. To mitigate this issue, pruning has been emerged as one of the most effective model compression techniques, reducing redundant parameters while preserving model performance.

Pruning techniques can be broadly categorized into unstructured pruning and structured pruning. Unstructured pruning removes specific weights from the network according to their size or contributions to the loss function. Preliminary works have found that unstructured pruning can considerably minimize the CNN parameters with minimal loss of accuracy [84], [85]. Nevertheless, due to the sparse and irregular weight matrices, they are not well-suited to hardware acceleration, and are ineffectual in inference [86]. Structured pruning, on the other hand, prunes out entire architectural components (filters, channels, or layers), which forms more regular, dense matrices [87]. This regularity enables improved general-purpose hardware and DL accelerator support, making it more suitable for real-time inference on resource-constrained platforms [44], [88]. Structured pruning is often further classified into filter pruning (removing whole convolutional filters) and channel pruning (removing feature map channels) [89], [90]. While structured pruning may incur a slightly higher accuracy trade-off than unstructured methods, it offers superior hardware compatibility and practical computational speed-ups.

TABLE II
SUMMARY OF RELATED STUDIES WITH OUR PROPOSED STUDY

Tasks	Studies	Key Idea	Advantages & Limitations
HAR Accuracy Improvement	[56]	Dual-task DL with domain-independent motion features	✓ Achieves high recognition accuracy in both in-domain and cross-domain settings while improving processing efficiency ✓ Supports adaptation to environmental changes without full retraining ✗ Requires complex DL architectures with high computational overhead, making deployment on resource-constrained devices challenging
	[57]	ResNet-based spatiotemporal dual-attention learning with domain-independent gesture features	
	[58]	Graph-based few-shot learning with dual attention for CSI-based HAR	
	[59], [60]	Online and incremental learning for robust sensing in dynamic environments	
HAR Efficiency Improvement	[24]	Channel pruning for efficient Wi-Fi CSI-based sensing	✓ Reduces model complexity and parameter count while maintaining good recognition performance, enabling deployment on resource-constrained devices ✗ Primarily improves inference efficiency, with limited gains in training efficiency
	[32]	Quantization for efficient on-device Wi-Fi sensing	
	[50]	Network pruning and quantization for efficient device-free wireless sensing	
QuantumTrain Framework	[45], [51]	QT-based training for CNNs	✓ Improves training efficiency while maintaining comparable accuracy with a significantly reduced number of trainable parameters ✗ Focuses primarily on the training stage, with limited impact on inference efficiency
	[46]	QT-based training for LLMs	
	[52]	QT-based training for LSTMs	
	[53]	QT-based training for RL models	
HAR Efficiency Improvement	Our Work	Novel quantum-assisted memory-efficient training framework for HAR	✓ Significantly improves training efficiency by reducing the number of trainable parameters and producing highly sparse models, enabling lightweight on-device inference without substantial performance loss

Note: ✓ indicates advantages and ✗ indicates limitations.

IV. On-Device Wi-Fi–Based Human Activity Recognition Systems

Wi-Fi-based HAR utilizes the CSI, extracted from the communications link between a transmitter and receiver, to characterize human movements. The received Wi-Fi signal in the frequency domain can be described as

$$Y(n) = H(n) \cdot X(n) + W(n), \tag{3}$$

where $H(n), X(n)$ and $W(n)$ represent channel frequency response (CFR), transmitted signal, and additive white Gaussian noise (AWGN) in frequency domain, respectively.

In practical deployment, depending on the specific CSI extraction tool used, the receiver samples the signal in the time domain, and the CFR is subsequently estimated at the subcarrier level, capturing both amplitude attenuation and phase shifts in complex form [21], [42]. The CFR of each subcarrier can be expressed as

$$H_k = |H_k| \, e^{j \angle H_k}, \tag{4}$$

where $|H_k|$ and $\angle H_k$ denote the amplitude and phase of the k-th subcarrier, respectively.

In this study, we investigate a specific deployment scenario for Wi-Fi-based HAR system deployment on devices, where computational resources, energy availability, and memory are significantly constrained. As shown in Fig. 1, an on-device Wi-Fi HAR system is typically structured into three primary stages: dataset preparation, training, and inference.

During the dataset preparation stage (Fig. 1(a)), CSI data is collected in an indoor environment where participants perform a predefined set of activities. These human motions induce characteristic fluctuations in the wireless channel, which are captured as time-series CSI measurements.

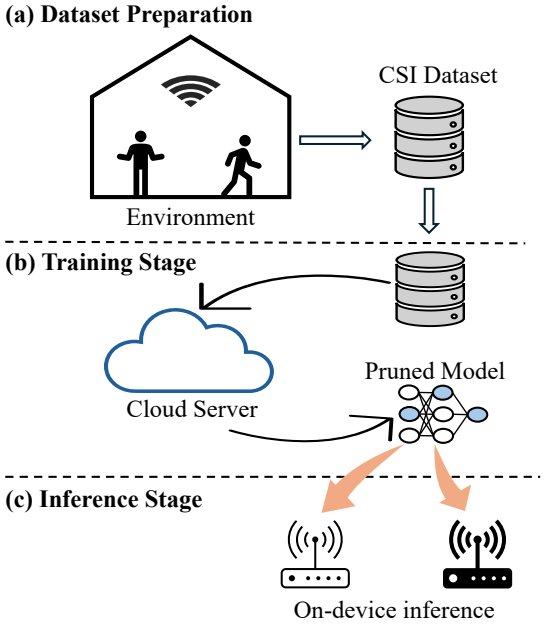


Fig. 1. Overview of the on-device Wi-Fi-based human activity recognition (HAR) system. (a) Dataset preparation, where CSI measurements are collected from human activities in an indoor Wi-Fi environment. (b) Training stage, where the CSI dataset is transferred to a cloud server for model training. (c) Inference stage, where the trained model is deployed on devices to perform real-time activity recognition.

Once the collection is complete, the dataset is transferred to a cloud computing server (Fig. 1(b)) that provides computational resources for signal processing and model training.

Since our objective is on-device deployment, the DL model undergoes compression techniques such as pruning to reduce its size before being deployed on the target devices. Finally,

in the inference stage (Fig. 1(c)), the compressed model is deployed onto the target devices (e.g., routers or IoT nodes) to classify human activities in real-time.

A. Signal Processing

The raw CSI data is a complex-valued tensor H. We use only the amplitude part $|H_k|$ as the model input, since the raw phase is often affected by hardware-induced sampling frequency offsets and carrier frequency offsets, which can reduce its reliability for robust HAR [21]. The two datasets come in different shapes, so we use a small dataset-specific initial block (Fig. 2(b)) and (Fig. 2(c)) to bring them into a common format before passing them through the shared ResNet-18 backbone:

- **UT-HAR**: Following SenseFi [21], each sample is reshaped into a single-channel 2D matrix $\mathbf{x} \in \mathbb{R}^{1\times 250\times 90}$, where 250 is the number of time snapshots and 90 comes from 3 receive antennas $\times$ 30 subcarriers. Each sample is then scaled to the range $[0, 1]$ using a simple per-sample min–max normalization: $\tilde{\mathbf{x}} = (\mathbf{x} - \min(\mathbf{x}))/(\max(\mathbf{x}) - \min(\mathbf{x}))$.
- **Widar3.0**: Following SenseFi [21], each sample is a tensor $\mathbf{x} \in \mathbb{R}^{22\times 20\times 20}$, where 22 is the time dimension and 20×20 is the velocity grid. Each sample is standardized using the dataset mean $\mu = 0.0025$ and standard deviation $\sigma = 0.0119$, computed once on the Widar3.0 training set: $\tilde{\mathbf{x}} = (\mathbf{x} - \mu)/\sigma$.

We use different normalization schemes for the two datasets because each one follows the convention of its original release [48], [49]. Moreover, this signal processing pipeline is consistent with the SenseFi benchmark [21], so our results can be directly compared with earlier HAR studies.

B. Deep Learning Model

While various DL architectures have been investigated for Wi-Fi-based HAR, including MLPs, CNNs, LSTM, Vision Transformers (ViTs), and Residual Networks (ResNets) [21], recent studies indicate that ResNet-based architectures provide strong robustness and high classification accuracy [21]. In particular, benchmarking results across multiple public datasets show that ResNet-18 consistently achieves the highest classification accuracy on both the UT-HAR and Widar3.0 datasets [21]. Therefore, we adopt ResNet-18 as the representative model for Wi-Fi-based HAR and use it as the case study in this work.

The architecture of ResNet-18 is shown in Fig. 2. The ResNet-18 model has about 11.6 million trainable parameters, which enables strong classification performance, but also makes it relatively expensive to train and deploy. To accommodate the distinct input tensor shapes of the UT-HAR and Widar3.0 datasets, specific initial blocks (as shown in Fig. 2(b) and Fig. 2(c)) are employed to unify the input dimensions. This enables the use of a common core network architecture across the two datasets.

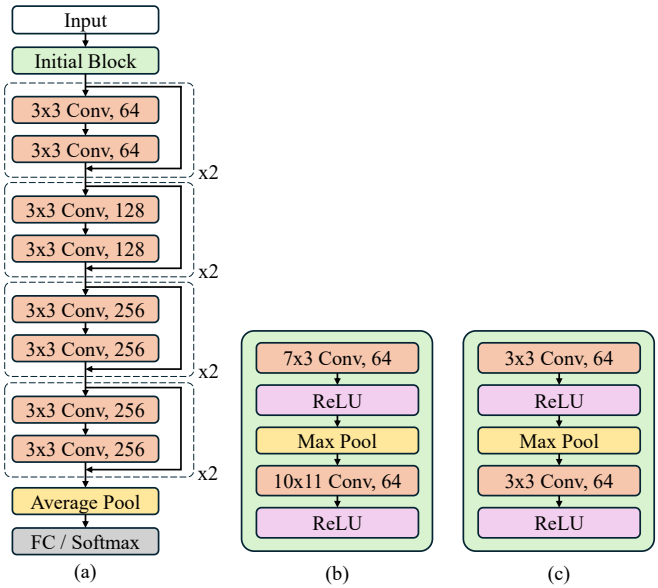


Fig. 2. Architecture of the ResNet-18 model. (a) Overall architecture. (b) Initial block used in UT-HAR. (c) Initial block used in Widar3.0.

V. Quantum-Assisted Memory-Efficient Training Framework

A. Framework Overview

As shown in Fig. 3(a), the Q-MET framework employs a quantum parameter generator (QPG) to dynamically generate the parameters for a classical ResNet-18 model. The QPG comprises three main components: a PQC, a sinusoidal embedding layer, and a classical mapping network.

