Published: 30 March 2026
Online AM: 27 February 2026
Accepted: 18 February 2026
Revised: 27 January 2026
Received: 18 October 2025

Citation in BibTeX format

RESEARCH-ARTICLE

WECAR: An End-Edge Collaborative Inference and Training Framework for WiFi-Based Continuous Human Activity Recognition

RONG LI, Soochow University, Suzhou, Jiangsu, China

TAO DENG, Soochow University, Suzhou, Jiangsu, China

SIWEI FENG, Soochow University, Suzhou, Jiangsu, China

HE HUANG, Soochow University, Suzhou, Jiangsu, China

JUNCHENG JIA, Soochow University, Suzhou, Jiangsu, China

DI YUAN, Uppsala University, Uppsala, Uppsala, Sweden

View all

Open Access Support provided by:

Uppsala University

Soochow University

SUNY New Paltz

WECAR: An End-Edge Collaborative Inference and Training Framework for WiFi-Based Continuous Human Activity Recognition

RONG LI, Soochow University School of Computer Science and Technology, Suzhou, China
TAO DENG, Soochow University School of Computer Science and Technology, Suzhou, China
SIWEI FENG, School of Computer Science and Technology, Soochow University, Suzhou, China
HE HUANG, Soochow University School of Computer Science and Technology, Suzhou, China
JUNCHENG JIA, School of of Computer Science and Technology, Soochow University, Suzhou, China
DI YUAN, Uppsala University Department of Information Technology, Uppsala, Sweden
KEQIN LI, Department of Computer Science, State University of New York, New Paltz, United States

WiFi-based human activity recognition (HAR) holds significant promise for ubiquitous sensing in smart environments. A critical challenge is enabling systems to dynamically adapt to evolving scenarios, learn new activities without catastrophically forgetting prior knowledge, and meet edge devices' computational constraints. Current approaches struggle to reconcile these due to high historical data storage demands and inefficient parameter utilization. We propose WECAR, an end-edge collaborative inference and training framework for WiFi-based continuous HAR. In this framework, edge devices handle model training, lightweight optimization, and updates, while end devices perform efficient inference. WECAR introduces two key innovations, i.e., dynamic continual learning with parameter efficiency and hierarchical distillation for end deployment. For the former, we propose a transformer-based architecture enhanced by task-specific dynamic model expansion and stability-aware selective retraining. For the latter, we propose a dual-phase distillation mechanism that includes multi-head self-attention relation distillation and prefix relation distillation. We implement WECAR based on heterogeneous hardware using *Jetson Nano* as edge devices and the *ESP32* as end devices, respectively. Our experiments across three public WiFi datasets reveal that WECAR not only outperforms several state-of-the-art methods in performance and parameter efficiency, but also achieves a substantial reduction in the model's parameter count post-optimization without sacrificing accuracy.

This article was partially presented at AAAI Conference on Artiffcial Intelligence (AAAI) 2025 [38].
The work of Rong Li and Tao Deng was supported by Natural Science Foundation of Jiangsu Province, China under Grant BK20230477. The work of Siwei Feng was supported by NSFC under Grant 62106167. The work of He Huang was supported by NSFC under Grant 62332013. The work of Juncheng Jia was supported by Collaborative Innovation Center of Novel Software Technology and Industrialization, Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Suzhou Frontier Science and Technology Program under Grant SYG202310. The work of Di Yuan was supported by the Swedish Research Council with grant number 2022-04123 and the EU MSCA project IPOSEE with grant number 101086219.
Authors' Contact Information: Rong Li, School of Computer Science and Technology, Soochow University, Suzhou, China; email: 20235227092@stu.suda.edu.cn; Tao Deng (corresponding author), School of Computer Science and Technology, Soochow University, Suzhou, China; email: dengtao@suda.edu.cn; Siwei Feng, School of Computer Science and Technology, Soochow University, Suzhou, China; email: swfeng@suda.edu.cn; He Huang, School of Computer Science and Technology, Soochow University, Suzhou, China; email: huangh@suda.edu.cn; Juncheng Jia, School of Computer Science and Technology, Soochow University, Suzhou, China; email: jiajuncheng@suda.edu.cn; Di Yuan, Department of Information Technology, Uppsala University, Uppsala, Sweden; email: di.yuan@it.uu.se; Keqin Li, Department of Computer Science, State University of New York, New Paltz, New York, USA; email: lik@newpaltz.edu.

ACM 1550-4859/2026/03-ART21
https://doi.org/10.1145/3799718

CCS Concepts: • **Computing methodologies → Machine learning;**

Additional Key Words and Phrases: Continual learning, end-edge collaboration, knowledge distillation, human activity recognition, WiFi

ACM Reference Format:
Rong Li, Tao Deng, Siwei Feng, He Huang, Juncheng Jia, Di Yuan, and Keqin Li. 2026. WECAR: An End-Edge Collaborative Inference and Training Framework for WiFi-Based Continuous Human Activity Recognition. *ACM Trans. Sensor Netw.* 22, 3, Article 21 (March 2026), 24 pages. https://doi.org/10.1145/3799718

1 Introduction

Human activity recognition (**HAR**) technologies are widely used in medical monitoring [1], smart homes [2], and security [3]. Traditional video surveillance systems for HAR face challenges like privacy issues, limited field of view, and lighting sensitivity. In contrast, wireless signal sensing, particularly WiFi-based HAR, offers a non-intrusive and privacy-respecting solution. WiFi is ideal for HAR due to its ubiquity and low hardware requirements [4, 5]. At present, WiFi-based HAR mainly relies on **channel state information** (**CSI**) that describes how the signal propagates from the WiFi transmitter to the receiver, including attenuation, scattering, fading, and power changes with distance on each WiFi transmission path. For WiFi-based HAR, traditional **deep learning** (**DL**) models [6, 7] face difficulties in recognizing new activities, as they rely on static models that cannot adapt to new human activities. Simple fine-tuning of DL models with new data without access to the original training data may result in catastrophic forgetting, where previously learned knowledge is overwritten. As shown in Figure 1, the model retrained on new activities (swim and jump) fails to recognize prior activities (sit and stand). Therefore, there is an urgent need to develop DL models that can effectively adapt to dynamically changing environments. To address the aforementioned drawbacks of traditional technologies, WiFi-based continual HAR technology has emerged. Its core strengths lie in contactless and wearable-free operation as well as strong environmental adaptability. By integrating continual learning methods, this technology effectively solves the problem of catastrophic forgetting in incremental tasks.

In practical WiFi deployments, WiFi receivers serve as the primary data acquisition units, capturing detailed CSI from ambient signals. In this article, we use the terms WiFi receiver and end device interchangeably. Due to cost considerations and practical constraints, these devices are typically resource-constrained (e.g., the ESP32-C3, which can be used directly as a WiFi receiver, is equipped with a RISC-V 32-bit single-core processor, 400 KB SRAM and 4 MB flash memory [8]), limiting their capabilities to model inference while rendering model training infeasible. This constraint inherently establishes a dual-layer architecture that decouples training from inference [9]. In addition, these constraints necessitate WiFi-based HAR systems to be capable of continuous sensing without storing user data, which is a critical requirement for preserving privacy and operational efficiency. In order to address privacy and storage issues, **exemplar-free class-incremental learning** (**EFCIL**) [10–12] aims to recognize both old and new activity classes without retaining previous exemplars. However, while existing EFCIL methods have demonstrated their efficacy in the area of **computer vision** (**CV**), their direct adaptation to WiFi-based HAR introduces unique challenges. First, unlike static images, WiFi signals experience subtle, time-sensitive fluctuations due to human activities, which complicates the extraction of stable features. The continuous and rapidly changing nature of WiFi data, coupled with the lack of clear spatial references, further exacerbates this issue. Additionally, conventional EFCIL approaches typically demand significant computational resources and extended training durations, rendering them impractical for deployment on resource-constrained edge devices like WiFi receivers. Consequently, there exists an

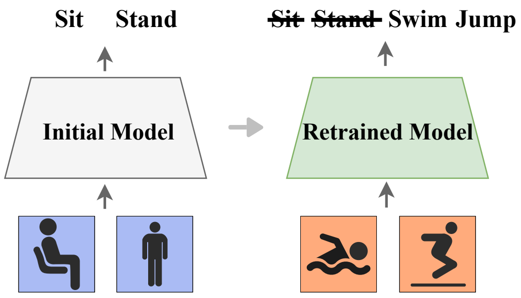


Fig. 1. The initial model was trained to recognize two activities: sitting and standing. However, after retraining the model with data from two new activities swimming and jumping, it loses the ability to recognize sitting and standing.

urgent need for a lightweight framework that simultaneously meet three critical requirements: (1) effective spatiotemporal feature extraction from sequential WiFi data, (2) mitigation of catastrophic forgetting in incremental learning scenarios, and (3) substantial reduction of computational and storage overhead for edge compatibility.

In this work, we propose an end-edge collaborative inference and training EFCIL framework for WiFi-Based HAR. We name the framework WECAR, as it involves WiFi, end-edge devices, continual learning, and activity recognition. In WECAR, model training, lightweighting, and updates occur on edge devices, while inference is delegated to end devices. The main contributions of this article are as follows.

(1) To capture spatiotemporal dependencies in sequential WiFi data, we leverage a transformer-based architecture augmented with two synergistic mechanisms, i.e., dynamic model expansion and selective retraining. In the dynamic model expansion mechanism, we introduce parameter-efficient task-specific prefixes—small-scale trainable parameters embedded within **multi-head self-attention** (**MHSA**) layers. These learnable prefixes are optimized per incremental task to isolate and preserve critical task-specific features while preventing interference with previously acquired knowledge. In the selective retraining mechanism, a neuron stability-aware weighting mechanism dynamically adjusts **multilayer perceptron** (**MLP**) layer contributions based on cross-task performance metrics. By prioritizing stable neurons for retention and adaptive ones for recalibration, this approach balances plasticity for new tasks with stability for past knowledge. This dual strategy enables rapid training through partial parameter updates while maintaining robust continual learning performance.
(2) To enable efficient edge-to-end device deployment, we develop a two-tier distillation framework, i.e., MHSA relation distillation and prefix relation distillation. The MHSA relation distillation aligns the student model's attention patterns with the teacher model by minimizing discrepancies in attention matrices across all heads, preserving inter-feature relationships critical for temporal dynamics. The prefix relation distillation transfers task-specific prefix embeddings from the teacher to student models through contrastive alignment in the MHSA latent space, ensuring retention of historical task knowledge during model compression. This paradigm reduces model parameters without significant accuracy degradation, producing deployable models that retain both generalization capacity and incremental learning readiness.
(3) We implement WECAR using *Jetson Nano* and *ESP32* as edge and end devices respectively. We validate WECAR on three publicly available WiFi datasets. The results show that it outperforms state-of-the-art EFCIL baselines in average incremental accuracy while requiring

fewer parameters. In addition, it maintains baseline accuracy despite parameter reduction, validating its practical viability for resource-constrained environments.