In the PQC, the U_3 gate is employed to manipulate qubit states, while the CU_3 gate establishes entanglement between qubits [53]. The U_3 and CU_3 gates are defined as follows

$$U_3(\theta, \varphi, \lambda) = \begin{bmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\varphi}\sin(\theta/2) & e^{i(\varphi+\lambda)}\cos(\theta/2) \end{bmatrix}, \quad (5)$$

$$CU_3(\tilde{\theta}, \tilde{\varphi}, \tilde{\lambda})_{q_0,q_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\tilde{\theta}}{2}\right) & 0 & -e^{i\tilde{\lambda}}\sin\left(\frac{\tilde{\theta}}{2}\right) \\ 0 & 0 & 1 & 0 \\ 0 & e^{i\tilde{\varphi}}\sin\left(\frac{\tilde{\theta}}{2}\right) & 0 & e^{i(\tilde{\varphi}+\tilde{\lambda})}\cos\left(\frac{\tilde{\theta}}{2}\right) \end{bmatrix}, \quad (6)$$

where $(\theta, \varphi, \lambda)$ and $(\tilde{\theta}, \tilde{\varphi}, \tilde{\lambda})$ are real-valued parameters that define the rotations applied by the U_3 and CU_3 gates, q_0 and q_1 denote the control and target qubits, respectively.

The CU_3 is designed with a circular layout, which enables for constructing multi-qubits interactions within the PQC. The PQC can be expressed as

$$|\psi(\theta_{PQC})\rangle = U(\theta_{PQC})|0\rangle^{\otimes N_q}, \quad (7)$$

where N_q denotes the number of qubits, and total number of trainable parameters in the PQC is $\theta_{PQC} = 6N_q$, as each U_3 and CU_3 gate has three tunable parameters. The

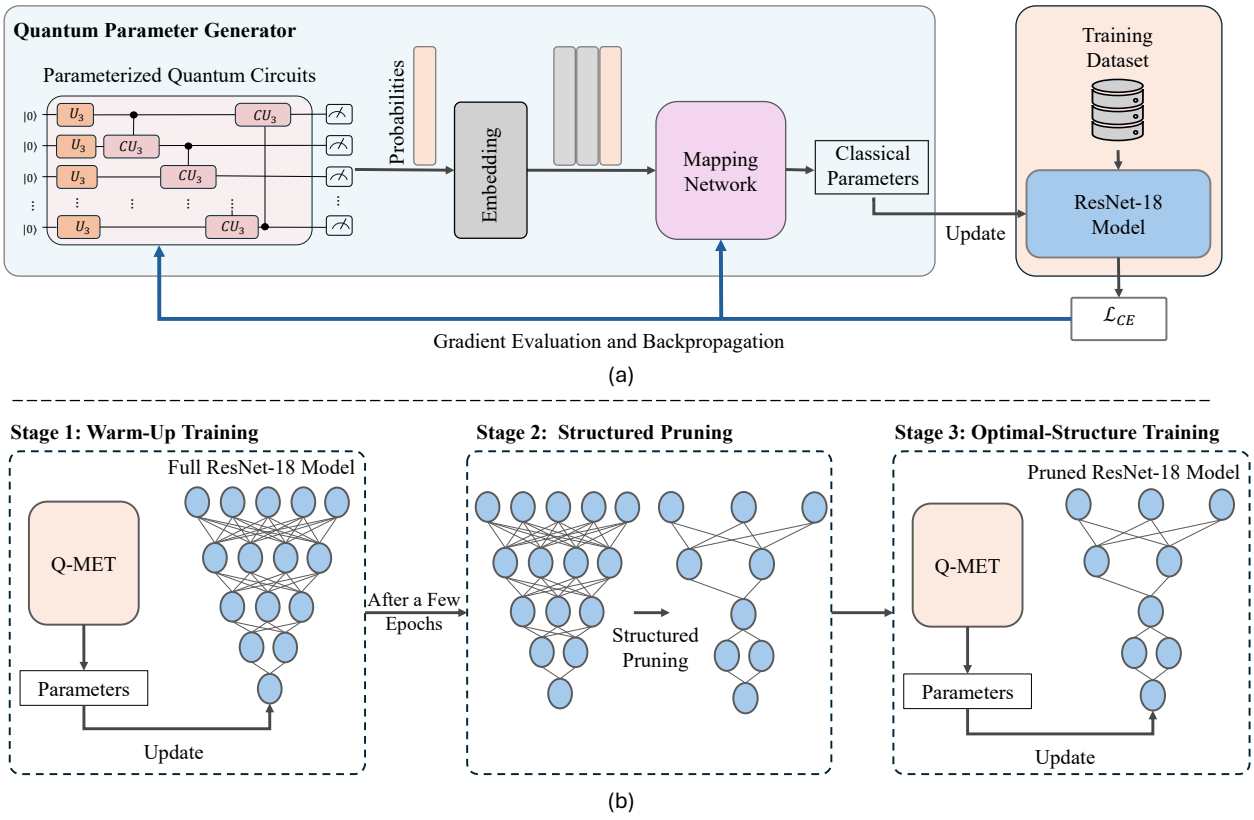


Fig. 3. Proposed quantum-assisted memory-efficient training framework. (a) Overall architecture where the Q-MET generates parameters for ResNet-18. (b) Training workflow of Q-MET: warm-up training, LAMP-based pruning, and optimal-structure training.

measurement probabilities of the PQC output can be expressed as a probability vector as follows

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

where $|\phi_k\rangle \in \{0,1\}^{N_q}$ represents the k-th computational basis state, $k \in \{1, 2, \ldots, \mathcal{P} = 2^{N_q}\}$ indexes the computational basis state, and $|\langle \phi_k | \psi(\theta_{PQC}) \rangle|^2$ is the probability of obtaining the outcome $|\phi_k\rangle$ upon measurement.

These probabilities are then further processed by an embedding function that adds information to represent the uniqueness of computational basis state. In this study, we employ sinusoidal embedding, as proposed in [51], because it offers a compact and uniquely identifiable representation of computational basis states, while significantly reducing the embedding dimensionality from exponential to constant size, thereby improving scalability and computational efficiency. The sinusoidal embedding is defined as follows

$$\mathcal{S}(\Psi) = [S_0, S_1, \Psi], \tag{9}$$

where S_0 and S_1 represent embedding vectors that serve as compact and distinctive identifiers for each set of measurement probabilities. As a result, the triplet $[S_0, S_1, \Psi]$ effectively encodes the identity of the corresponding computational basis

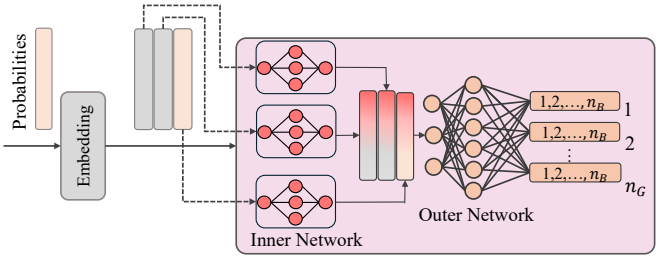


Fig. 4. Overview of mapping network in the proposed quantum-assisted memory-efficient training framework.

state linked to each probability. The explicit expansion of Eq. (9) is presented as follows

$$\mathcal{S}(\Psi) = \begin{bmatrix} S_{0,0}, & S_{1,0}, & |\langle \phi_1 | \psi(\theta_{PQC}) \rangle|^2 \\ \vdots & \vdots & \\ S_{0,k_e}, & S_{1,k_e}, & |\langle \phi_k | \psi(\theta_{PQC}) \rangle|^2 \\ \vdots & \vdots & \\ S_{0,\mathcal{P}-1}, & S_{1,\mathcal{P}-1}, & |\langle \phi_{\mathcal{P}} | \psi(\theta_{PQC}) \rangle|^2 \end{bmatrix}, \tag{10}$$

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

$$S_{\iota,k_e} = \sin\left(\frac{k_e}{\mathcal{P}} 2\pi + \frac{\iota\pi}{2}\right), \tag{11}$$

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

TABLE III
DETAILED ARCHITECTURE OF MAPPING NETWORK MODEL

Component	Layer Type	Parameters
Inner Function†	FC ($1 \to 16$) + BN	64
	FC ($16 \to 1$)	17
Outer Function	FC ($3 \to 32$) + BN	192
	FC ($32 \to 64$)	2112
	FC ($64 \to 5$)	325
	FC ($5 \to n_B$)	$6 \times n_B$
Total	-	$2872 + 6 \times n_B$

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

In the Q-MET framework, these embedded probability vectors are then mapped to the parameters of the ResNet-18 model using a classical neural network, referred to as the mapping network, denoted by M with tunable parameters θ_m. As depicted in Fig. 4, the mapping network comprises an inner network and an outer network. Detailed layer specifications are provided in Table III. The structure of the mapping network follows the sinusoidal-embedding mapping design introduced in our prior QT work [51], where an inner function is applied independently to each input variable and is followed by an outer function. Building on that design, the intermediate layer widths in Table III were enlarged and selected through a small search over candidate dimensions, guided by the trade-off between classification accuracy and the size of the mapping network. Since the mapping network accounts for the dominant share of the trainable parameters in the Q-MET framework, the smallest widths that retained competitive accuracy were chosen to keep the overall trainable-parameter count as low as possible. The output width n_B is dataset-dependent and is set by (15) so that the total number of generated parameters matches the size of the target model, while the remaining widths are fixed across both datasets. The output of mapping network is a matrix parameter, which can be modeled as

$$\mathbf{\Theta}_m = M_{\theta_m}(\mathcal{S}(\Psi)), \tag{12}$$

where $\mathbf{\Theta}_m \in \mathbb{R}^{n_G \cdot n_B}$ denotes the matrix of classical parameters, with n_G and n_B representing the number of parameter groups and the number of parameters in each group, respectively.

The output matrix is then reshaped into a 1D vector of the length $n_G \cdot n_B$, denoted as $\overrightarrow{\theta}_{ResNet-18}$, which serves as the classical parameter set of ResNet-18. It is worth noting that the QPG acts purely as a weight generator and does not take CSI data as input. This makes Q-MET applicable to any classical DL architecture regardless of the input modality.

B. Structured Pruning

To address inference inefficiency, Q-MET employs a structured channel pruning strategy guided by layer-adaptive magnitude-based pruning (LAMP) [91] scores, where channel importance is derived from the LAMP scores of their associated weights. Compared to uniform or global magnitude pruning, LAMP better preserves sensitive layers, resulting in improved accuracy retention under high sparsity, while maintaining significant reductions in model size and computational cost. It is worth noting that Q-MET is compatible with any pruning technique, from unstructured to structured methods. Structured pruning is preferred here because it produces regular, dense matrices that are well-suited to hardware acceleration on resource-constrained platforms, whereas unstructured pruning yields irregular sparse matrices that are ineffectual for practical inference speedup [86]. Among structured criteria, LAMP is adopted for its layer-adaptive importance scoring, which better preserves sensitive layers under high sparsity compared to uniform magnitude pruning [91].

Consider a neural network in which each layer is associated with a weight tensor corresponding to a fully connected or convolutional layer. To provide a unified formulation across different layer types, each weight tensor W is reshaped into a one-dimensional vector (i.e., vectorized). Without loss of generality, the elements of this vector are sorted and indexed in ascending order of magnitude, such that

$$|W[u]| \leq |W[v]| \quad \text{for } u < v, \tag{13}$$

where $W[u]$ denotes the u-th element of the flattened weight tensor W, and u and v index the entries of this vector after sorting in ascending order of magnitude.

Based on this ordering, the LAMP score assigned to the weight indexed by u is defined as [91]

$$\text{score}(u; W) = \frac{(W[u])^2}{\sum_{v \geq u} (W[v])^2}. \tag{14}$$

This score measures the relative importance of a weight with respect to the remaining higher-magnitude weights within the same layer, as the denominator represents the cumulative squared magnitude of all weights ranked at or above index u.

To extend LAMP to structured pruning, a channel-level importance score is computed by aggregating the LAMP scores of all weights within each channel. After computing the channel scores, all channel scores across the network are concatenated into a single global score vector. Given a target pruning ratio p, channels with the smallest scores are globally removed until the desired sparsity level is reached.

C. Training Flow for Quantum-Assisted Memory-Efficient Training Framework

As shown in Fig. 3(b), the training process of Q-MET consists of three stages: warm-up training, structured pruning, and optimal-structure training.

In the warm-up training stage, the full model is trained for a few epochs. The goal is not convergence, but to establish a weight distribution sufficient for identifying redundant connections. This is followed by structured pruning using LAMP. The LAMP scores are evaluated to identify and remove less important filters, resulting in a sparse model architecture by keeping only the most informative components. Finally, the pruned model, now considered a compact structure, is further trained using Q-MET, which produces parameters only for the active channels. Therefore, the number of parameters generated is significantly lower compared to the full model, resulting in higher memory efficiency during training.

1) Quantum Parameter Generator Procedure: To construct the QPG, we first determine the number of qubits required to construct the PQC. Based on the number of qubits and the total number of parameters in the classical ResNet-18 model, we then design the mapping network by specifying the parameters n_B and n_G, which are given by

$$n_G = 2^{N_q}, \quad n_B = \left\lceil \frac{\mathcal{C}_{\text{ResNet-18}}}{2^{N_q}} \right\rceil. \tag{15}$$

Once the QPG is initialized, it is used to dynamically generate the parameters of the ResNet-18 model during training. At each training iteration, the QPG produces a parameter set $\overrightarrow{\theta}_{\text{ResNet-18}}$ corresponding only to the active (unpruned) components of the network. These parameters are then injected into the ResNet-18 architecture using PyTorch's functional operators (e.g., `F.conv2d`, `F.linear`), which accept `weight` and `bias` as arguments and thus enable on-the-fly parameter injection into the corresponding layers. After the parameter update, the model processes a training input x and produces a prediction $\hat{y}$ via the forward pass

$$\hat{y} = Z(x; \overrightarrow{\theta}_{ResNet-18}), \tag{16}$$

where $Z(\cdot)$ denotes the forward function of ResNet-18 with the parameters generated by Q-MET.

2) Loss Function and Gradient Estimation: For multi-class classification tasks, the training loss is computed using the cross-entropy (CE) loss function

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N_{\text{Batch}}} \sum_{i=1}^{N_{\text{Batch}}} \sum_{c=1}^{N_c} y_{i,c} \log(\hat{y}_{i,c}), \tag{17}$$

where $y_{i,c}$ denotes the ground-truth label of the i-th sample for class c, and $\hat{y}_{i,c}$ represents the predicted probability assigned to class c by the model. N_{Batch} is the batch size and N_c denotes the number of output classes.

Since the parameters of ResNet-18 are generated by QPG, which consists of both the PQC with parameters θ_{PQC} and the classical mapping network with parameters θ_m, the loss gradient needs to be propagated through the entire QPG pipeline [45]. By applying the chain rule, the gradient of the loss with respect to the joint QPG parameters is computed as [45]

$$\nabla_{(\theta_{\text{PQC}}, \theta_m)} \mathcal{L}_{\text{CE}} = \left(\frac{\partial \theta_{\text{ResNet-18}}}{\partial (\theta_{\text{PQC}}, \theta_m)} \right)^{\top} \nabla_{\theta_{\text{ResNet-18}}} \mathcal{L}_{\text{CE}}, \tag{18}$$

where $\partial \theta_{\text{ResNet-18}} / \partial (\theta_{\text{PQC}}, \theta_m)$ denotes the Jacobian matrix, which quantifies the sensitivity of the generated ResNet-18 parameters with respect to the PQC and mapping network parameters.

The parameters of the PQC and the mapping network are updated via gradient descent as follows

$$\theta_{PQC}^{(t+1)},\ \theta_m^{(t+1)} = \theta_{PQC}^{(t)},\ \theta_m^{(t)} - \eta \nabla_{(\theta_{PQC}, \theta_m)} \mathcal{L}_{\text{CE}}, \tag{19}$$

where η represents the learning rate.

D. Parameter Efficiency Analysis

During training, the Q-MET framework dynamically generates the parameters required for the ResNet-18 model, thereby eliminating the need to train and update the massive number of parameters in ResNet-18. Instead, the trainable parameters are restricted to the compact QPG, which significantly reduces memory overhead and enables efficient training in resource-constrained environments. This reduction is particularly important, as memory consumption during training arises not only from model weights, but also from optimizer states, activation tensors, and gradient buffers. By minimizing the trainable parameter count, Q-MET significantly lowers the memory overhead for gradients and optimizer states, enabling efficient training in resource-constrained environments.

This subsection quantifies the efficiency of Q-MET in terms of the number of trainable parameters. Let $\mathcal{C}_{\text{ResNet-18}}$ denote the total number of parameters in the baseline ResNet-18 model. In contrast, the parameter count of Q-MET consists of the combined contributions from the PQC and the mapping network, which can be modeled as

$$C_{\text{Q-MET}} = \underbrace{6N_q}_{\text{PQC}} + \underbrace{(2872 + 6n_B)}_{\text{Mapping Network}}, \tag{20}$$

When substituting (15), the following results are obtained

$$\mathcal{C}_{\text{Q-MET}} = 6N_q + \left\lceil \frac{6\mathcal{C}_{\text{ResNet-18}}}{2^{N_q}} \right\rceil + 2872. \tag{21}$$

The parameter efficiency gain is then defined as

$$\Delta\mathcal{C}(\%) = \left(1 - \frac{\mathcal{C}_{\text{Q-MET}}}{\mathcal{C}_{\text{ResNet-18}}}\right) \times 100, \tag{22}$$

Substituting (21) yields

$$\Delta\mathcal{C}(\%) = \left(1 - \frac{6N_q}{\mathcal{C}_{\text{ResNet-18}}} - \frac{6}{2^{N_q}} - \frac{2872}{\mathcal{C}_{\text{ResNet-18}}}\right) \times 100. \tag{23}$$

In the Q-MET framework, N_q is significantly smaller than $\mathcal{C}_{\text{ResNet-18}}$ (approximately 11.6×10^6), both terms $6N_q/\mathcal{C}_{\text{ResNet-18}}$ and $2872/\mathcal{C}_{\text{ResNet-18}}$ become negligible. Therefore, (23) can be approximated as

$$\Delta\mathcal{C}_{\text{approx}}(\%) \approx \left(1 - \frac{6}{2^{N_q}}\right) \times 100. \tag{24}$$

From (24), the relationship between the number of qubits and the corresponding parameter efficiency gain is summarized in Table IV. When $N_q = 2$, the negative gain (-50%) indicates that Q-MET requires more parameters than the baseline ResNet-18 model, and is therefore less efficient at very low qubit counts, but begins to outperform the baseline when $N_q \geq 3$, where the gain is approximately 25%.

As N_q increases, the efficiency improves rapidly. When $N_q = 4$, the gain rises to 62.5%, increasing to 81.25% at $N_q = 5$, reaching 90.63% at $N_q = 6$. These results demonstrate that more than 90% parameter efficiency can be achieved using only six qubits. This highlights the potential of Q-MET to reduce model complexity in a highly efficient manner with relatively low quantum resource requirements.

TABLE IV
APPROXIMATE PARAMETER EFFICIENCY GAINS ACHIEVED BY Q-MET ACROSS DIFFERENT QUBIT COUNTS

Number of Qubits (N_q)	$\Delta\mathcal{C}_{\textbf{approx}}(\%)$
2	-50.00
3	25.00
4	62.50
5	81.25
6	90.63
7	95.31
8	97.66
9	98.83
10	99.41

TABLE V
SUMMARY OF DATASETS

Datasets	**UT-HAR**	**Widar3.0**
Hardware Platform	Intel 5300 NIC	Intel 5300 NIC
Number of Classes	7	22
Sample Dimensionality	(1,250,90)	(22,20,20)
Training samples	3,977	34,926
Testing samples	996	8,726

Beyond six qubits, however, the additional gains become smaller. For example, increasing N_q from 6 to 10 yields an additional gain of only 8.78%, reaching a maximum of 99.41%. This suggests a point of diminishing returns, where most of the parameter savings are achieved early, and further increases in qubit count provide limited improvements.

E. Computational Complexity Analysis

The per-iteration training complexity of Q-MET consists of three main components: the PQC, the mapping network, and the pruned ResNet-18. When the PQC is simulated classically and the full probability vector is evaluated, its cost scales as $\mathcal{O}(2^{N_q})$. The mapping network generates $n_G n_B$ parameters and therefore has complexity $\mathcal{O}(n_G n_B)$. For the pruned ResNet-18, let $F_{\text{ResNet-18}}$ denote the training cost of the full model. At sparsity level s, the cost is approximated as $\mathcal{O}((1-s)F_{\text{ResNet-18}})$. Thus, the total per-iteration complexity of Q-MET is

$$\mathcal{O}_{\text{Q-MET}} = \mathcal{O}(2^{N_q}) + \mathcal{O}(n_G n_B) + \mathcal{O}((1-s)F_{\text{ResNet-18}}). \quad (25)$$

In comparison, standard backpropagation-based training of the full ResNet-18 has complexity $\mathcal{O}(F_{\text{ResNet-18}})$. Therefore, Q-MET introduces additional parameter-generation overhead, which is consistent with the empirical training-time increase of 18%–36% observed in Fig. 5.

The main benefit of Q-MET is memory efficiency rather than reduced computation. Standard training stores weights, gradients, and optimizer states for all ResNet-18 parameters, resulting in model-state memory scaling with $\mathcal{O}(C_{\text{ResNet-18}})$. In contrast, Q-MET maintains trainable states only for the compact QPG, reducing this component to $\mathcal{O}(C_{\text{Q-MET}})$, where $C_{\text{Q-MET}} \ll C_{\text{ResNet-18}}$.