2 Related Work

The field of WiFi-based HAR has gained extensive attention. For instance, the works in Refs. [13] and [14] use the **received signal strength indication** (**RSSI**) of WiFi for gesture and activity recognition. However, RSSI's effectiveness is constrained by multipath and fading effects. In contrast, WiFi's CSI can provide more reliable information, as highlighted in Ref. [15]. This has led to its adoption in works such as [16], which employs the Fresnel zone model to enhance recognition accuracy. To bypass the limitations of manual feature extraction, DL based methods have been increasingly utilized. For example, [17] applies 1D **Convolutional Neural Networks** (**CNNs**) for indoor positioning, while [18] uses CNNs to classify activities from CSI. Despite these advancements, traditional DL models are static models that cannot adapt to new human actions.

Class-incremental learning (**CIL**) emerges as a promising solution to address the dynamic adaptation challenges in HAR. CIL can be broadly categorized into three types: rehearsal-based methods [19, 20], regularization-based methods [21, 22], and dynamic architecture-based methods [23, 24]. These approaches primarily rely on storing exemplars from past tasks to mitigate catastrophic forgetting. However, in real-world WiFi systems, receivers are often resource-constrained and face privacy concerns in user data retention. This makes data storage impractical and calls for WiFi-based HAR systems that can operate continuously without storing user data, ensuring both privacy and efficiency. To tackle this issue, several EFCIL methods have been proposed. For example, the work in Ref. [10] integrates knowledge distillation into CIL, although its efficacy is limited when relying solely on new data. The work in Ref. [25] proposes a framework that separates representation and classifier learning, thereby improving data synthesis for previous tasks. The work in Ref. [26] introduces a prototype-sample relation distillation by combining supervised contrastive loss [27], self-supervised learning [28], and class prototype evolution techniques [29]. These methods mainly employ ResNet and other CNN-based models. However, CSI data differ from image data in spatial-temporal properties, making 2D convolution kernels inefficient for capturing their non-local spatial correlations and long-range temporal dynamics. In contrast, Transformer's MHSA directly models global CSI dependencies, which is critical for HAR. Moreover, its architecture natively supports prefix-based parameter-efficient adaptation, a key enabler for our continual learning task that is not easily replicated by complex convolution models. While numerous CNN and ResNet-based approaches have been developed for EFCIL, transformer-based EFCIL remains a relatively unexplored area. DyTox [24] introduces task tokens interacting with the network at the output stage, while ConTraCon [30] modulates attention by transforming MHSA weights via convolution and gating. In contrast, WECAR injects task-specific key/value prefixes within the MHSA module and keeps the original MHSA projections frozen. This design encodes task knowledge as explicit prefix embeddings instead of modifying backbone weights. Additionally WECAR integrates stability-aware selective retraining in MLP layers and dual-phase distillation to compress the expanded teacher into a lightweight student for deployment.

A few recent works investigate WiFi-based HAR with incremental learning [31–33]. The work in Ref. [31] employs exemplars and distillation loss for knowledge retention, though its reliance on stored exemplars raises privacy concerns and memory constraints. The work in Ref. [32] introduces an enhanced CNN with attention and dual-loss functions. However, [32] is limited to processing one category at a time, restricting its applicability. The work in Ref. [33] proposes a solution tailored for wireless sensing services, suitable for customized incremental services, but it also requires storing exemplars. The aforementioned research has primarily focused on single-person WiFi sensing. Several studies have explored multi-person WiFi sensing [34–37]. The work in Ref. [34] transfers vision foundation model knowledge to the RF modality for cross-modal multi-person HAR. The work in Ref. [35] exploits near-field channel dominance for physically separable multi-user sensing. The work in Ref. [36] emulates ultra-wideband sensing using fast channel hopping over a wide spectrum and reconstructing signals from sparse and irregular samples. The work in Ref. [37] leverages commodity MU-MIMO beam patterns to jointly support sensing and communication.

We previously proposed a ConSense model for WiFi-based HAR [38]. Compared with Ref. [38], the contributions of this article are as follows. First, while [38] focuses on a single-device continual learning, this article introduces a novel end-edge collaborative paradigm that decouples training/lightweighting (edge) from inference (end devices). This architectural shift enables privacy preservation, reduces on-device computational burden, and supports scalable deployment; such advanced have not been addressed in prior work. In addition, to enable efficient edge-to-end device deployment, this article develops a two-tier distillation framework. This framework addresses a critical gap in Ref. [38] by enabling task-aware model compression and zero exemplar leakage during edge-to-end device deployment, thus ensuring strict privacy compliance. Finally, this article realizes an end-to-end system. Unlike the simulation-based validation in [38], this article demonstrates real-world practicality through implementation on heterogeneous hardware (Jetson Nano edge and ESP32 end devices).

3 Background and Problem Statement

3.1 WiFi Channel State Information

The principle of WiFi-based HAR stems from the dynamic interplay between propagating radio waves and human motion. When transmitted signals interact with environmental perturbations and obstacles caused by human activities, alterations in propagation paths occur through reflection, diffraction, and scattering phenomena. These path deviations induce time-variant multipath interference patterns, encoding motion-induced environmental signatures into measurable signal distortions. By analyzing these time-variant signal distortions, we can derive precise environmental sensing.

CSI quantifies fine-grained signal propagation characteristics in multipath WiFi channels, capturing physical-layer distortions such as frequency-selective fading, delay spread, and Doppler shifts. Leveraging the inherent subcarrier granularity of **orthogonal frequency division multiplexing (OFDM)**, CSI enables multidimensional feature extraction by decomposing channel responses across orthogonal subcarriers. This process is formally modeled as

$$H(f_k) = |H(f_k)|e^{j\angle H(f_k)}, \tag{1}$$

where $H(f_k)$ denotes the CSI of the subcarrier with central frequency of f_k and $\angle H$ denotes its phase. The value of each CSI is represented in complex form as $a+bi$, with its amplitude expressed as $\sqrt{a^2+b^2}$. The CSI is organized into a matrix $D \in \mathbb{R}^{n\times d}$. Here, n represents the temporal dimension, defined as $n = fs$, where f denotes the frequency of data packet collection, and s represents the sampling time for an action. The channel dimension d is defined as $d = ew$, where e refers to the number of transmitting and receiving antennas, and w represents the number of subcarriers per antenna pair.

3.2 Problem Statement

In practical WiFi-based sensing systems, the WiFi receiver serves as the principal data acquisition node, capturing discrete activity classes set (denoted by $\mathcal{C}_t$, where t represents the t-th data collection) at each sampling interval. At each data collection session t, the receiver captures a new activity class set $\mathcal{C}_t$. The incremental classes at session t are exactly the classes contained in $\mathcal{C}_t$. Accordingly, the incremental training stage updates the model using only the newly collected data from $\mathcal{C}_t$.

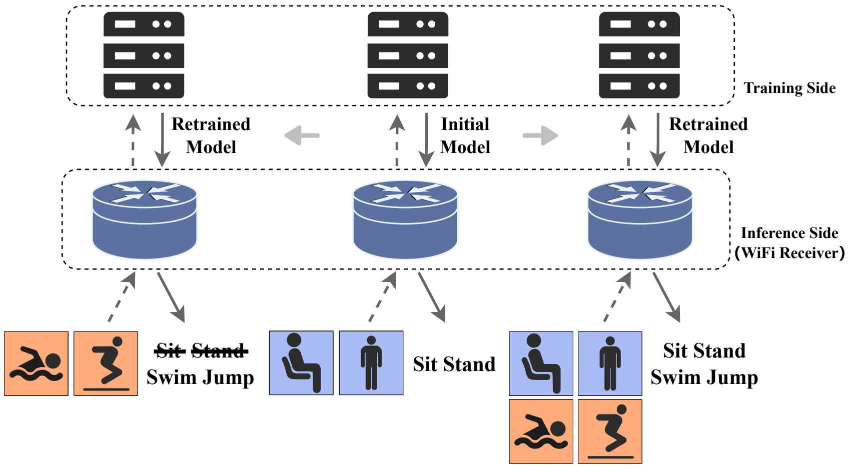


Fig. 2. The middle layer includes WiFi receivers, which are responsible for data collection and inference. The upper layer represents the server, which is in charge of model training. After the receiver collects the CSI data, it transmits it to the server for model training. Once the training is complete, the trained model is sent back to the receiver for inference.

In our experiments, $\mathcal{C}_t$ is instantiated by the standard class incremental splits of each dataset, with fixed number of classes per task. For example, as shown in the central panel of Figure 2, the initial data collection phase detects two distinct activities (sitting and standing). While end devices are constrained by limited storage and computational capacity for local model training, cloud-centric approaches, though theoretically capable of addressing computational bottlenecks, introduce prohibitive privacy vulnerabilities and bandwidth overhead due to frequent model updates and bulk data transmission. To mitigate these challenges, we delegate the training process to a co-located edge module. This architecture ensures privacy preservation through localized data processing and reduces bandwidth demands via LWM. The optimized models are then deployed on end devices for real-time inference. Thus, from the perspectives of real-time performance, overhead, and privacy protection, it is preferable to decouple inference and training. In real-world deployment scenarios, a WiFi signal receiver collects CSI data on new activity classes and then transmits the newly gathered task data to a WiFi signal transmitter. A round of incremental training is subsequently executed on this transmitter, and the LWM generated upon training completion is pushed to the receiver to performance inference. Regarding the task triggering mechanism, we adopt a data-driven scheme. Specifically, a new task is triggered when edge devices accumulate a sufficient volume of valid samples associated with new activities.

Traditional DL models face inherent limitations in WiFi-based HAR scenarios, particularly in recognizing emerging activities. Static DL models lack adaptability to novel actions, and naive fine-tuning with new data (absent original training data) triggers catastrophic forgetting–a phenomenon where parameter overwriting erases prior knowledge. For example, when the WiFi receiver detects two additional activities (swimming and jumping, see the left panel of Figure 2), retraining the DL model disrupts its ability to recognize previously learned classes (sitting and standing). Conventional mitigation strategies (see the right panel of Figure 2) necessitate joint retraining with historical and new data to retain prior knowledge, but this approach remains impractical under privacy regulations and device storage constraints.