VI. EXPERIMENTAL EVALUATION

A. Dataset

The experiments were conducted using two public datasets, which are UT-HAR [49] and Widar3.0 [48]. Table V summarizes their specifications. Both datasets contain CSI data collected using the Intel 5300 Wi-Fi network interface card under real-world conditions.

The UT-HAR dataset contains 7 different human motions, namely *lie down, fall, walk, pick up, run, sit down, stand up*. These activities cover a range of basic human motions and are commonly used to evaluate the performance of CSI-based HAR models [21], [49].

In contrast, the Widar3.0 dataset is a large-scale benchmark focused on fine-grained gesture recognition. These classes include common gesture primitives such as *push & pull, sweep, clap, and slide*, as well as 18 drawing gestures [21], [48]. The Widar3.0 dataset is widely used to assess the scalability, discriminative capabilities, and cross-domain generalization of CSI-based sensing systems [21].

B. Training Hyperparameter Configuration

Training was conducted on a Dell Precision 3680 workstation equipped with an Intel® Core™ i7-14700 processor, 64 GB of RAM, a 2 TB SSD, and an NVIDIA GeForce RTX 4090 GPU. We used the Adam optimizer with an initial learning rate of 0.001 and a batch size of 128. Training ran for up to 500 epochs, with an early stopping mechanism triggered if validation loss did not improve for 30 consecutive epochs.

C. Evaluation Metrics

1) Accuracy: Accuracy is a key metric for assessing the classification performance of DL-based HAR models. It represents the ratio of correctly predicted instances to the total number of instances in the test set. The accuracy can be expressed as

$$Acc = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}. \quad (26)$$

To evaluate the impact of Q-MET relative to the traditional approach, we define the accuracy loss as

$$Acc_{Loss} = Acc_{baseline} - Acc_{\text{Q-MET}}, \quad (27)$$

where $Acc_{\text{Q-MET}}$ and $Acc_{baseline}$ denote the accuracies of the ResNet-18 model trained using the Q-MET framework and the classical approach, respectively.

2) Parameter Efficiency: We use the number of trainable parameters as a primary metric to evaluate efficiency, as parameter count is the canonical and widely accepted efficiency measure in the parameter-efficient training literature such as LoRA [92], and the QT family [45], [46], [51]–[53].

For a DL model with $\mathcal{C}$ trainable parameters, the model-state component of training memory, comprising the parameters, gradients, and the two Adam moment estimates stored in FP32, can be expressed as [34]

$$M_{\text{model}}^{\text{train}} = M_p + M_{\text{grads}} + M_{\text{opt}}, \quad (28)$$

where $M_p = \mathcal{CS}$ denotes the memory required to store the model parameters, $M_{\text{grads}} = \mathcal{CS}$ corresponds to the memory for storing gradients, and $M_{\text{opt}} = 2\mathcal{CS}$ represents the memory required by the optimizer states when using the Adam optimizer. Here, $\mathcal{S}$ denotes the number of bytes per parameter (e.g., $\mathcal{S} = 4$ for FP32).

Therefore, (28) can be rewritten as

$$M_{\text{model}}^{\text{train}} = 4\mathcal{CS}. \tag{29}$$

Under FP32 precision, this corresponds to $M_{\text{model}}^{\text{train}} = 16\mathcal{C}$ bytes. Concretely, the model-state memory of the baseline ResNet-18 is approximately 177 MB on UT-HAR and 171 MB on Widar3.0, whereas Q-MET reduces it to about 8.4 MB (QT-7) and 17 MB (QT-6), respectively, corresponding to a reduction of roughly 95% and 90%. These figures follow directly from the trainable-parameter counts and the relation $M_{\text{model}}^{\text{train}} = 16\mathcal{C}$. We emphasize that this quantifies the model-state component of training memory (parameters, gradients, and optimizer states). The activation and buffer memory, which is determined by the network forward pass and batch size, is orthogonal to and not the target of the proposed framework, though structured pruning incidentally reduces it.

During inference, gradient and optimizer states are not required, and the memory footprint is primarily determined by the model parameters. As a result, inference memory remains strongly correlated with the parameter count, further justifying the use of parameter efficiency as a unifying metric for evaluating both training and inference memory requirements.

To quantify the efficiency of the proposed Q-MET framework, we adopt the notion of parameter efficiency, as defined in (22). This metric measures the percentage reduction in the number of trainable parameters achieved by Q-MET relative to the classical training approach.

3) Sparsity: Sparsity is a key metric for assessing the effectiveness of structured pruning [93]. It represents the percentage of the model that has been pruned and can be defined as

$$\text{Sparsity}(\%) = \frac{\text{Number of pruned parameters}}{\text{Total number of parameters}} \times 100. \tag{30}$$

A higher sparsity level indicates a more aggressively pruned model, which typically results in reduced model size and faster inference times [93].

4) Training Time: To evaluate the computational efficiency of the QT framework, we define the average training time per epoch, T_e (in seconds), as

$$T_e = \frac{\text{Total training time}}{\text{Number of epochs}}, \tag{31}$$

where the total training time is measured until model convergence.

D. Experimental Results

In this subsection, we first evaluate the effectiveness of the QT framework in training the classical ResNet-18 model to determine the optimal QT configuration. We then systematically analyze the performance of the proposed Q-MET framework under various training settings, including different numbers of Warm-Up epochs and pruning ratios. Finally, we discuss the observed performance-efficiency trade-offs and provide insights of Q-MET for resource-constrained HAR deployments.

TABLE VI
BASELINE PERFORMANCE OF RESNET-18 ON UT-HAR AND WIDAR3.0

Model	Datasets	Parameters	Acc (%)		
			Highest	Lowest	Average
ResNet-18	UT-HAR	11,622,599	98.80	97.60	98.08 ± 0.44
	Widar3.0	11,227,798	72.22	69.71	71.29 ± 1.02

1) Performance of Classical ResNet-18 Model and QT Framework: We first examine the performance of the ResNet-18 model that is trained via conventional training method for both UT-HAR and Widar3.0 datasets, which serves as the baseline for our study. Table VI reports the number of parameters, highest and lowest accuracy across five runs, average accuracy with standard deviation.

ResNet-18 achieves strong classification performance on the UT-HAR dataset, yielding an average classification accuracy of 98.08 ± 0.44%. In contrast, performance on the Widar3.0 dataset is lower, with an average accuracy of 71.29 ± 1.02%. These results are consistent with those reported in [21]. The lower classification accuracy on the Widar3.0 dataset is primarily attributed to the greater complexity of the dataset, as the Widar3.0 dataset includes 22 gesture classes, compared to only 7 activity classes in the UT-HAR dataset. The larger number of classes increases the difficulty of the classification task, making it more challenging for the model to distinguish between similar gestures. Additionally, the number of parameters is slightly different between the two settings, with 11.6 million in UT-HAR and 11.2 million in Widar3.0. This difference is due to small adjustments in the initial block made for each dataset, as described in Section IV-B.

Subsequently, we evaluate the effectiveness of the proposed QT framework for training the classical ResNet-18 models. As presented in Section IV-D, the QT framework starts attaining efficiency when the qubits count is more than three. To analyze the performance of the framework under various levels of compression, we conducted experiment with qubit settings ranging from 3 to 10 qubits (QT-3 to QT-10). As shown in Table VII, the actual parameter efficiency achieved by the QT framework ($\Delta\mathcal{C}(\%)$) closely aligns with the approximated values ($\Delta\mathcal{C}_{\text{approx}}(\%)$), thereby validating the theoretical analysis presented in Section V-D.

Table VII presents the performance results of the QT framework on the UT-HAR and Widar3.0 datasets. For UT-HAR, we observe that all QT variants consistently improve or match baseline accuracy, even under aggressive parameter compression. For example, QT-5 and QT-7 not only reduce the parameter count by approximately 81.22% and 95.29%, but also yield improved average accuracy (99.20 ± 0.22% and 99.08 ± 0.24%). These results suggest that the QT framework can successfully reduce parameter complexity without sacrificing model performance. In contrast, the performance on the more complex Widar3.0 dataset follows a different curve. QT-3 to QT-5 perform comparably or slightly better

TABLE VII
PERFORMANCE OF RESNET-18 TRAINED WITH QT USING DIFFERENT NUMBERS OF QUBITS

QT Variants	$\Delta\mathcal{C}_{\text{approx}}(\%)$	$\Delta\mathcal{C}(\%)$	UT-HAR Acc (%)			Widar3.0 Acc (%)		
			Highest	Lowest	Average	Highest	Lowest	Average
QT-3	25.00	24.98	99.20	99.00	99.12 ± 0.10	73.02	72.43	72.78 ± 0.22
QT-4	62.50	62.48	99.20	98.80	99.08 ± 0.16	73.14	71.82	72.35 ± 0.47
QT-5	81.25	81.22	99.40	98.80	99.20 ± 0.22	71.74	71.05	71.40 ± 0.23
QT-6	90.63	90.60	99.20	98.60	98.80 ± 0.22	72.16	69.02	70.61 ± 0.45
QT-7	95.31	95.29	99.40	98.80	99.08 ± 0.24	70.40	66.82	68.76 ± 1.31
QT-8	97.66	97.63	99.00	97.40	98.56 ± 0.62	67.92	65.30	66.40 ± 1.94
QT-9	98.83	98.80	99.20	98.60	98.72 ± 0.27	68.70	63.55	66.28 ± 1.97
QT-10	99.41	99.39	99.00	97.00	98.08 ± 0.70	67.40	61.82	65.84 ± 2.03

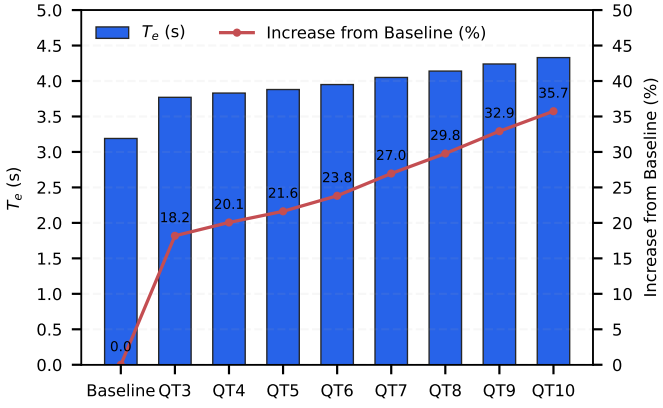


Fig. 5. Training time per epoch of the baseline and QT variants.

than the baseline, with QT-3 reaching 72.78 ± 0.22% and QT-5 reaching 71.40 ± 0.23%. Beyond QT-6, however, the performance starts to degrade, with QT-9 and QT-10 dropping to 66.28 ± 1.94% and 65.84 ± 2.03%, respectively. Furthermore, the variance in accuracy increases from 0.22 for QT-3 to 2.03 for QT-10, indicating a less stable model. This increasing variance, together with the drop in accuracy, hints that excessive compression may limit the performance of the QT framework. As the number of qubits increases, the dimensionality of the mapping network shrinks, thereby limiting its ability to learn good PQC probabilities and risking underfitting.