At present time, the critical problem is: **How can resource-constrained devices retain real-time recognition capabilities for historical activities without storing raw data?**

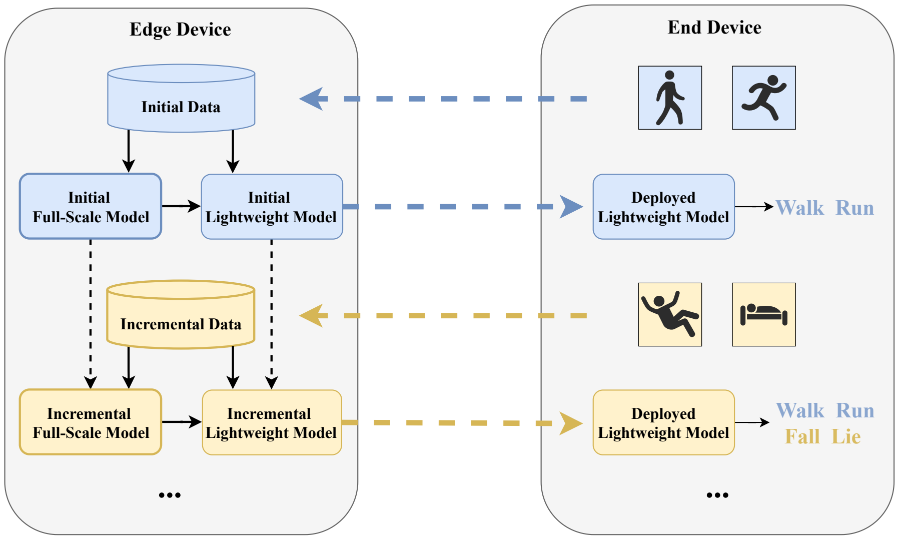

Edge Device
Initial Data
Initial Full-Scale Model
Initial Lightweight Model
Incremental Data
Incremental Full-Scale Model
Incremental Lightweight Model
...
End Device
Deployed Lightweight Model
Walk Run
Deployed Lightweight Model
Walk Run
Fall Lie
...

Fig. 3. Framework overview.

Table 1. List of Main Notations

Notation	Definition
D	The matrix of CSI
n	The temporal dimension of CSI
d	The channel dimension of CSI
X	The original stream
X'	The stream processed by Gaussian positional encoding
h	The number of attention heads
Q	The query matrix
K	The key matrix
V	The value matrix
t	The task index
P_K^t	The trainable key prefix parameters for task t
P_V^t	The trainable value prefix parameters for task t
$P_{K,frozen}^{t-1}$	The frozen key prefix parameters from the previous task
$P_{V,frozen}^{t-1}$	The frozen value prefix parameters from the previous task

4 System Design

4.1 Overall Framework

In order to solve the above problem, we propose WECAR, an end-edge collaborative inference and training framework, for WiFi-based continuous HAR, as shown in Figure 3. The main notations used in this article are summarized in Table 1. WECAR establishes a hierarchical architecture where edge devices execute model training, lightweight optimization, and updates, while end devices conduct efficient inference. The framework operates through two core models, i.e., the **full-scale model** (**FSM**) and the LWM. In the FSM, the original unoptimized model with complete network architecture and parameters are present. The LWM has instead a compressed variant derived from

FSM through lightweight. WiFi CSI is a noisy and highly time-dependent signal, and its feature distribution can shift significantly across environments and users, while practical deployments often cannot store historical data due to privacy and memory constraints. WECAR is preferable in this setting because MHSA-level prefixes provide task-specific context that steers temporal attention without overwriting frozen backbone representations, which helps retain old activities under exemplar-free incremental learning. In addition, selective retraining limits training cost on the edge side, and the distillation stage transfers historical prefix knowledge into a compact model, making the final system suitable for resource-constrained WiFi devices.

The interaction process between edge devices and end devices mainly includes four stages, i.e., the initial training stage, initial lightweight stage, incremental training stage, and incremental lightweight stage. In the initial training stage T_1, the edge device trains FSM on the first activity class set $\mathcal{C}_1$, updating all parameters (MHSA layers, MLP, and classifier). When the initial training stage is completed, MHSA layer parameters $W_{frozen}^{(MHSA)} = \{W_{frozen}^{Q}, W_{frozen}^{K}, W_{frozen}^{V}\}$ are fixed to preserve learned representations. When an incoming task exhibits a larger feature distribution shift, WECAR preserves old knowledge by freezing the backbone MHSA projections, but still maintains plasticity. Task specific trainable prefixes modulate attention for the new task, and the stability aware selective retraining automatically keeps more MLP neurons trainable because fewer neurons satisfy the stability criterion under larger shifts. In the initial lightweight stage, FSM serves as a teacher to distill knowledge into the student model LWM, which shares FSM's architecture but employs an MLP with reduced dimension. The MHSA relation distillation leverages three losses, i.e., attention relation distillation loss $\mathcal{L}_{AT}$, value relation distillation loss $\mathcal{L}_{VR}$ and logits distillation loss $\mathcal{L}_{log}$. Once the initial lightweight stage is completed, LWM's MHSA weights become aligned with FSM's frozen parameters. The optimized LWM is deployed to end devices for inference. In the incremental training stage at session t, for new class set $\mathcal{C}_t$, WECAR dynamically adapts FSM through MHSA expansion and MLP selective retraining. Specifically, MHSA expansion introduce trainable prefixes $P^t = \{P_K^t, P_V^t\}$ while retaining historical prefixes P_{frozen}^{t-1}, where $P_{frozen}^{t-1} = \{P_{K,frozen}^{t-1}, P_{V,frozen}^{t-1}\}$ represent the frozen key and value prefixes from the previous task. MLP selective retraining freezes stable neurons identified via activation analysis and updates only the unstable neurons. After completing task t, the previous task prefixes P_{frozen}^{t-1} are updated to incorporate the new prefixes from task t. Specifically, P_{frozen}^{t-1} is updated as follows: $P_{frozen}^{t} = \{P_{K,frozen}^{t}, P_{V,frozen}^{t}\}$, where $P_{K,frozen}^{t} = [P_K^t, P_{K,frozen}^{t-1}]$ and $P_{V,frozen}^{t} = [P_V^t, P_{V,frozen}^{t-1}]$, where [,] denotes the concatenation operation. We introduce prefixes only in the early stage. From Task 3 onward, we apply an accumulative in place update strategy. For each new task, we learn a small update prefix and merge it into the existing prefix bank under a fixed size budget. As a result, the prefix related parameter count stops increasing, and the training and inference costs do not grow with the number of classes. In the incremental lightweight stage, in order to reduce the number of parameters, the student model's MHSA layer does not retain the prefixes from previous tasks. Knowledge transfers from the teacher model FSM to the student model LWM takes place via prefix relation distillation loss L_P and logits distillation loss L_{log}. The lightweighted incremental model LWM is then processed and transmitted to the end device for inference. In the following, we will introduce WECAR in more detail.

4.2 Incremental Learning at the Edge Side

The proposed framework of FSM employs a transformer-based architecture, as shown in Figure 4, which mainly consists of three components: positional encoding layer, MHSA layer, and MLP layer.

4.2.1 Positional Encoding. Temporal sequence modeling plays a pivotal role in distinguishing reversible activities (e.g., sitting down vs. standing up) for WiFi-based HAR. Traditional absolute or

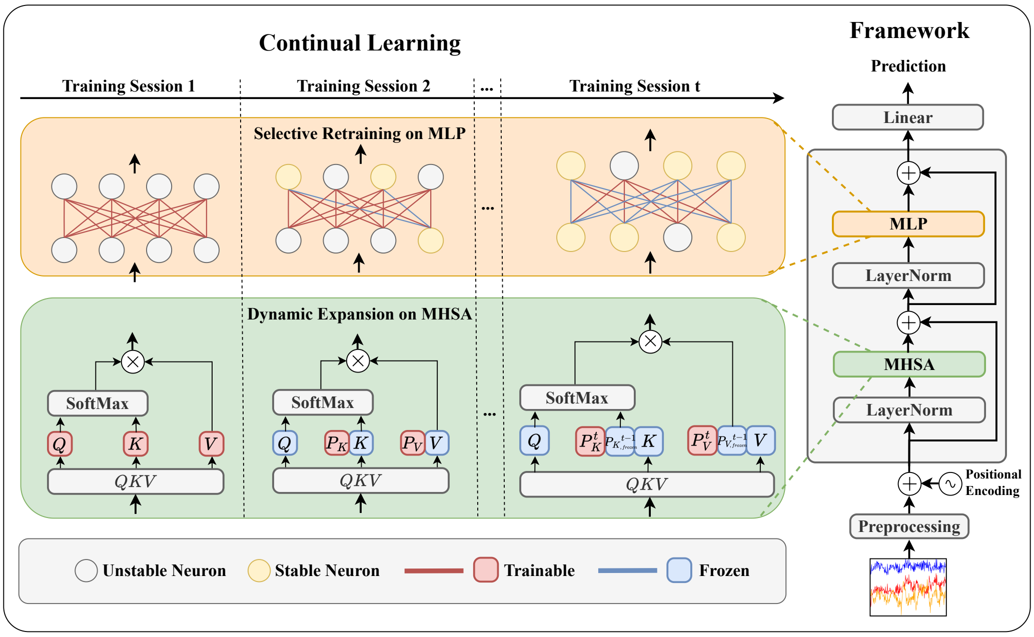


Fig. 4. Architecture of FSM. The right part contains the framework. The left part details how the model dynamically expands and selectively retrains during continual learning from training session 1 to training session *t*. As new tasks are introduced, the model dynamically expands with new prefixes in the MHSA layer. In the MLP, a selective retraining strategy is implemented to adjust neuron weights, preserving learned outcomes from stable neurons while updating unstable neurons to accommodate new tasks.

relative position encodings assign a unique and highly distinctive code to each individual point, which may induce noise. We use Gaussian range encoding to address this challenge, which is proposed in Ref. [39]. Gaussian range encoding refers to a set of positions that share similar encoding properties based on their temporal or spatial characteristics. This encoding allows a position to belong to multiple ranges simultaneously, and dynamically adjusts these ranges during training. Assume that there are G ranges, and denote by z_{ij} a random variable that represents the occurrence of position i ($i \in \{1, 2, \ldots, n\}$, where n represents the temporal dimension of CSI) belonging to the j-th range, $j \in \{1, 2, \ldots, G\}$, with

$$z_{ij} \sim N(\mu_j, \sigma_j), \tag{2}$$

where μ_j and σ_j are trainable parameters defining the j-th range. Denote by p_i the range distribution for position i, which is expressed as

$$p_i = \langle \frac{p^1(i)}{\zeta}, \frac{p^2(i)}{\zeta}, \ldots, \frac{p^G(i)}{\zeta} \rangle, \tag{3}$$

where ζ represents a normalization factor, and $p^g(i)$ represents the probability of the gth range for position i. Denote by v_j the value of the jth range. Given a value vector $v = \langle v_1, v_2, \ldots, v_K \rangle$, the expectation for position i is $p_i v^T$. Here, μ_j, σ_j, and v_j are unobserved latent variables, dynamically updated through neural network back propagation. Denote by β ($\beta \in \mathbb{R}^{n \times G}$) the normalized weight for the G Gaussian distributions, which is expressed as

$$\beta = \text{softmax}(B), \tag{4}$$

where each element b_{ij} of matrix B is expressed as

$$b_{ij} = -\frac{(i-\mu_j)^2}{2\sigma_j^2} - \log(\sigma_j). \tag{5}$$

Denote by X the original stream. Adding the range encoding to X generates a new range-biased stream, denoted by X', which is expressed as

$$X' = X + \beta E, \tag{6}$$

where E ($E \in \mathbb{R}^{G\times d}$, where d represents the spatial dimension of CSI) represents a set of learnable range encodings.

4.2.2 Task-Specific Dynamic Expansion on MHSA. The input sequence of MHSA X', $X' \in \mathbb{R}^{n\times d}$, has been processed using Gaussian range encoding, matching the original CSI dimensions of the input. For each attention head h_i, three linear transformation matrices, denoted by W_i^Q, W_i^K and W_i^V, are used to transform the input X' into queries (denoted by Q), keys (denoted by K) and values (denoted by V), respectively. The relationship between them is expressed as

$$\begin{cases} Q_i = X' W_i^Q, \\ K_i = X' W_i^K, \\ V_i = X' W_i^V. \end{cases} \tag{7}$$

Each attention head subsequently employs the scaled dot-product attention mechanism to calculate the corresponding attention weights. Specifically, by computing the dot product between the queries and keys, applying softmax to obtain the weights, and then using these weights to perform a weighted sum of the values, the final attention output Attention() is produced. The formula is:

$$\text{Attention}(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{\frac{d}{h}}}\right) V_i, \tag{8}$$

where $\frac{1}{\sqrt{\frac{d}{h}}}$ is the scaling factor to avoid large inner product values that could cause gradient vanishing issues. Finally, the results from all heads are concatenated, and then transformed using the linear transformation matrix W_O, $W_O \in \mathbb{R}^{d\times d}$, to derive the final output, denoted by MultiHead(Q, K, V), which is expressed as

$$\text{MultiHead}(Q, K, V) = [\text{head}_1, \ldots, \text{head}_h] W_O, \tag{9}$$

where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$.

The objective of acquiring new knowledge while retaining old knowledge without storing exemplars can be achieved by introducing task-specific prefixes to the MHSA layers. This approach facilitates knowledge transfer between tasks while minimally altering the model's original parameters. By keeping the model's parameters fixed and updating only the prefixes, the system effectively prevents catastrophic forgetting while preserving adaptability. Specifically, each MHSA layer comprises h attention heads, and the addition of prefixes to these layers enables CIL for new tasks. Concretely, we have

$$\begin{cases} W_i^{K'} = \left[P_K^t, P_{K,frozen}^{t-1}, W_{frozen}^K\right], \\ W_i^{V'} = \left[P_V^t, P_{V,frozen}^{t-1}, W_{frozen}^V\right]. \end{cases} \tag{10}$$

The output of head i in the self-attention layer is expressed as

$$\text{head}_i = \text{Attention}\left(Q_i, K_i{}', V_i{}'\right), \tag{11}$$

where $Q_i = X'W^Q_{frozen}$, $K_i{}' = X'W_i^{K'}$, and $V_i{}' = X'W_i^{V'}$.

To effectively integrate prior knowledge with new information, the model sequentially concatenates the new trainable prefixes P^t_K and P^t_V with the previously frozen prefixes $P^{t-1}_{K,frozen}$ and $P^{t-1}_{V,frozen}$. These concatenated prefixes are then merged with the consistently frozen weights W^K_{frozen} and W^V_{frozen}. This concatenation ensures a seamless transition and retention of learned features across tasks. Starting from the third task, the newly learned task-update prefix is merged into the existing prefix parameters via an accumulative in-place update instead of allocating additional prefix tokens, thereby keeping the total number of prefix parameters in the MHSA layer constant.

While prefixes are effective for CIL tasks, their random initialization can lead to unstable performance due to the variability in the initial weights. To address this issue, we draw inspiration from the parallel attention design [40], which employs a parallel adapter to stabilize the prefixes. Specifically, given an input sequence X', the prefix generation is expressed as

$$P = Adapter(X') = Tanh(X'W_{down})W_{up}, \tag{12}$$

where P represents all prefixes, $Tanh$ represents the activation function, and W_{down} and W_{up} denote the parameters of the parallel adapter's scaling layers. Specifically W_{down} denotes a linear transformation layer that reduces the dimensionality of X', and W_{up} denotes another linear transformation that expands the transformed output.

4.2.3 Stability-Aware Selective Retraining on MLP. While the MHSA layer extracts the temporal features of CSI through linear transformations, the subsequent MLP layer enhances the representational capacity through non-linear mappings. To mitigate catastrophic forgetting in incremental learning, we propose a stability-aware selective retraining approach based on neuron activation for MLP layers. This method involves three main steps, i.e., calculating each neuron's average activation value, identifying stable neurons, and generating freeze masks for parameter updates. This procedure is executed independently across all MLP layers, ensuring localized adaptation while preserving critical historical knowledge.

Given a training set, denote by $\bar{a}^{(l)}_p$ the average activation value for each neuron in the l-th layer, which is expressed as

$$\bar{a}^{(l)}_p = \frac{1}{B}\sum_{q=1}^{B} a^{(q,l)}_p, \tag{13}$$

where B, l, and $a^{(q,l)}_p$ denote the size of the training set, the layer index, and the activation value of the pth neuron in the lth layer for the qth sample, respectively. Then, by comparing the current activation values with those from the previous task, denote by $S^{(l)}$ the set of stable neurons in each layer, which is expressed as

$$S^{(l)} = \left\{p \mid \|\bar{a}^{(l,t)}_p - \bar{a}^{(l,t-1)}_p\|_2 \leq \epsilon\right\}, \tag{14}$$

where $S^{(l)}$ accumulates stable neurons across the entire training period, ϵ denotes a predefined threshold, and $\bar{a}^{(l,t)}_p$ and $\bar{a}^{(l,t-1)}_p$ represent the average activation values for the current and previous tasks in the lth layer, respectively. The threshold ϵ specifies the tolerance for cross-task activation difference. A neuron is defined as stable if the ℓ_2 distance between its dataset-level mean activation in tasks t and t–1 is not greater than ϵ, and its associated parameters are frozen; otherwise it is treated as unstable and remains trainable. Finally, we generate the freeze mask set $M^{(l)} = \{M_W^{(l)}, M_b^{(l)}\}$ based on the set of stable neurons $S^{(l)}$ for each MLP layer, where $M_W^{(l)}$ and $M_b^{(l)}$ denote mask matrices corresponding to the weight matrix $W^{(l)}$ and bias vector $b^{(l)}$ in the lth layer, respectively. Matrices $M_W^{(l)}$ and $M_b^{(l)}$ are initialized with values set to one. Subsequently, for stable neurons in the set $S^{(l)}$, the corresponding values in $M_W^{(l)}$ and $M_b^{(l)}$ are set to zero.

During backpropagation, these masks are applied across all layers by identifying positions where $M_W^{(l)}$ and $M_b^{(l)}$ have a value of zero. At these positions, the corresponding gradients of $W^{(l)}$ and $b^{(l)}$ are set to zero, ensuring that these parameters are not updated. Parameters that are not frozen continue to be updated normally. Specifically, for the weight matrix W, the gradient update rule is expressed as

$$\nabla W_{ij} = \begin{cases} 0, & \text{if } (i,j) \in M^{(l)}, \\ \nabla W_{ij}, & \text{otherwise.} \end{cases} \tag{15}$$

For the bias vector b, the gradient update rule is expressed as

$$\nabla b_i = \begin{cases} 0, & \text{if } i \in M^{(l)}, \\ \nabla b_i, & \text{otherwise.} \end{cases} \tag{16}$$

In this manner, the approach reduces forgetting by preserving the weights of stable neurons while preventing excessive computational load during new task learning. The freezing strategy targets neurons that exhibit stable activations across tasks, indicating that they encode reusable and task-invariant representations that should be preserved for mitigating forgetting. Neurons with higher task-dependent variation remain trainable and provide adequate capacity for adaptation to incoming classes. Moreover, task adaptation is primarily supported by attention modulation through prefixes. This module complements the partially frozen MLP and maintains sufficient plasticity under long incremental sequences.

4.3 Lightweight at Edge Side

Knowledge distillation is one of the model lightweighting techniques, which can transfer the knowledge from a large teacher model *Tea* to a smaller student model *Stu*. The student model is trained to emulate the behavior of the teacher model, where the behavior functions of the teacher and student models are represented as f_{Tea} and f_{Stu}, respectively. These functions are responsible for transforming the network input into meaningful representations, typically defined by the output of any layer in the network. In transformer-based distillation, the output of the MHSA layer, the MLP layer, or other intermediate representations (such as the attention matrix A) can be used as the behavior function.

The fundamental objective of knowledge distillation is to minimize the representational discrepancy between the teacher and student models in feature space. This is formalized through the following optimization objective

$$\mathcal{L}_{\text{KD}} = \sum_{x \in X} L(f_{Stu}(x), f_{Tea}(x)), \tag{17}$$

where X represents the training dataset, L denotes the loss function, $f_{Stu}(x)$ and $f_{Tea}(x)$ are the feature representations extracted by the student and teacher models for input x, respectively. A prevalent choice for the loss function is **mean squared error** (**MSE**), which quantifies the Euclidean distance between feature vectors, and is expressed as

$$\text{MSE}(f_{Stu}(x), f_{Tea}(x)) = \|f_{Stu}(x) - f_{Tea}(x)\|_2^2, \tag{18}$$

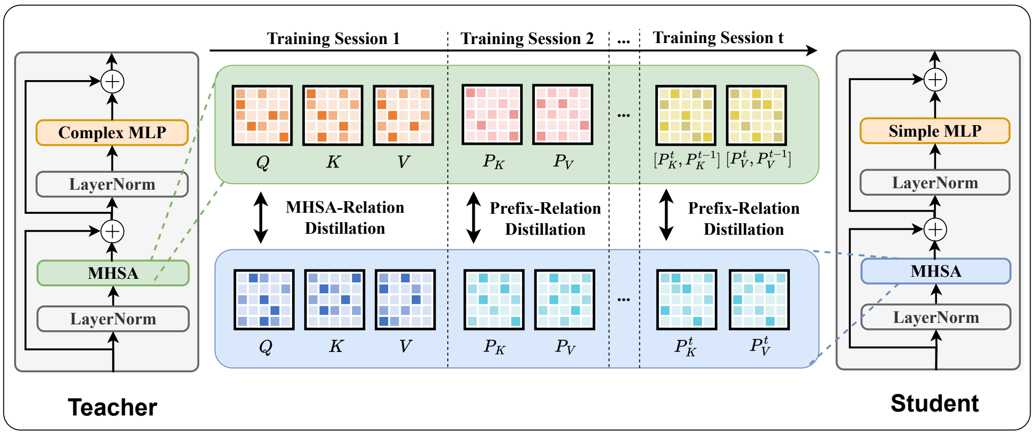


Fig. 5. Overview of dual-phase distillation. The left side illustrates the teacher model, which includes complex MLP layers and MHSA layer with specific prefixes. The right side depicts the student model, which employs simplified MLP layer and MHSA layer without retaining historical prefixes. The middle section provides a detailed overview of the distillation process from the initial training stage to the t-th training stage. In the initial lightweight phase, knowledge transfer is achieved through MHSA relation distillation. In the incremental lightweight phase, prefix relation distillation compresses the teacher model's historical prefix knowledge into the student model.

where $\|\cdot\|_2$ denotes the Euclidean norm. By minimizing this loss function, the student model will attempt to make its feature outputs as close as possible to those of the teacher model, thereby better mimicking the teacher's behavior.