Regarding training efficiency, Fig. 5 shows that the baseline model on the UT-HAR dataset requires 3.2 seconds per epoch. In comparison, the QT framework exhibits a longer training time per epoch across all settings. Specifically, when increasing the number of qubits from 3 to 10, the training time increases gradually. For instance, QT-3 has an 18.2% overhead compared to the baseline, and QT-10 has a 35.7% increase. Although this introduces extra training cost, the gain in parameter efficiency makes it especially suitable for resource-constrained devices. Moreover, this additional training cost is incurred only during offline training and does not impact inference efficiency.

These outcomes provide a comprehensive evaluation of the QT framework on classification performance, and training cost. The main advantage of QT is that it can vastly reduce the parameters without sacrificing performance, (and in some cases increases effectiveness) in classification. A smaller qubit configuration (e.g., QT-3 to QT-5) exhibits moderate compression with low accuracy loss, while a larger number of qubit configurations (QT-8 to QT-10) achieves more compression but at the cost of reduced accuracy, particularly on more complex datasets. For the UT-HAR dataset, it is found that QT-7 has the best trade-off, achieving a high classification accuracy of 99.08 ± 0.24% with a parameter reduction score of over 95% and acceptable training overhead. In contrast, for the more complex Widar3.0 dataset, QT-6 has the best trade-off, maintaining an accuracy of 70.61 ± 0.45% while reducing parameters by more than 90%.

TABLE VIII
COMPARISON OF QT-7 AGAINST CLASSICALLY TRAINED LIGHTWEIGHT RESNET-18 AND A STATIC HYPERNETWORK

Model	UT-HAR		Widar3.0	
	Parameters	Acc (%)	Parameters	Acc (%)
Lightweight ResNet-18	529,295	97.60 ± 0.42	525,958	62.60 ± 0.46
Hypernetwork	547,323	98.60 ± 0.75	547,244	63.50 ± 7.39
QT-7	547,424	99.08 ± 0.24	528,830	68.76 ± 1.31

2) Comparison with Lightweight ResNet-18: To assess whether Q-MET's gains stem from the training framework rather than from architectural choice, we compare QT-7 against a classically trained lightweight ResNet-18. Specifically, we reduce the number of channels in every convolutional layer of ResNet-18 by a width multiplier of 0.22, meaning each layer retains only 22% of its original channel count. This yields a slim ResNet-18 with approximately 529k parameters, closely matched to the QT-7 parameter budget, ensuring a fair parameter-controlled comparison. As reported in Table VIII, the lightweight ResNet-18 achieves 97.60 ± 0.42% on UT-HAR and 62.60 ± 0.46% on Widar3.0, whereas QT-7 achieves 99.08 ± 0.24% and 68.76 ± 1.31%, respectively. The accuracy gap widens from 1.48% on UT-HAR to 6.16% on Widar3.0, demonstrating that Q-MET's advantage is attributable to the training framework and not merely to parameter budget reduction.

3) Comparison with classical hypernetworks: To assess whether Q-MET's gains stem from the proposed quantum-assisted framework rather than from classical parameter generation, we compare QT-7 against a static hypernetwork [94], [95]. This baseline is chosen for two reasons: it is a well-established hypernetwork formulation, and it was originally designed to generate the convolutional kernels of CNNs, making it directly compatible with our ResNet-18 model. The hypernetwork is configured to generate all parameters of the ResNet-18 model, and its embedding size is tuned so that its

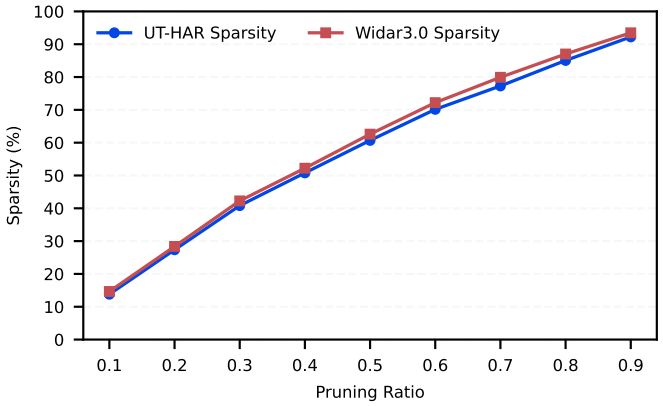


Fig. 6. Sparsity of the model on the UT-HAR and Widar3.0 datasets across varying pruning ratios.

trainable parameter count matches that of QT-7, ensuring a fair comparison under an equal parameter budget. As shown in Table VIII, the hypernetwork achieves an average classification accuracy of 98.60 ± 0.75% on UT-HAR and 63.50 ± 7.39% on Widar3.0, whereas QT-7 attains 99.08 ± 0.24% and 68.76 ± 1.31%, respectively. The accuracy gap widens from 0.48% on UT-HAR to 5.26% on Widar3.0. Moreover, the hypernetwork exhibits much higher run-to-run variance on Widar3.0 (a standard deviation of ±7.39 compared with ±1.31 for QT-7), indicating less stable training. These results indicate that Q-MET's advantage is attributable specifically to the proposed quantum-assisted framework rather than to the broader parameter-generation paradigm.

4) Impact of Pruning Ratios and Warm-Up Epochs on Q-MET: Fig. 6 illustrates the sparsity of the ResNet-18 model on both datasets with different pruning ratios. Both models tend to have distinct and monotonic sparsity increases upon an increase in the pruning ratio. Importantly, the sparsity on both datasets exceeds 70% when the pruning ratio is higher than 0.6, ranging up to 90% at a pruning ratio of 0.9.

Fig. 7 shows the impact of warm-up epochs on Q-MET performance under the selected pruning ratio. The duration of the Warm-Up is varied from 1 to 5 epochs to investigate whether using a longer warm-up period improves performance. This span is in line with common practice, where Warm-Up is generally applied for a few epochs to stabilize the early training process [96], [97]. The results suggest the number of warm-up epochs has little effect on classification accuracy in the two datasets. Accuracy on the UT-HAR dataset is stable and high in all warm-up settings with slight differences around 1%. At a pruning ratio of 0.6, accuracy increases marginally from 98.44% at one warm-up epoch to 98.88% for five epochs, while at an aggressive pruning ratio (0.9), accuracy fluctuates around 96.96% to 98.12% without much indication of increases. For the Widar3.0 dataset, warm-up epochs generate small accuracy differences at moderate pruning ratios and no sustained performance improvement with the time spent warming up. In fact, at a pruning ratio of 0.9, accuracy drops from 66.01% in one warm-up epoch to 60.81% in five, suggesting that extended Warm-Up can be harmful with extreme sparsity. Due to the negligible gains and additional training overhead, a single warm-up epoch gives similar results to those for longer schedules and is thus elected as the optimal choice.

5) Q-MET Performance: Following the previous analysis, we set the Warm-Up epoch to one and compare Q-MET against differing pruning ratios. Fig. 8 displays the classification performance of Q-MET at different pruning ratios on the UT-HAR and Widar3.0 datasets with reference to ResNet-18 (Classical) and QT-based baselines. Overall, for both datasets, Q-MET provides competitive classification accuracy across many sparsity levels.

On the UT-HAR dataset, as shown in Fig. 8(a), Q-MET achieves high accuracy across pruning ratios from 0.1 to 0.8, with average accuracy remaining above 98.4%. In particular, at moderate pruning levels (0.3 to 0.5), Q-MET reaches accuracies of 99.00%, similar to or higher than the QT-7 baseline (99.08%) and distinctly superior to the classical ResNet-18 baseline (98.08%). Q-MET achieves over 98.44% accuracy at a pruning ratio of 0.6 (where more than 70% of parameters are removed), which signifies a small degradation in performance. A severe decline is evident at a pruning ratio of 0.9, where accuracy reaches 97.16%, whilst variance levels rise to 1.15, further suggesting that extreme sparsity starts to limit model capacity. However, Q-MET can still score well against the unpruned ResNet-18 baseline, even under aggressive pruning.

For the Widar3.0 dataset, as shown in Fig. 8(b), Q-MET also demonstrates stable performance at moderate pruning ratios, delivering average accuracies around 70% for pruning ratios up to 0.6. The results suggest that when using a 0.2 pruning ratio, Q-MET attains an accuracy peak of 70.02%, remarkably close to the QT-6 baseline (70.61%) and within a reasonable margin of the dense ResNet-18 baseline (71.29%). Q-MET achieves about 69.5% accuracy at pruning ratios of 0.3, 0.4, and 0.5. With the pruning ratio exceeding 0.6, there is a continuous gradual decrease in accuracy, which is 69.20% at a pruning ratio of 0.6 and 66.01% at 0.9. This trend reflects the higher sensitivity of Widar3.0 to model compression, consistent with the higher complexity of the Widar3.0 dataset. The higher standard deviation at the higher pruning ratio is consistent with reduced stability under extreme sparsity.

Fig. 9 shows the training loss and validation loss convergence curves on the UT-HAR dataset, comparing the baseline ResNet-18 trained with conventional backpropagation and Q-MET with 7 qubits and a pruning ratio of 0.7. Q-MET starts from a higher initial loss than the baseline, but after the early epochs both methods follow a comparable descent trajectory. The baseline converges and early-stops at epoch 83, while Q-MET converges and early-stops at epoch 93, despite using only 5% of the original trainable parameters. A brief loss spike is observed in Q-MET immediately after the pruning point at epoch 1, where more than 70% of the parameters are removed. However, the model recovers quickly within a few epochs and continues to decrease smoothly. Both methods reach similar final training loss values (approximately 0.45) and final validation loss values (approximately 0.46), confirming that Q-MET does not compromise convergence stability.