During training session evolution, the teacher model accumulates multi-scale spatiotemporal features through task-specific prefixes. Traditional single-phase distillation struggles to transfer historical task knowledge and new task features. The attention module captures cross-timestep global dependencies, while prefixes encode distributions across tasks. We therefore propose a dual-phase distillation mechanism, as shown in Figure 5. The initial phase consolidates fundamental spatiotemporal perception via MHSA relation distillation, followed by prefix relation distillation for task knowledge. This strategy preserves the core transformer attention mechanism's representational generalization while at the same time compressing dynamically expanded parameters to a deployable scale. Such dual optimization achieves lightweight deployment without compromising historical task knowledge. The detailed implementation is as follows.

4.3.1 MHSA Relation Distillation. In the initial lightweight stage, the student model learns the behavior of the teacher model through attention relation distillation and value relation distillation. The goal is to transfer the knowledge of the teacher model to the student model via knowledge distillation. The first key component of the distillation process is the attention relation distillation loss $\mathcal{L}_{AT}$. In this stage, the student model aims to mimic the attention matrix of the teacher model. The attention matrix of each attention head in the teacher model is compared with the corresponding attention matrix in the student model. By using the MSE to characterize the difference between them, the goal is to ensure that the student model's attention distribution matches the teacher model as closely as possible. Denote by $\mathcal{L}_{AT}$ the attention relation distillation loss function, which is expressed as

$$\mathcal{L}_{AT} = \frac{1}{h} \sum_{i=1}^{h} \mathrm{MSE}\left(A_i^{Tea}, A_i^{Stu}\right), \tag{19}$$

where A_i^{Tea} and A_i^{Stu} denote the attention matrices of the teacher and student models for the i-th attention head, respectively.

To further align intermediate representations, we introduce a value relation distillation loss, denoted by $\mathcal{L}_{VR}$, which uses the relationship between the values in the attention module to guide the student's training. The value relation distillation loss function is defined as

$$\mathcal{L}_{VR} = \frac{1}{h}\sum_{i=1}^{h} \text{MSE}\left(V_i^{Tea}, V_i^{Stu}\right), \tag{20}$$

where V_i^{Tea} and V_i^{Stu} denote the attention values for the teacher and student models for the ith attention head, respectively. By introducing the relationship between the values, the student model is guided to deeply emulate the self-attention behavior of the teacher model.

4.3.2 Prefix Relation Distillation. In the incremental lightweight stage, to reduce the parameter size and computational complexity of the student model, the MHSA layer of the student model does not store the prefix information from the previous tasks. Instead, the student model learns the knowledge obtained from the teacher model through a prefix relation distillation loss function. On the teacher side, the effective prefix used by MHSA at session t is the concatenation of the newly learned prefixes and all frozen historical prefixes (i.e., $[P^t, P^{t-1}_{\text{frozen}}]$). This concatenated prefix set is exactly the entity that participates in the key and value construction for attention in the teacher. Therefore, it represents the teacher's accumulated temporal memory across tasks, rather than only the current task information. On the student side, we deliberately do not keep multiple per task prefix sets to avoid growing storage and inference overhead on end devices. Instead, at each session t, the student maintains only one prefix set $P^{t,\text{Stu}}$. We train it to match the teacher's concatenated prefixes via MSE, so that $P^{t,\text{Stu}}$ becomes a compact summary that approximates the teacher's overall prefix conditioned attention context. In this way, historical task knowledge is absorbed into the single student prefix implicitly, even though explicit historical prefixes are not stored in the student. This design enables constant memory usage while preserving cross task temporal representations learned by the teacher. Denote by $\mathcal{L}_P$ the specific loss function expression, which is expressed as

$$\begin{aligned}\mathcal{L}_P = \text{MSE}&\left(\left[P_K^{t,Tea}, P_{K,frozen}^{t-1,Tea}\right], P_K^{t,Stu}\right)\\ +\text{MSE}&\left(\left[P_V^{t,Tea}, P_{V,frozen}^{t-1,Tea}\right], P_V^{t,Stu}\right),\end{aligned} \tag{21}$$

where $P_K^{t,Tea}$ and $P_V^{t,Tea}$ represent the key and value prefixes from the teacher model for task t, and $P_{K,frozen}^{t-1,Tea}$ and $P_{V,frozen}^{t-1,Tea}$ represent the frozen key and value prefixes from the teacher model for task T_{t-1}, $P_K^{t,Stu}$ and $P_V^{t,Stu}$ represent the key and value prefixes in the student model for task t, respectively.

Although both optimization objectives are formulated based on MSE loss, they act on different supervision targets and convey highly complementary information. MHSA relation distillation achieves precise matching of self-attention behavioral patterns by aligning attention matrices and value features. In contrast, prefix relation distillation accomplishes task conditioning alignment by matching the constructed teacher prefix sequence with the student prefix sequence, thus enabling efficient compression of task knowledge without retaining historical prefixes.

4.4 Empty Frame Data Processing and Inference at End Side

4.4.1 Empty Frame Data Processing. In the data collection phase, a multi-antenna WiFi receiver collects CSI in real-time. However, due to environmental interference or signal instability, some subcarriers may exhibit missing measurements at specific time points, resulting in empty frames

Table 2. Statistics of the Datasets

Dataset	Class	Size	Train	Test
WiAR	16	270×90	384	96
MMFi	27	10×342	2160	540
XRF	48	50×270	672	288

The size of each dataset is $n \times d$.

within the CSI sequence. To address these gaps, we employ linear interpolation to fill the missing values. Specifically, for each subcarrier in matrix $\mathbf{D}$, if a value is missing at time point t, we estimate the missing value using the linear interpolation formula

$$\mathbf{D}(t,i) = \mathbf{D}(t_1,i) + \frac{(\mathbf{D}(t_2,i) - \mathbf{D}(t_1,i))(t-t_1)}{(t_2-t_1)}, \tag{22}$$

where t_1 and t_2 are the nearest valid time points before and after t, and $\mathbf{D}(t_1,i)$ and $\mathbf{D}(t_2,i)$ are the corresponding CSI measurements, respectively.

4.4.2 Inference. During the inference phase on the end device, the deployed LWM identifies human activities based on sensing environments. As the system continuously interacts with edge devices, it is capable of recognizing an expanding range of activities.

4.5 Computational Complexity of WECAR

The computational complexity of Gaussian range encoding is $O(nGd)$. The computational complexity of the linear projections for queries, keys, values, and the output projection is $O(nd^2)$. The computational complexity of the feed-forward network is $O(ndh)$. The computational complexity of MHSA is $O((n+2p)^2d)$. Thus, the computational complexity of WECAR is $O(nGdT + nd^2T + (n+2p)^2dT + ndhT)$, which is a polynomial function with respect to the number of tasks T, sequence length n, feature dimension d, and feed-forward hidden width h.

5 Experiment Evaluation

5.1 Datasets and Settings

We implement WECAR on heterogeneous hardware. Specifically, we employ the Nvidia Jetson Nano, a compact but powerful AI development kit with a quad-core Cortex-A57 processor, and a 128-core GPU and 2GB memory [41], as the edge device, and the ESP32C3 as the end device, respectively. We use the existing dataset to simulate the data collected by the ESP32. To validate the effectiveness of WECAR, we conduct experiments on three widely recognized public datasets, namely WiAR [42], MMFi [43], and XRF [44] datasets, which offer a broader range of categories. The statistics of these datasets are summarized in Table 2.

(1) **WiAR**: WiAR consists of 480 CSI samples, evenly distributed across 16 distinct classes. We divided the dataset into training and testing subsets at a 4:1 ratio. After data processing, the sample size is 270 x 90. Additionally, we organized WiAR into two task types: short and long, with the latter having more number of tasks. The short task set includes five tasks: the first task covers 8 classes, while the following 4 tasks cover 2 classes each. In contrast, the long task set comprises 8 tasks, with each task consistently including 2 classes.

(2) **MMFi**: MMFi comprises 2700 CSI samples, evenly distributed across 27 classes. After data processing, the sample size is 10 x 342. In MMFi, the short task category includes a total of 6 tasks. The first task covers 12 classes, while each of the next five tasks covers 3 classes. In contrast, the long task category consists of 9 tasks, with each task handling 3 classes.

(3) **XRF**: XRF initially includes 55 classes, with 7 dedicated to dual-person actions. We exclude these dual-person classes due to our focus on single-person activities, leaving 48 classes and 960 CSI samples. Each class contains 20 samples, with 14 samples per class allocated for training and the remaining 6 used for testing. After data processing, the sample size is 50 x 270. In XRF, the short task category consists of 5 tasks: the first task covers 24 classes, while each of the subsequent 4 tasks contains 6 classes. In contrast, the long task category is organized into 8 tasks, each responsible for analyzing 6 classes.

5.2 Baselines

We compare FSM with five existing EFCIL methods:

(1) **LWF** [10]: LWF applies knowledge distillation to prevent forgetting old tasks. It uses the original model's predictions as soft targets during new task training.

(2) **PASS** [45]: PASS reduces catastrophic forgetting in incremental learning by combining prototype augmentation and self-supervised learning, effectively retaining past knowledge while adapting to new tasks.

(3) **R-DFCIL** [25]: R-DFCIL synthesizes data for previous classes using model inversion and applies relation-guided representation learning to minimize the domain gap between synthetic and real data.

(4) **PRD** [26]: PRD introduces a new distillation loss to maintain the relevance of class prototypes during new task learning, ensuring that the current model maintains relevance to the prototypes of previous tasks while adapting to new tasks.

(5) **ConTraCon** [30]: ConTraCon modifies the MHSA layer weights via convolutional operations to adapt the transformer architecture for new tasks.

5.3 Evaluation Metrics

We use two metrics, i.e., the average accuracy and average forgetting to measure the performance of WECAR. The accuracy after each task, denoted by A_t, represents the accuracy over all classes learned up to and including the tth task. Subsequently, the average accuracy across all tasks, represented by $\bar{A}$, is expressed as

$$\bar{A} = \frac{1}{N}\sum_{t=1}^{N} A_t,$$

where N represents the number of tasks. The average forgetting measure [46] is used to estimate the forgetting of previous tasks. For each task t, the forgetting measure of predicting previous task k is denoted by f_k^t, which is expressed as

$$f_k^t = \max_{z\in\{1,\ldots,k-1\}} (\alpha_{z,t} - \alpha_{k,t}),$$

where $\alpha_{m,j}$ represents the accuracy of task j after training task m. The average forgetting measure represents the forgetting measure of the last task, denoted by $\bar{F}$, which is expressed as

$$\bar{F} = \frac{1}{N-1}\sum_{k=1}^{N-1} f_k^N.$$

5.4 Implementation Details

Our method is implemented by PyTorch [47] and trained on Nvidia Jetson Nano GPU with 2GB memory. The optimizer chosen is Adam [48], with an initial learning rate of 0.001 and a batch size of 4. The model's training cycle is set to 50 epochs. After completing the lightweighting process, the model is converted into the ONNX format. Subsequently, the ONNX model is processed

Table 3. Comparison of Average Accuracy $\bar{A}$ (%) Between WECAR and Five Other Methods on WiAR, MMFi, and XRF Datasets, for Short Task ($N = 5$ or $N = 6$) and Long Task ($N = 8$ or $N = 9$)

Method	WiAR			MMFi			XRF		
	Params	$N = 5$	$N = 8$	Params	$N = 6$	$N = 9$	Params	$N = 5$	$N = 8$
LWF	18.52M	41.25 ± 2.12	39.76 ± 2.05	18.52M	33.85 ± 1.89	30.69 ± 1.91	18.52M	32.02 ± 1.78	29.76 ± 2.02
PASS	11.32M	60.74 ± 1.92	40.72 ± 2.12	11.32M	45.54 ± 2.05	39.69 ± 2.04	11.32M	47.57 ± 1.92	35.74 ± 1.92
R-DFCIL	12.81M	59.62 ± 2.25	58.80 ± 2.32	12.81M	48.90 ± 2.10	47.88 ± 2.15	12.81M	49.68 ± 1.84	44.89 ± 2.01
PRD	11.75M	63.14 ± 1.51	59.04 ± 2.02	11.75M	53.69 ± 2.03	52.45 ± 2.18	11.75M	54.02 ± 2.12	51.68 ± 2.05
ConTraCon	3.60M	-	50.76 ± 2.08	2.50M	-	44.12 ± 2.06	2.10M	-	40.54 ± 2.07
WECAR	**1.33M**	**87.23 ± 2.93**	**86.97 ± 1.42**	**0.79M**	**74.46 ± 1.49**	**67.11 ± 0.50**	**0.58M**	**65.71 ± 1.37**	**64.61 ± 1.19**

The term "Params" refers to the initial number of model parameters, measured in millions. None of the methods utilize real historical data for replay. R-DFCIL employs synthetic data to simulate the replay data.

Table 4. Comparison of the Average Forgetting Measure $\bar{F}$ (%) Between WECAR and Five Other Methods on the WiAR, MMFi, and XRF Datasets, with Short and Long Tasks (Lower Values Indicate Better Performance)

Method	WiAR		MMFi		XRF	
	$N = 5$	$N = 8$	$N = 6$	$N = 9$	$N = 5$	$N = 8$
LWF	30.32	30.21	31.05	32.57	33.53	30.01
PASS	24.14	29.34	22.02	25.17	21.09	29.03
R-DFCIL	23.24	23.96	20.98	24.61	21.35	26.83
PRD	20.90	22.02	19.81	19.24	24.97	20.69
ConTraCon	-	27.56	-	28.98	-	25.86
WECAR	**15.36**	**13.19**	**17.94**	**19.09**	**20.29**	**18.69**

to a binary file, which is transferred to the ESP32C3 and flashed onto the device to execute inference.

During training, we set the number of Gaussian distributions in the positional encoding to $G = 10$ as a lightweight tradeoff between temporal resolution and overhead. We initialize the Gaussian centers μ_j by uniformly spacing them along the temporal dimension so that the ranges cover the whole sequence, and we scale this uniform grid to each dataset length. For WiAR, μ_j ranges from 13.5 to 256.5 with a step size of 27. For MMFi, μ_j ranges from 0.5 to 9.5 with a step size of 1. For XRF, μ_j ranges from 2.5 to 47.5 with a step size of 5. We set the Gaussian width to $\sigma_j = 8$ on all datasets to ensure sufficient overlap between neighboring ranges and to avoid dataset specific tuning. The number of stacks in the module is set to 1. The input dimensions for the three datasets are set to 90, 342, and 270, respectively, while maintaining a consistent number of heads at 9 for each, and employing a dropout rate of 0.1.

5.5 Result Comparison

5.5.1 Performance Comparison. Tables 3 and 4 summarize the results of WECAR compared with baselines in terms of the average accuracy $\bar{A}$ and the average forgetting $\bar{F}$, respectively. The two tables reveal that WECAR significantly outperforms the other baselines across all the datasets. Specifically, on the WiAR dataset with long task sequences ($N = 8$ or $N = 9$), the average accuracy of WECAR surpasses that of the other methods by nearly 30%. On the MMFi dataset with the short task sequences ($N = 5$ or $N = 6$), the average accuracy improvement exceeds 20%. Especially compared with LWF, the advantage of WECAR reaches 40%. In addition, WECAR achieves a forgetting rate of less than 21% for both short and long tasks across the three datasets, and outperforms the other methods. Notably, the baseline methods are evaluated in their original non-lightweighted forms, while WECAR adopts a lightweight design. Even after reducing parameters through lightweighting, WECAR still performs better than the non-lightweight baselines. This means if we would compare WECAR with the lightweight versions of the other methods, it should perform even better. Moreover, WECAR does well in both plasticity and stability. The reason is that in WECAR, we utilize MHSA and positional encoding, which are particularly well-suited to the characteristics of time-series data, such as the patterns and intensity of signal changes in CSI. These features pose challenges to traditional image-based network architectures, like Resnet, which primarily is optimized for spatial feature extraction and struggles with the dynamic characteristics of time-series data. In contrast, other methods face limitations in adapting to the dynamic environment of CSI. For instance, the forgetting-preventing approach of LWF performs poorly in dynamically changing environments. R-DFCIL's synthetic data approach fails to accurately capture the true characteristics of CSI. PRD attempts to mitigate forgetting by maintaining relationships between class prototypes, but the high dynamics and complexity of CSI may render this prototype-based method ineffective. ConTraCon, although using the transformer architecture to adapt to new tasks, relies heavily on its attention mechanism. If this mechanism fails to capture the temporal and frequency domain characteristics of CSI, its performance becomes suboptimal. Moreover, its entropy-based task prediction, which relies on image enhancement techniques, is unsuitable for CSI, as temporal delays and frequency shifts in CSI do not correspond to visual changes.

5.5.2 Comparison of Model Parameter Counts. As shown in Table 3, WECAR demonstrates a significant reduction in model parameters compared with the other methods. Specifically, WECAR requires only one-third of the parameter count of ConTraCon and at least eight times fewer parameters than the other methods. Notably, on the MMFi and XRF datasets, WECAR's parameter count is as low as one-tenth of those of the other methods, excluding ConTraCon. This advantage indicates that WECAR is particularly well-suited for deployment on edge devices, such as WiFi-based HAR terminal.

5.5.3 Comparison of Accuracy for Each Task. In Figure 6, we compare the accuracy of WECAR with other methods across each task on the three datasets. We observe that the accuracy for the initial task is generally comparable among all methods, with the exception of WECAR, which exhibits slightly lower performance on the short and long tasks of the XRF dataset. However, the accuracy of WECAR significantly outperforms the other methods in subsequent tasks. This demonstrates that our method achieves a balance between knowledge forgetting and acquisition when dealing with CSI. In addition, the performance gap between WECAR and the other methods widens as the number of tasks increases. For example, in Figure 6(e), at the fifth task, the gain of WECAR compared with PRD is approximately 10%, while by the ninth task, the improvement grows to 20%. This trend demonstrates the effectiveness of WECAR in handling extended task sequences, highlighting its robustness in continually sensing.

5.6 Ablation Test

5.6.1 Effect of Dual-Phase Distillation. Figure 7 illustrates the task-wise accuracy comparison with and without dual-phase distillation, while Table 5 presents the model metrics and average accuracy comparison. Here, CFM denotes the results without dual-phase distillation, which maintains complete parameter integrity during continual learning. It can be observed that WECAR significantly reduces parameters, model size, and GPU memory usage while maintaining high classification accuracy. Across the WiAR, MMFi, and XRF datasets, WECAR achieves a 60%–70% reduction in parameter count and around a 25% decrease in GPU memory consumption compared with CFM, with a classification accuracy dropping of only 2%–4% for short and long tasks. These