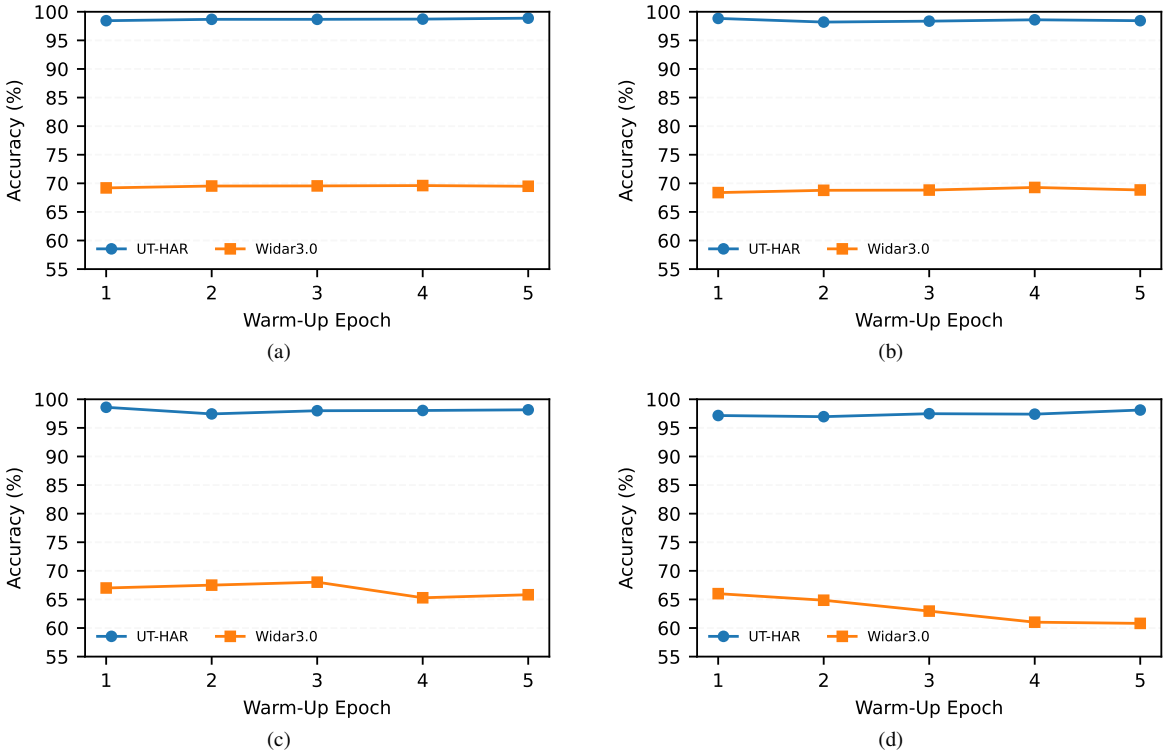


Fig. 7. Performance of Q-MET across warm-up epochs and pruning ratios on UT-HAR and Widar3.0 datasets. (a) Pruning ratio is 0.6. (b) Pruning ratio is 0.7. (c) Pruning ratio is 0.8. (d) Pruning ratio is 0.9.

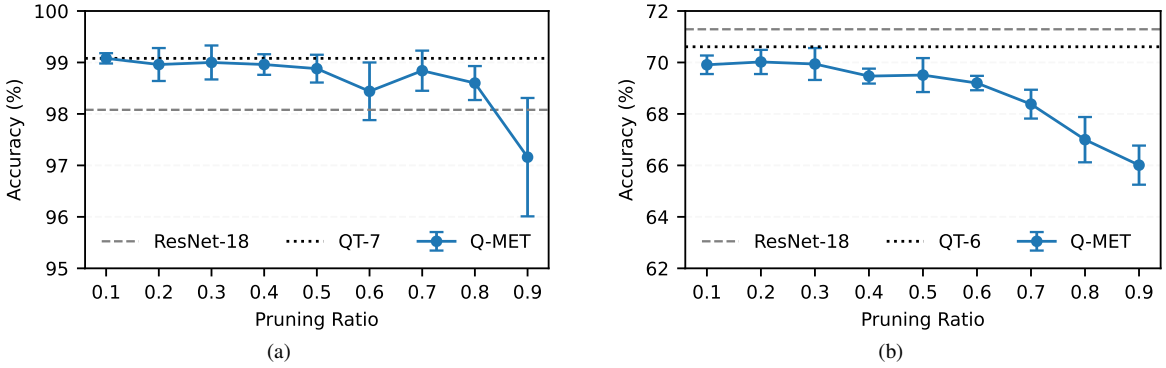
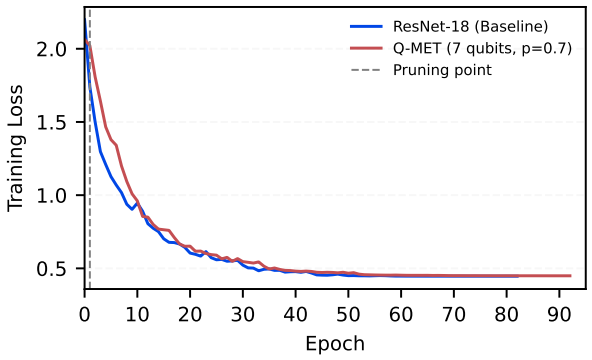


Fig. 8. Performance of Q-MET across varying pruning ratios on UT-HAR and Widar3.0 datasets. (a) Q-MET on UT-HAR dataset. (b) Q-MET on Widar3.0 dataset.

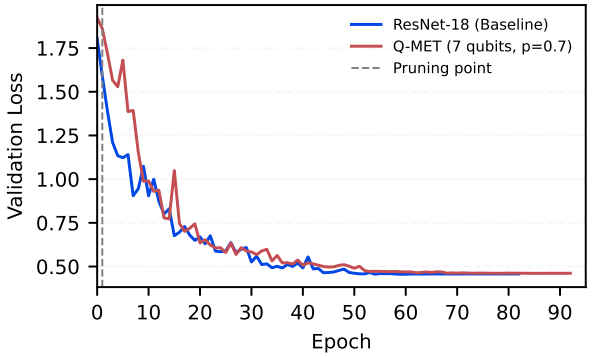
6) Comparison with Train-Then-Prune Pipeline: To justify integrating pruning within the QT training process rather than applying it after training, we compare Q-MET against a train-then-prune (TTP) baseline in which QT-7 is first trained to convergence and then pruned with LAMP using the same criterion and pruning ratios. We evaluate three TTP variants to isolate the effect of fine-tuning after pruning: TTP_0 (no fine-tuning), TTP_{10} (fine-tuning for 10 epochs), and TTP_{50} (fine-tuning for 50 epochs). As shown in Table IX, without any fine-tuning (TTP_0), the network fails completely at moderate-to-high pruning ratios. At a ratio of 0.7, accuracy drops to $9.60 \pm 1.96\%$ on UT-HAR and $3.71 \pm 2.77\%$ on Widar3.0. Adding more fine-tuning epochs recovers much of this loss. At the same ratio, TTP_{50} reaches 98.80% on UT-HAR and 64.74% on Widar3.0, which is close to Q-MET's 98.84% and 66.72%. This recovery, however, requires 50 extra training epochs, while Q-MET needs no additional training after pruning. The benefit of Q-MET becomes clearer when more channels are removed. On UT-HAR at a ratio of 0.9, Q-MET still reaches $97.16 \pm 1.15\%$, while TTP_{50} falls to $92.64 \pm 4.60\%$ and becomes much less stable across runs. On the more difficult Widar3.0 dataset, Q-MET performs best at every pruning ratio, and the gap widens as pruning increases. At a ratio of 0.9, Q-MET reaches 65.73% compared with 62.52% for TTP_{50} and only 6.10% for TTP_0. Q-MET is also consistent, with standard deviations below 1.7 in all settings. These results support the design of Q-MET. By pruning during training, the QPG keeps adapting its parameter generation to the channels that remain active. The model therefore retains capacity that would be permanently lost if pruning were applied after training, and that fine-tuning can recover only in part.

TABLE IX
COMPARISON OF Q-MET AND TTP PIPELINES UNDER VARYING PRUNING RATIOS.

Pruning Ratio	UT-HAR				Widar3.0			
	Q-MET	TTP_0	TTP_{10}	TTP_{50}	Q-MET	TTP_0	TTP_{10}	TTP_{50}
QT-7 (Unpruned)	99.08 ± 0.24				68.76 ± 1.31			
0.1	99.08 ± 0.10	84.32 ± 18.42	98.36 ± 0.59	**99.16 ± 0.15**	**68.43 ± 1.33**	62.08 ± 2.73	67.65 ± 0.89	66.17 ± 1.78
0.2	98.96 ± 0.32	46.88 ± 21.82	97.84 ± 0.86	**99.16 ± 0.23**	**69.00 ± 1.46**	47.27 ± 4.77	67.46 ± 1.10	66.16 ± 1.15
0.3	99.00 ± 0.33	24.88 ± 14.44	98.00 ± 0.55	**99.28 ± 0.10**	**68.96 ± 1.31**	19.19 ± 2.89	66.97 ± 1.22	65.89 ± 1.49
0.4	98.96 ± 0.20	10.84 ± 3.86	97.04 ± 1.22	**99.36 ± 0.23**	**68.46 ± 1.60**	7.46 ± 2.94	66.34 ± 0.92	65.50 ± 1.16
0.5	98.88 ± 0.27	9.72 ± 3.21	94.20 ± 2.86	**99.12 ± 0.20**	**68.71 ± 1.02**	5.14 ± 2.23	65.87 ± 0.51	65.04 ± 1.05
0.6	98.44 ± 0.56	9.60 ± 2.51	91.40 ± 2.75	**99.16 ± 0.23**	**67.15 ± 1.30**	1.47 ± 0.27	65.87 ± 0.91	64.94 ± 0.73
0.7	**98.84 ± 0.39**	9.60 ± 1.96	89.44 ± 1.34	98.80 ± 0.38	**66.72 ± 1.19**	3.71 ± 2.77	64.22 ± 1.09	64.74 ± 0.78
0.8	**98.60 ± 0.33**	9.52 ± 1.98	80.84 ± 2.85	98.56 ± 0.41	**66.70 ± 1.16**	3.59 ± 3.08	60.41 ± 2.81	63.84 ± 1.32
0.9	**97.16 ± 1.15**	9.36 ± 2.22	59.72 ± 5.39	92.64 ± 4.60	**65.73 ± 0.66**	6.10 ± 3.07	52.78 ± 6.24	62.52 ± 1.79



(a)



(b)

Fig. 9. Convergence curves on the UT-HAR dataset, comparing the baseline ResNet-18 and Q-MET (7 qubits, $p = 0.7$). The dashed line marks the pruning point at epoch 1. (a) Training loss. (b) Validation loss.

E. Discussion

The experimental results confirm that the Q-MET approach effectively balances training efficiency, inference performance, and model compactness, making it highly suitable for deployment in resource-constrained scenarios. During training, Q-MET leverages the QT framework to significantly reduce the trainable parameters, saving a large amount of memory while maintaining classification accuracy. In the UT-HAR dataset, Q-MET decreases the size of trainable parameters by about 95%, which is accompanied by an improvement in classification accuracy from 98.08% to 99.08%. On the more challenging Widar3.0 dataset, Q-MET produces a 90% decrease of trainable parameters and a minor decrease in accuracy from 71.29% to 70.61%. This small decrease in performance shows the trade-off between trainable parameters and model capacity on challenging datasets, even though it confirms that Q-MET can substantially reduce the trainable parameter count with minimal impact on recognition accuracy. Although QT framework introduces additional training time overhead compared to conventional training techniques, this is incurred only during offline training and is offset by substantial gains in memory and storage efficiency.