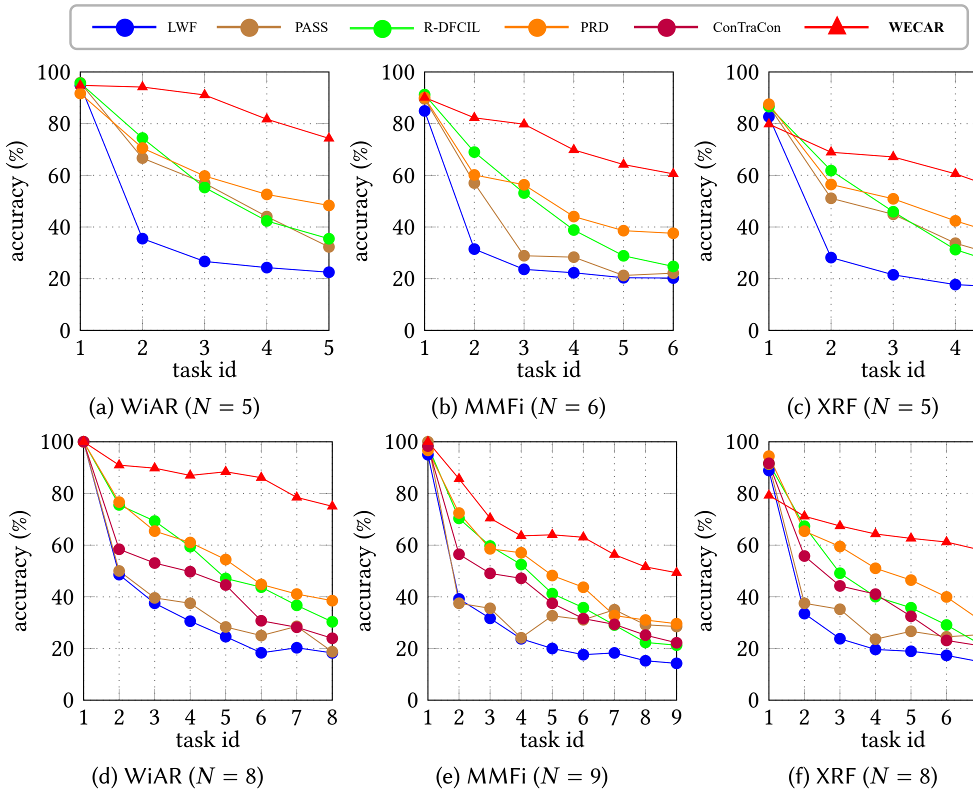


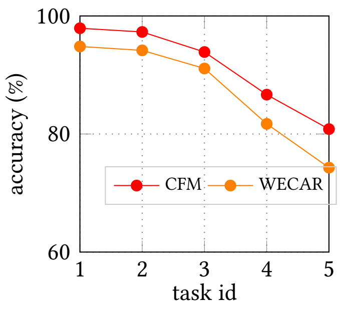
Fig. 6. Accuracy comparison between WECAR and the five other methods across each task. The x-axis represents task t, and the corresponding value of the y-axis is the accuracy, i.e., A_t.

findings highlight the effectiveness of the proposed dual-phase distillation in achieving substantial model simplification and resource efficiency, making WECAR well-suited for deployment on resource-constrained edge devices.

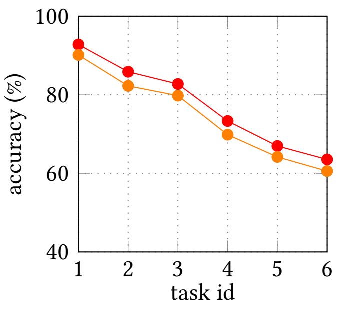
5.6.2 *Effect of Dynamic Expansion and Selective Retraining.* Table 6 demonstrates that the two strategies, i.e., dynamic expansion on MHSA and selective retraining on MLP, significantly enhance the performance of WECAR in long-term EFCIL ablation experiments. More specifically, dynamic expansion significantly enhances the accuracy of the last task and average task accuracy across all the datasets. It achieves this by adding trainable prefixes to the MHSA layers, which enhances the model's robustness against forgetting and its adaptability to new tasks. In contrast, while the gains from selective retraining are less dramatic, it still contributes to overall the performance enhancement, particularly by stabilizing trained parameters in specific scenarios.

5.6.3 *Effect of Parallel Adapter.* Figure 8 shows that the parallel adapter initialization (Prefix-A) outperforms zero (Prefix-Z) and random (Prefix-R) initializations in WECAR. Specifically, Prefix-A achieves an average accuracy of 71.97%, compared with 69.00% for Prefix-R and 68.30% for Prefix-Z. These results underscore the effectiveness of parallel adapter initialization in enhancing prefix handling and overall the model performance.

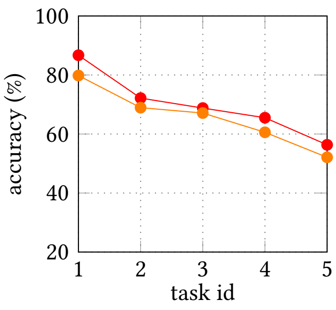
5.6.4 *Effect of Gaussian Positional Encoding.* WiFi CSI is a short, highly dynamic time series with correlated samples, noise, and speed variations. Sinusoidal positional encoding mismatches



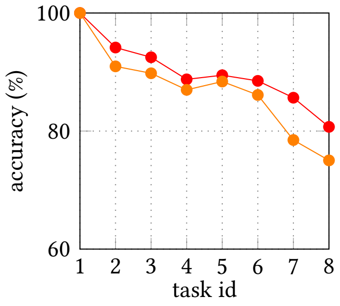
(a) WiAR ($N = 5$)



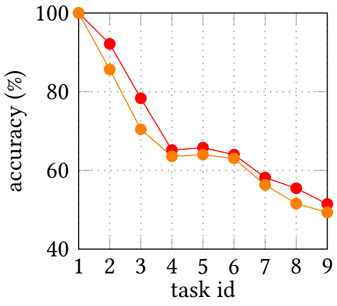
(b) MMFi ($N = 6$)



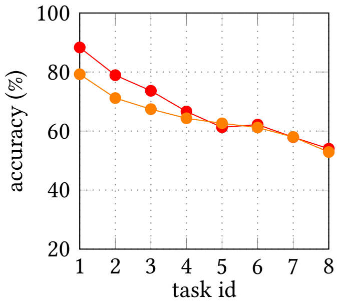
(c) XRF ($N = 5$)



(d) WiAR ($N = 8$)



(e) MMFi ($N = 9$)



(f) XRF ($N = 8$)

Fig. 7. Accuracy comparison of WECAR and CFM across each task. The x-axis represents task t, and the corresponding value of the y-axis is the accuracy, i.e., A_t.

Table 5. Comparison of Model Metrics With and Without Dual-Phase Distillation

Dataset		CFM	WECAR
WiAR	Params	3.35M	1.33M
	Size	12.47MB	4.95MB
	GPU	342.55MB	257.67MB
	$N = 5$	91.32 ± 2.22	87.23 ± 2.93
	$N = 8$	89.97 ± 1.31	86.97 ± 1.42
MMFi	Params	1.92M	0.79M
	Size	7.15MB	3.03MB
	GPU	20.99MB	15.23MB
	$N = 6$	77.54 ± 2.02	74.46 ± 1.49
	$N = 9$	70.06 ± 1.41	67.11 ± 0.50
XRF	Params	1.50M	0.58M
	Size	5.59MB	2.16MB
	GPU	94.09MB	67.18MB
	$N = 5$	69.91 ± 1.42	65.71 ± 1.37
	$N = 8$	67.87 ± 1.65	64.61 ± 1.19

"Size" indicates the model file storage, quantified in megabytes (MB). "GPU" denotes the peak memory consumption during training, measured in MB. Notably, the GPU metric for WECAR corresponds to its memory usage without employing CFM as a teacher model in dual-phase distillation.

Table 6. Ablation Study of Two Strategies for Long Tasks Across the Three Datasets

Dataset	Strategy 1	Strategy 2	A_N	$\bar{A}$
WiAR	×	×	36.45	52.08
	✓	×	51.03	67.70
	×	✓	44.79	59.37
	✓	✓	**78.58**	**89.85**
MMFi	×	×	25.91	46.27
	✓	×	38.87	60.15
	×	✓	34.24	54.78
	✓	✓	**53.52**	**71.97**
XRF	×	×	22.56	40.96
	✓	×	41.66	53.12
	×	✓	30.90	48.60
	✓	✓	**48.71**	**65.79**

Strategy 1 represents dynamic expansion on MHSA. Strategy 2 represents selective retraining on MLP.

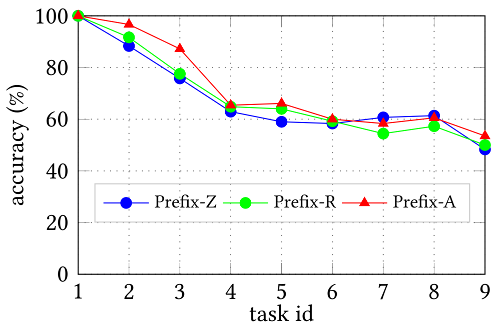


Fig. 8. Ablation study examining the impact of parallel adapters on long tasks using the MMFi dataset.

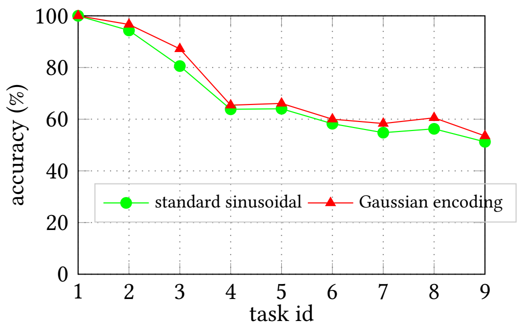


Fig. 9. Ablation study examining the impact of Gaussian encoding on long tasks using the MMFi dataset.

non-periodic CSI dynamics, while learnable embeddings induce excess noise sensitivity. Gaussian range encoding enables soft representations and adjacent weight sharing, boosting temporal smoothness and robustness against shifts. Figure 9 compares the performance of Gaussian positional encoding with that of Sinusoidal positional encoding. It can be observed that Gaussian positional encoding achieves superior performance.

6 Conclusion

We have presented WECAR, an end-edge collaborative inference and training framework for WiFi-based continuous HAR that simultaneously addresses dynamic environment adaptation and resource constraints in edge deployment. The proposed architecture features two key technical contributions: (1) a parameter-efficient dynamic continual learning mechanism that enables environment-aware model evolution, and (2) a dual-phase knowledge distillation strategy optimized for resource-constrained edge devices. Implemented on heterogeneous hardware, our framework demonstrates superior performance with 10% average accuracy improvement across all the tasks while maintaining a forgetting rate below 21%, outperforming state-of-the-art methods by significant margins. What's more, WECAR achieves these advancements with only 33% parameter count in comparison to the baseline approaches, and cutting model footprints of baseline models by 90% in specific scenarios. Extensive experiments validate its practical viability for privacy-sensitive applications, particularly demonstrating robust operation under strict computational and memory constraints. The presented framework establishes new design paradigms for adaptive edge intelligence systems in ubiquitous sensing applications. The framework can be applied to other sensing modalities (such as millimeter-wave and IMU sensors) by replacing modality specific preprocessing and task heads, while keeping the continual adaptation and distillation pipeline unchanged. WECAR can be extended to multi-person continuous sensing scenarios by adding a multi-person feature disentanglement module at the front end and reconstructing the prediction head to support multi-label recognition. Meanwhile, the original prefix-based continual adaptation and selective retraining strategies remain unchanged. With this extended architectural design, some preliminary experiments show an accuracy of 82.42% for the XRF55 dataset (including 7 dual-person activity classes). More thorough investigations and experiments will be presented in our future work.

References

[1] Yufei Wang, Qixin Wang, Guanbo Zheng, Zheng Zeng, Rong Zheng, and Qian Zhang. 2014. WiCop: Engineering WiFi temporal white-spaces for safe operations of wireless personal area networks in medical applications. *IEEE Transactions on Mobile Computing* 13, 5 (2014), 1145–1158. DOI: http://doi.org/10.1109/TMC.2013.31

[2] Zhongcheng Wei, Wei Chen, Shuli Ning, Weidong Lin, Nan Li, Bin Lian, Xiang Sun, and Jijun Zhao. 2025. A survey on WiFi-based human identification: Scenarios, challenges, and current solutions. *ACM Transactions on Sensor Networks* 21, 1 (2025), 1–32.

[3] Fangzhan Shi, Wenda Li, Chong Tang, Yuan Fang, Paul V. Brennan, and Kevin Chetty. 2025. ML-track: Passive human tracking using WiFi multi-link round-trip CSI and particle filter. *IEEE Transactions on Mobile Computing* (2025), 1–18. DOI: http://doi.org/10.1109/TMC.2025.3529897

[4] Ahmed Shokry, Moustafa Elhamshary, and Moustafa Youssef. 2021. DynamicSLAM: Leveraging human anchors for ubiquitous low-overhead indoor localization. *IEEE Transactions on Mobile Computing* 20, 8 (2021), 2563–2575. DOI: http://doi.org/10.1109/TMC.2020.2984320

[5] Chi Lin, Asfandeyar Ahmad, Rongsheng Qu, Yi Wang, Lei Wang, Guowei Wu, Qiang Lin, and Qiang Zhang. 2024. A handwriting recognition system with WiFi. *IEEE Transactions on Mobile Computing* 23, 4 (2024), 3391–3409. DOI: http://doi.org/10.1109/TMC.2023.3279608

[6] Ke Xu, Jiangtao Wang, Hongyuan Zhu, and Dingchang Zheng. 2025. Evaluating self-supervised learning for WiFi CSI-based human activity recognition. *ACM Transactions on Sensor Networks* 21, 2 (2025), 1–38.

[7] Zhenghua Chen, Le Zhang, Chaoyang Jiang, Zhiguang Cao, and Wei Cui. 2019. WiFi CSI based passive human activity recognition using attention based BLSTM. *IEEE Transactions on Mobile Computing* 18, 11 (2019), 2714–2724. DOI: http://doi.org/10.1109/TMC.2018.2878233

[8] Neil Cameron. 2023. ESP32 microcontroller. In *ESP32 Formats and Communication: Application of Communication Protocols with ESP32 Microcontroller*. Springer, 1–54.

[9] Guanzhong Wang, Dongheng Zhang, Tianyu Zhang, Shuai Yang, Qibin Sun, and Yan Chen. 2024. Learning domain-invariant model for WiFi-based indoor localization. *IEEE Transactions on Mobile Computing* 23, 12 (2024), 13898–13913. DOI: http://doi.org/10.1109/TMC.2024.3438454

[10] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2017), 2935–2947.

[11] Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zheng-Jun Zha. 2022. Self-Sustaining Representation Expansion for Non-Exemplar Class-Incremental Learning. (2022). arXiv:2203.06359. Retrieved from https://arxiv.org/abs/2203.06359

[12] Dipam Goswami, Albin Soutif-Cormerais, Yuyang Liu, Sandesh Kamath, Bart Twardowski, Joost van de Weijer, et al. 2024. Resurrecting old classes with new data for exemplar-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 28525–28534.

[13] Yu Gu, Lianghu Quan, and Fuji Ren. 2014. WiFi-assisted human activity recognition. In *2014 IEEE Asia Pacific Conference on Wireless and Mobile.* 60–65. DOI: http://doi.org/10.1109/APWiMob.2014.6920266

[14] Heba Abdelnasser, Moustafa Youssef, and Khaled A. Harras. 2015. WiGest: A ubiquitous WiFi-based gesture recognition system. In *2015 IEEE Conference on Computer Communications.* 1472–1480. DOI: http://doi.org/10.1109/INFOCOM.2015.7218525

[15] Xuanzhi Wang, Anlan Yu, Kai Niu, Weiyan Shi, Junzhe Wang, Zhiyun Yao, Rahul C. Shah, Hong Lu, and Daqing Zhang. 2024. Understanding the diffraction model in static multipath-rich environments for WiFi sensing system design. *IEEE Transactions on Mobile Computing* 23, 11 (2024), 10393–10410. http://doi.org/10.1109/TMC.2024.3377708

[16] Daqing Zhang, Hao Wang, and Dan Wu. 2017. Toward centimeter-scale human activity sensing with Wi-Fi signals. *Computer* 50, 1 (2017), 48–57. http://doi.org/10.1109/MC.2017.7

[17] Fei Wang, Stanislav Panev, Ziyi Dai, Jinsong Han, and Dong Huang. 2019. Can WiFi estimate person pose? arXiv:1904.00277. Retrieved from https://arxiv.org/abs/1904.00277

[18] Mahnaz Chahoushi, Mohammad Nabati, Reza Asvadi, and Seyed Ali Ghorashi. 2023. CSI-based human activity recognition using multi-input multi-output autoencoder and fine-tuning. *Sensors* 23, 7 (2023). DOI: http://doi.org/10.3390/s23073591

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2001–2010.

[20] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. 2019. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 374–382.

[21] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017), 3521–3526.

[22] Yang Yang, Da-Wei Zhou, De-Chuan Zhan, Hui Xiong, Yuan Jiang, and Jian Yang. 2021. Cost-effective incremental deep model: Matching model capacity with the least sampling. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 3575–3588.

[23] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. arXiv:1606.04671. Retrieved from https://arxiv.org/abs/1606.04671

[24] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. 2022. DyTox: Transformers for continual learning with DYnamic TOken eXpansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 9285–9295.

[25] Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. 2022. R-DFCIL: Relation-guided representation learning for data-free class incremental learning. In *European Conference on Computer Vision.* 423–439.

[26] Nader Asadi, MohammadReza Davari, Sudhir Mudur, Rahaf Aljundi, and Eugene Belilovsky. 2023. Prototype-sample relation distillation: Towards replay-free continual learning. In *International Conference on Machine Learning.* 1093–1106.

[27] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *Advances in Neural Information Processing Systems* 33 (2020), 18661–18673.

[28] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 857–876.

[29] Matthias De Lange and Tinne Tuytelaars. 2021. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 8250–8259.

[30] Anurag Roy, Vinay K. Verma, Sravan Voonna, Kripabandhu Ghosh, Saptarshi Ghosh, and Abir Das. 2023. Exemplar-free continual transformer with convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 5897–5907.

[31] Yong Zhang, Fei He, Yujie Wang, Dingchao Wu, and Guangwei Yu. 2023. CSI-based cross-scene human activity recognition with incremental learning. *Neural Computing and Applications* 35, 17 (2023), 12415–12432.

[32] Xue Ding, Yi Zhong, Sheng Wu, Chunxiao Jiang, and Weiliang Xie. 2023. Passive sensing for class-incremental human activity recognition. *IEEE Geoscience and Remote Sensing Letters* 20 (2023), 1–5.

[33] Qunhang Fu, Fei Wang, Mengdie Zhu, Han Ding, Jinsong Han, and Tony Xiao Han. 2024. CCS: Continuous Learning for Customized Incremental Wireless Sensing Services. (2024). arXiv:2412.04821. Retrieved from https://arxiv.org/abs/2412.04821

[34] Yuxuan Weng, Tianyue Zheng, Yanbing Yang, and Jun Luo. 2026. FM-Fi 2.0: Foundation model for cross-modal multi-person human activity recognition. *IEEE Transactions on Mobile Computing* 25, 1 (2026), 566–582.

[35] Jingzhi Hu, Tianyue Zheng, Zhe Chen, Hongbo Wang, and Jun Luo. 2023. MUSE-Fi: Contactless muti-person sensing exploiting near-field Wi-Fi channel variation. In *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*. 1–15.

[36] Xin Li, Hongbo Wang, Zhe Chen, Zhiping Jiang, and Jun Luo. 2024. Uwb-fi: Pushing wi-fi towards ultra-wideband for fine-granularity sensing. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. 42–55.

[37] Yinghui He, Mingming Xu, Zhe Chen, Fu Xiao, and Jun Luo. 2025. Beam-Fi: Integrated sensing and communication via MU-MIMO upon commodity Wi-Fi. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 9, 3 (2025), 1–22.

[38] Rong Li, Tao Deng, Siwei Feng, Mingjie Sun, and Juncheng Jia. 2025. ConSense: Continually sensing human activity with WiFi via growing and picking. In *Proceedings of the AAAI Conference on Artificial intelligence*, Vol. 39. 14292–14300.

[39] Bing Li, Wei Cui, Wei Wang, Le Zhang, Zhenghua Chen, and Min Wu. 2021. Two-stream convolution augmented transformer for human activity recognition. In *Proceedings of the AAAI Conference on Artificial intelligence*, Vol. 35. 286–293.

[40] Bruce X. B. Yu, Jianlong Chang, Lingbo Liu, Qi Tian, and Chang Wen Chen. 2022. Towards a unified view on visual parameter-efficient transfer learning. arXiv:2210.00788. Retrieved from https://arxiv.org/abs/2210.00788

[41] Agus Kurniawan. 2021. *Introduction to NVIDIA jetson nano*. Apress, Berkeley, CA, 1–6. DOI: http://doi.org/10.1007/978-1-4842-6452-2_1

[42] Linlin Guo, Lei Wang, Chuang Lin, Jialin Liu, Bingxian Lu, Jian Fang, Zhonghao Liu, Zeyang Shan, Jingwen Yang, and Silu Guo. 2019. Wiar: A public dataset for Wifi-based activity recognition. *IEEE Access* 7 (2019), 154935–154945.

[43] Jianfei Yang, He Huang, Yunjiao Zhou, Xinyan Chen, Yuecong Xu, Shenghai Yuan, Han Zou, Chris Xiaoxuan Lu, and Lihua Xie. 2024. MM-Fi: Multi-modal non-intrusive 4D human dataset for versatile wireless sensing. *Advances in Neural Information Processing Systems* 36 (2024), 18756–18768.

[44] Fei Wang, Yizhe Lv, Mengdie Zhu, Han Ding, and Jinsong Han. 2024. XRF55: A radio frequency dataset for human indoor action analysis. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 1 (2024), 1–34.

[45] Fei Zhu, Xu-Yao Zhang, Chuang Wang, Fei Yin, and Cheng-Lin Liu. 2021. Prototype augmentation and self-supervision for incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5871–5880.

[46] Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision*. 532–547.

[47] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems* 32 (2019), 1–12.

[48] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980

Received 18 October 2025; revised 27 January 2026; accepted 18 February 2026