In terms of inference, Q-MET achieves substantial efficiency gains through structured pruning. Q-MET can achieve more than 85% sparsity with a pruning ratio of 0.8 on the UT-HAR dataset and an accuracy of 98.60%, within 1% of the unpruned QT-7 model. Importantly, despite removing more than 85% of the parameters, the Q-MET-trained pruned model still outperforms the unpruned model trained with the classical approach, which achieves an accuracy of 98.08%. On the Widar3.0 dataset, it finds that models trained with the Q-MET framework still work effectively when about 75% of the parameters have been trimmed (pruning ratio of 0.6), achieving results of 69.20% in recognition accuracy, just 1.42% lower than that achieved by an unpruned QT-6. Finally, this warm-up epoch analysis shows that one warm-up epoch is enough to obtain a steady performance under pruning, hence reducing the training costs overall and unnecessary computational overhead.

VII. CONCLUSION

In this study, we proposed Q-MET, a novel memory-efficient end-to-end training framework for HAR. Q-MET leverages a hybrid quantum classical neural network architecture to efficiently train the classical ResNet-18 model. To optimize inference, we integrated LAMP directly into the QT process, effectively eliminating redundant channels and significantly reducing model size for deployment. Experimental results on the UT-HAR and Widar3.0 datasets demonstrate that Q-MET simultaneously enhances both training and inference efficiency. On the UT-HAR dataset, Q-MET enabled training of the ResNet-18 model using only 5% of the original trainable parameters (a 95% reduction) and compressed the final model by approximately 85%. Notably it achieved superior classification accuracy compared to the unpruned ResNet-18. On the more complex Widar3.0 dataset, Q-MET reduced the number of trainable parameters to 10% of the original count and achieved 75% model sparsity, with accuracy degradation limited to within 2%. These findings demonstrate that Q-MET is a robust solution for enabling lightweight inference and

memory-efficient training within resource-constrained HAR structures. Finally, our analysis of warm-up epochs revealed that a single warm-up epoch is sufficient to ensure stable pruning, thereby minimizing the computational overhead of the pre-training phase. Furthermore, since Q-MET is compatible with any compression technique, integrating it with other model-compression methods, such as quantization or hybrid pruning and quantization, is a promising direction for further reducing memory and storage costs. In addition, since HAR systems are often deployed in privacy-sensitive environments, incorporating security and privacy-preserving mechanisms into the Q-MET framework is another important direction. These include privacy-preserving training approaches such as federated learning, which avoid transferring raw CSI data to the cloud, as well as the evaluation and enhancement of the adversarial robustness of Q-MET-trained models against channel-level perturbations.

References

[1] J. Liu, J. Yuan, G. Yu, and J. Han, "Efficient one-shot gesture recognition for WiFi ISAC via aug-meta learning," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 11, pp. 3766–3781, Nov. 2025.

[2] Z. Wang and S. Mao, "AIGC for wireless sensing: Diffusion-empowered human activity sensing," *IEEE Trans. on Cogn. Commun. Netw.*, vol. 11, no. 2, pp. 657–671, Apr. 2025.

[3] Z. Lian, Q. Zeng, Z. Liu, H. Wang, C. Ma, W. Meng, C. Su, and K. Sakurai, "Privacy-enhanced federated WiFi sensing for health monitoring in internet of things," *IEEE Internet Things J.*, vol. 12, no. 3, pp. 2994–3002, Feb. 2025.

[4] V. V. Ratnam, H. Chen, H.-H. Chang, A. Sehgal, and J. Zhang, "Optimal preprocessing of WiFi CSI for sensing applications," *IEEE Trans. Wireless Commun.*, vol. 23, no. 9, pp. 10 820–10 833, Sep. 2024.

[5] J. Liu, T. Mi, X. Shi, Y. Huang, Z. Li, W. Yang, R. Xiong, and R. C. Qiu, "RISAR: Reconfigurable intelligent surfaces-assisted human activity recognition with commercial Wi-Fi devices," *IEEE Internet Things J.*, vol. 11, no. 24, pp. 40 478–40 491, Dec. 2024.

[6] G. Yin, J. Zhang, G. Shen, and Y. Chen, "Fewsense, towards a scalable and cross-domain wi-fi sensing system using few-shot learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 453–468, Jan. 2024.

[7] T. Ropitault, C. R. C. M. da Silva, S. Blandino, A. Sahoo, N. Golmie, K. Yoon, C. Aldana, and C. Hu, "IEEE 802.11bf WLAN sensing procedure: Enabling the widespread adoption of WiFi sensing," *IEEE Commun. Stand. Mag.*, vol. 8, no. 1, pp. 58–64, Mar. 2024.

[8] N. Keshtiarast, P. K. Bishoyi, I. M. Lumbantobing, and M. Petrova, "When sensing meets communication: Coexistence analysis of IEEE 802.11bf and IEEE 802.11ax," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, Jun. 2025, pp. 6486–6491.

[9] R. Du, H. Hua, H. Xie, X. Song, Z. Lyu, M. Hu, Narengerile, Y. Xin, S. McCann, M. Montemurro, T. X. Han, and J. Xu, "An overview on IEEE 802.11bf: WLAN sensing," *IEEE Commun. Surveys Tuts.*, vol. 27, no. 1, pp. 184–217, Feb. 2025.

[10] F. Meneghello, C. Chen, C. Cordeiro, and F. Restuccia, "Toward integrated sensing and communications in IEEE 802.11bf Wi-Fi networks," *IEEE Commun. Mag.*, vol. 61, no. 7, pp. 128–133, Jul. 2023.

[11] J. Liu, H. Liu, Y. Chen, Y. Wang, and C. Wang, "Wireless sensing for human activity: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1629–1645, 3rd Quart. 2020.

[12] C. Wu, B. Wang, O. C. Au, and K. R. Liu, "Wi-Fi can do more: Toward ubiquitous wireless sensing," *IEEE Commun. Stand. Mag.*, vol. 6, no. 2, pp. 42–49, Jun. 2022.

[13] S. Tan, Y. Ren, J. Yang, and Y. Chen, "Commodity WiFi sensing in ten years: Status, challenges, and opportunities," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 17 832–17 843, Sep. 2022.

[14] Z. He, M. Bouazizi, G. Gui, and T. Ohtsuki, "A cross-subject transfer learning method for CSI-based wireless sensing," *IEEE Internet Things J.*, vol. 12, no. 13, pp. 23 946–23 960, Jul. 2025.

[15] F. Wang, W. Gong, and J. Liu, "On spatial diversity in WiFi-based human activity recognition: A deep learning-based approach," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2035–2047, Apr. 2019.

[16] J. Zhang, Y. Li, H. Xiong, D. Dou, C. Miao, and D. Zhang, "HandGest: Hierarchical sensing for robust-in-the-air handwriting recognition with commodity WiFi devices," *IEEE Internet Things J.*, vol. 9, no. 19, pp. 19 529–19 544, Oct. 2022.

[17] S. M. Hernandez and E. Bulut, "WiFi sensing on the edge: Signal processing techniques and challenges for real-world systems," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 46–76, 1st Quart. 2023.

[18] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[19] Q. Feng, C. Duan, J. Xue, C. Li, F. Huang, X. Zhang, J. Weng, and P. S. Yu, "Imbalanced semi-supervised learning for WiFi gesture recognition via dynamic threshold-based spatio-temporal attention networks," *IEEE Trans. Mobile Comput.*, vol. 25, no. 1, pp. 483–499, Jan. 2026.

[20] X. Wu, Z. Chu, P. Yang, C. Xiang, X. Zheng, and W. Huang, "TW-See: Human activity recognition through the wall with commodity Wi-Fi devices," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 306–319, Jan. 2019.

[21] J. Yang, X. Chen, H. Zou, C. X. Lu, D. Wang, S. Sun, and L. Xie, "SenseFi: A library and benchmark on deep-learning-empowered WiFi human sensing," *Patterns*, vol. 4, no. 3, p. 100703, Mar. 2023.

[22] W. Jiao, C. Zhang, W. Du, and S. Ma, "WiSDA: Subdomain adaptation human activity recognition method using Wi-Fi signals," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 2876–2888, Apr. 2025.

[23] X. Chen, C. Li, C. Jiang, W. Meng, and W. Xiao, "WiPhase: A human activity recognition approach by fusing of reconstructed WiFi CSI phase features," *IEEE Trans. Mobile Comput.*, vol. 24, no. 1, pp. 394–406, Jan. 2025.

[24] C. Liu, Y. Hao, Y. Liu, X. Zhang, X. Wang, and Y. Liu, "LWiHS: A lightweight Wi-Fi-enabled human sensing using feature fusion strategy," *IEEE Trans. Instrum. Meas.*, vol. 74, pp. 1–15, May 2025.

[25] X. Hoang Nguyen, V.-D. Nguyen, Q.-T. Luu, T. Dinh Gian, and O.-S. Shin, "Robust WiFi sensing-based human pose estimation using denoising autoencoder and CNN with dynamic subcarrier attention," *IEEE Internet Things J.*, vol. 12, no. 11, pp. 17 066–17 079, Jun. 2025.

[26] F. Shi, W. Li, C. Tang, Y. Fang, P. V. Brennan, and K. Chetty, "ML-Track: Passive human tracking using WiFi multi-link round-trip CSI and particle filter," *IEEE Trans. Mobile Comput.*, vol. 24, no. 6, pp. 5155–5172, Jun. 2025.

[27] T. D. Gian, D. T. Tran, Q.-V. Pham, L.-N. Tran, and V.-D. Nguyen, "Multi-modal human pose estimation: A Wi-Fi-driven approach with adaptive kernel selection," *IEEE Trans. Artif. Intell.*, pp. 1–14, May 2026.

[28] Y. Zhou, J. Yang, H. Huang, and L. Xie, "AdaPose: Toward cross-site device-free human pose estimation with commodity WiFi," *IEEE Internet Things J.*, vol. 11, no. 24, pp. 40 255–40 267, Dec. 2024.

[29] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dublin, Ireland, Jun. 2020, pp. 1–6.

[30] X. Cao, T. Başar, S. Diggavi, Y. C. Eldar, K. B. Letaief, H. V. Poor, and J. Zhang, "Communication-efficient distributed learning: An overview," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 851–873, Apr. 2023.

[31] Á. L. García, J. M. De Lucas, M. Antonacci, W. Zu Castell, M. David, M. Hardt, L. L. Iglesias, G. Moltó, M. Plociennik, V. Tran *et al.*, "A cloud-based framework for machine learning workloads and applications," *IEEE Access*, vol. 8, pp. 18 681–18 692, Jan. 2020.

[32] M. K. Lenka and A. Chakraborty, "WISDOM: A framework for scaling on-device Wi-Fi sensing solutions," *Ad Hoc Netw.*, vol. 178, p. 103915, Jun. 2025.

[33] M. De Sanctis, "The paradigm of massive wireless human sensing: Concept, architecture and challenges," *IEEE Internet Things Mag.*, vol. 8, no. 3, pp. 134–141, May 2025.

[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, Georgia, Nov. 2020.

[35] H. Cai, C. Gan, L. Z. Massachusetts Institute of Technology, and S. H. Massachusetts Institute of Technology, "TinyTL: reduce memory, not parameters for efficient on-device learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2020.

[36] T. T. An, A. Argyriou, A. A. Puspitasari, S. L. Cotton, and B. M. Lee, "Efficient automatic modulation classification for next-generation wireless networks," *IEEE Trans. Green Commun. Netw.*, vol. 10, pp. 249–259, May 2025.

[37] R. Raina, N. Simmons, D. E. Simmons, M. D. Yacoub, and T. Q. Duong, "To trust or not to trust: On calibration in ML-based resource allocation for wireless networks," *IEEE Trans. Netw. Sci. Eng.*, pp. 1–18, Jul. 2025.

[38] Z. Li, J. Zhang, C. R. C. M. Da Silva, O. Yurduseven, T. Q. Duong, C. Fischione, and S. L. Cotton, "Efficient 6-GHz Wi-Fi-Based occupancy detection: Channel model-informed feature engineering and random forest optimization," *IEEE Internet Things J.*, vol. 12, no. 24, pp. 53 800–53 813, Dec. 2025.

[39] T. T. An and B. M. Lee, "Robust automatic modulation classification in low signal to noise ratio," *IEEE Access*, vol. 11, pp. 7860–7872, Jan. 2023.

[40] H.-T. P. Dang, H.-T. Nguyen, Q.-V. Pham, V.-C. Phan, and T. Huynh-The, "Compact spectrum sensing with RPNet+CLRu: Reducing model size for 5G-LTE signal identification," *IEEE Trans. Veh. Technol.*, vol. 75, no. 4, pp. 5819–5830, Apr. 2026.

[41] I. Ahmad, A. Ullah, and W. Choi, "WiFi-based human sensing with deep learning: Recent advances, challenges, and opportunities," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 3595–3623, Jun. 2024.

[42] J. Yang, X. Chen, H. Zou, D. Wang, Q. Xu, and L. Xie, "EfficientFi: Toward large-scale lightweight WiFi sensing via CSI compression," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13 086–13 095, Aug. 2022.

[43] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Salt Lake City, Utah, Jun. 2018, pp. 2704–2713.

[44] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 2900–2919, May 2024.

[45] 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.

[46] 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.

[47] 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, pp. 1–6.

[48] Y. Zhang, Y. Zheng, K. Qian, G. Zhang, Y. Liu, C. Wu, and Z. Yang, "Widar3.0: Zero-effort cross-domain gesture recognition with Wi-Fi," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 8671–8688, Nov. 2022.

[49] S. Yousefi, H. Narui, S. Dayal, S. Ermon, and S. Valaee, "A survey on behavior recognition using WiFi channel state information," *IEEE Commun. Mag.*, vol. 55, no. 10, pp. 98–104, Oct. 2017.

[50] Y. Sun, F. Li, G. Li, X. Ma, Q. Gao, M. Pan, and J. Wang, "Enabling lightweight device-free wireless sensing with network pruning and quantization," *IEEE Sensors J.*, vol. 22, no. 1, pp. 969–979, Jan. 2022.

[51] T. T. An, G. Yin, J. Zhang, Y. Ding, T. Q. Duong, and S. L. Cotton, "A quantum-optimized training framework for radio frequency fingerprint identification," *IEEE J. Sel. Areas Commun.*, vol. 44, pp. 5327–5340, Jun. 2026.

[52] 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.

[53] 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.

[54] F. S. Abuhoureyah and Y. C. Wong, "Location independent human activity recognition using self-training CSI-based techniques for wireless sensor networks," *IEEE Internet Things J.*, vol. 12, no. 14, pp. 27 419–27 434, Jul. 2025.

[55] J. Zhang, Y. Li, and W. Xiao, "Integrated multiple kernel learning for device-free localization in cluttered environments using spatiotemporal information," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4749–4761, Mar. 2021.

[56] C. Li, M. Liu, and Z. Cao, "WiHF: Gesture and user recognition with WiFi," *IEEE Trans. Mobile Comput.*, vol. 21, no. 3, pp. 757–768, Mar. 2022.

[57] Y. Gu, X. Zhang, Y. Wang, M. Wang, H. Yan, Y. Ji, Z. Liu, J. Li, and M. Dong, "WiGRUNT: WiFi-enabled gesture recognition using dual-attention network," *IEEE Trans. Hum.-Mach. Syst.*, vol. 52, no. 4, pp. 736–746, Aug. 2022.

[58] Y. Zhang, Y. Chen, Y. Wang, Q. Liu, and A. Cheng, "CSI-based human activity recognition with graph few-shot learning," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4139–4151, Mar. 2022.

[59] J. Zhang, J. Xue, Y. Li, and S. L. Cotton, "Leveraging online learning for domain-adaptation in Wi-Fi-based device-free localization," *IEEE Trans. Mobile Comput.*, vol. 24, no. 8, pp. 7773–7787, Aug. 2025.

[60] J. Zhang, Y. Li, W. Xiao, and Z. Zhang, "Online spatiotemporal modeling for robust and lightweight device-free localization in nonstationary environments," *IEEE Trans. Ind. Informat.*, vol. 19, no. 7, pp. 8528–8538, Jul. 2023.

[61] 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.

[62] V. Dunjko, J. M. Taylor, and H. J. Briegel, "Quantum-enhanced machine learning," *Phys. Rev. Lett.*, vol. 117, no. 13, p. 130501, Sep. 2016.

[63] B. Narottama, B. Canberk, S. L. Cotton, H. Shin, G. K. Karagiannidis, and T. Q. Duong, "Non-centralized quantum neural networks for cell-free MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 25, pp. 9971–9985, Dec. 2025.

[64] H.-K. Lau, R. Pooser, G. Siopsis, and C. Weedbrook, "Quantum machine learning over infinite dimensions," *Phys. Rev. Lett.*, vol. 118, no. 8, p. 080501, Feb. 2017.

[65] 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.

[66] J. Tian, X. Sun, Y. Du, S. Zhao, Q. Liu, K. Zhang, W. Yi, W. Huang, C. Wang, X. Wu, M.-H. Hsieh, T. Liu, W. Yang, and D. Tao, "Recent advances for quantum neural networks in generative learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 10, pp. 12 321–12 340, Oct. 2023.

[67] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nat. Commun.*, vol. 12, no. 1, p. 2631, May 2021.

[68] N. Q. T. Thoong, A. A. Cheema, B. Canberk, D. T. Tran, O. A. Dobre, and T. Q. Duong, "Channel estimation for reconfigurable intelligent surface-aided 6G NOMA systems: A quantum machine learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 13, pp. 2197–2218, Sep. 2025.

[69] 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.

[70] D. Peral-García, J. Cruz-Benito, and F. J. García-Peñalvo, "Comparing natural language processing and quantum natural processing approaches in text classification tasks," *Expert Syst. Appl.*, vol. 254, p. 124427, Nov. 2024.

[71] M. O. Butt, N. Waheed, T. Q. Duong, and W. Ejaz, "Quantum-inspired resource optimization for 6G networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 27, no. 5, pp. 2973–3019, Oct. 2025.

[72] C.-Y. Liu, C.-H. A. Lin, and K.-C. Chen, "Quantum-train with tensor network mapping model and distributed circuit ansatz," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Hyderabad, India, Apr. 2025, pp. 1–4.

[73] C.-Y. Liu, C.-H. A. Lin, W.-J. Huang, and M.-H. Hsieh, "Introduction to quantum-train toolkit," in *Proc. IEEE Int. Conf. Quantum Comput. Eng. (QCE)*, vol. 02, Montreal, QC, Canada, Sep. 2024, pp. 456–457.

[74] C.-H. A. Lin, C.-Y. Liu, S. Y.-C. Chen, and K.-C. Chen, "Quantum-trained convolutional neural network for deepfake audio detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. Workshops (ICASSPW)*, Hyderabad, India, Apr. 2025, pp. 1–5.

[75] C.-Y. Liu and S. Y.-C. Chen, "Federated quantum-train with batched parameter generation," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Jeju Island, South Korea, Oct. 2024, pp. 1133–1138.

[76] W. O'Quinn and S. Mao, "Quantum machine learning: Recent advances and outlook," *IEEE Wireless Commun.*, vol. 27, no. 3, pp. 126–131, Jun. 2020.

[77] 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.

[78] S. C. Prabhashana, D. V. Huynh, H. Jung, B. Canberk, S. L. Cotton, and T. Q. Duong, "Quantum deep reinforcement learning for URLLC satellite-air-ground integrated networks with digital twin applications," *IEEE Internet Things J.*, vol. 13, no. 3, pp. 4230–4246, Feb. 2026.

[79] A. A. Puspitasari and B. M. Lee, "Quantum reinforcement learning for uav-enhanced next-generation wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 28, pp. 3089–3124, Jan. 2026.

[80] M. Cerezo, A. Arrasmith, R. Babbush, S. Benjamin, S. Endo, K. Fujii, J. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. Coles, "Variational

quantum algorithms," *Nat. Rev. Phys.*, vol. 3, no. 9, pp. 625–644, Sep. 2021.

[81] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, Jun. 2017.

[83] H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 10 558–10 578, Dec. 2024.

[84] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Munich, Germany, Sep. 2018, pp. 184–199.

[85] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Montreal, QC, Canada, Dec. 2015, pp. 1135–1143.

[86] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[87] Y.-J. Zheng, S.-B. Chen, C. H. Q. Ding, and B. Luo, "Model compression based on differentiable network channel pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 12, pp. 10 203–10 212, Dec. 2023.

[88] W. Zhao, L. Zou, Z. Wang, X. Yao, and B. Yu, "HAPE: Hardware-aware LLM pruning for efficient on-device inference optimization," *ACM Trans. Design Autom. Electron. Syst.*, vol. 30, no. 4, pp. 1–18, Jun. 2025.

[89] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 1398–1406.

[90] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 5068–5076.

[91] J. Lee, S. Park, S. Mo, S. Ahn, and J. Shin, "Layer-adaptive sparsity for the magnitude-based pruning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Virtual Event, Austria, May 2021.

[92] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Virtual Conference, Apr. 2022.

[93] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," *J. Mach. Learn. Res.*, vol. 22, no. 241, pp. 1–124, 2021.

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

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

[96] M. Shen, P. Molchanov, H. Yin, and J. M. Alvarez, "When to prune? A policy towards early structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 12 247–12 256.

[97] A. Peste, E. Iofinova, A. Vladu, and D. Alistarh, "AC/DC: alternating compressed/decompressed training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Dec. 2021, pp. 8557–8570.