IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

Low-Cost Wi-Fi CSI-Based Human Activity Recognition Using ESP32 and a Gated Recurrent Unit Classifier

Pavani M B[1], Priya M S[2], Vindhya G M[3], Jeevan H G[4]

Student, Dept. of ISE (Information Science and Engineering), AMC Engineering College,

AMC Campus Bannerghatta Rd, Kalkere, Bengaluru – 560083, Karnataka, India[1,2,3,4]

Abstract: Human Activity Recognition (HAR) has traditionally relied on cameras or wearable sensors, which raise privacy and user-compliance concerns. This paper presents a device-free HAR system based on Wi-Fi Channel State Information (CSI) captured with a single low-cost ESP32 microcontroller, addressing the accessibility gap left by prior work that depends on expensive, multi-antenna, research-grade Wi-Fi hardware. Raw CSI logs were collected from four subjects performing ten activities, parsed, and audited for recording-duration quality; 13 of 150 recordings were excluded as truncated. The remaining data were converted to CSI magnitude, segmented into overlapping one-second sliding windows, and normalized using training-set statistics. A two-layer Gated Recurrent Unit (GRU) classifier with 54,346 parameters was trained with class-weighted cross-entropy loss and early stopping. On a held-out, recording-level test split of 607 windows, the model achieved 64.74% accuracy and 65.38% macro F1-score, and macro F1 rose from 59.19% to 65.38% after the data-quality audit (measured on differently sized test splits). High-motion activities were recognized well (F1 > 0.75), whereas low-motion activities were less separable, which is consistent with the limited resolution of single-antenna CSI. A controlled live test that held the Wi-Fi hotspot constant and changed only the room showed that the learned CSI signatures are room-specific, confirming the cross-environment generalization limitation reported in the literature. The work provides an accessible, reproducible baseline for low-cost CSI-based HAR and an empirical characterization of its present limitations.

Keywords: Wi-Fi Sensing; Channel State Information (CSI); Human Activity Recognition (HAR); ESP32; Gated Recurrent Unit (GRU); Deep Learning; Device-Free Sensing

I. INTRODUCTION

Human Activity Recognition (HAR) is the task of automatically identifying the physical activity a person is performing from sensor data, with applications in elder care, smart homes, healthcare monitoring, and security. The two dominant approaches, camera-based vision systems and wearable inertial sensors, each carry significant drawbacks. Camera-based systems raise privacy concerns and fail under poor lighting or occlusion, while wearable systems depend on continuous user compliance, which is unreliable for elderly or non-compliant users.

Wi-Fi Channel State Information (CSI) has emerged as an alternative, device-free sensing medium. CSI describes the amplitude and phase of a wireless signal across multiple subcarriers as it propagates from transmitter to receiver; human movement within the signal path perturbs this propagation in ways characteristic of the activity performed. Unlike the Received Signal Strength Indicator (RSSI), CSI provides fine-grained, per-subcarrier information, enabling substantially richer activity discrimination [1], [2].

Most existing CSI-HAR research depends on specialized, multi-antenna Wi-Fi network interface cards such as the Intel 5300 [3], which are costly and not widely accessible. This paper investigates the feasibility of CSI-based HAR on the ESP32, a low-cost microcontroller with official firmware support (Espressif's esp-csi [4]) for CSI extraction, and presents a complete, reproducible pipeline from raw signal capture to a trained, deployable classifier, together with an empirically grounded account of its limitations.

The main contributions of this work are: (i) an end-to-end CSI-HAR pipeline built entirely on a single ESP32; (ii) a recording-duration data-quality audit, with a measurement of its effect on accuracy; (iii) a recording-level evaluation protocol that prevents leakage between overlapping windows; and (iv) a controlled cross-room experiment in which the physical environment is the only changed variable.

The remainder of this paper is organized as follows. Section II reviews related work. Section III describes the system architecture and methodology, including data acquisition, cleaning, feature extraction, and model design. Section IV

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151098

details the experimental setup. Section V reports results and discussion. Section VI discusses limitations, and Section VII concludes with directions for future work.

II. RELATED WORK

Halperin et al. [3] released one of the first open-source CSI extraction tools, for the Intel 5300 NIC, enabling much of the subsequent CSI-sensing research; however, the tool remains hardware-specific and does not support low-cost microcontrollers. Wang et al. [5] proposed CARM, a CSI-speed and CSI-activity correlation model that achieves high recognition accuracy on commodity Wi-Fi cards but relies on multi-antenna hardware. Yousefi et al. [1] surveyed RSSI- and CSI-based recognition systems and concluded that CSI provides significantly richer information for activity recognition than RSSI. Ma et al. [2] provided a broader survey of CSI-sensing applications, identified deep learning, particularly recurrent architectures, as an increasingly dominant approach, and noted that the environment dependence of CSI signatures remains a largely unsolved challenge, a finding independently corroborated by the present work.

Recurrent networks suit CSI because activities unfold as time series. The Long Short-Term Memory (LSTM) network [6] introduced gated memory cells to mitigate vanishing gradients. Cho et al. [7] proposed the Gated Recurrent Unit (GRU), a lighter alternative that uses two gates instead of three and no separate cell state, originally for machine translation, and Chung et al. [8] empirically compared gated recurrent units on sequence-modelling tasks. This work adapts the GRU to CSI-based time-series classification, where its smaller parameter count is attractive for eventual on-device deployment.

The consistent gap across this literature, summarized in Table I, is a dependence on expensive, multi-antenna, research-grade Wi-Fi hardware. It remains open whether reliable activity recognition is achievable on low-cost, single-antenna, commodity microcontrollers such as the ESP32; this is the gap the present paper addresses.

TABLE I SUMMARY OF RELATED WORK AND THE PRESENT STUDY

Ref.	Contribution	Hardware	Gap / limitation
[3]	Open-source CSI extraction tool	Intel 5300 NIC	Hardware-specific; no microcontroller support
[5]	CARM: CSI-speed and CSI-activity models	Commodity multi-antenna Wi-Fi cards	Depends on multi-antenna hardware
[1]	Survey of RSSI- and CSI-based behavior recognition	Not applicable (survey)	CSI richer than RSSI; no low-cost implementation
[2]	Survey of CSI sensing and deep-learning trends	Not applicable (survey)	Environment dependence left as an open problem
This work	End-to-end GRU-based HAR pipeline with data-quality audit	Single ESP32 with esp-csi firmware	Single room and four subjects; weak low-motion separability

III. SYSTEM DESIGN AND METHODOLOGY

The proposed system follows a four-layer architecture, shown in Fig. 1: a hardware layer responsible for raw CSI acquisition, a data layer that transforms raw signals into clean, normalized model inputs, a model layer implementing the GRU classifier, and an application layer exposing the trained model through offline, live, and web-based interfaces.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

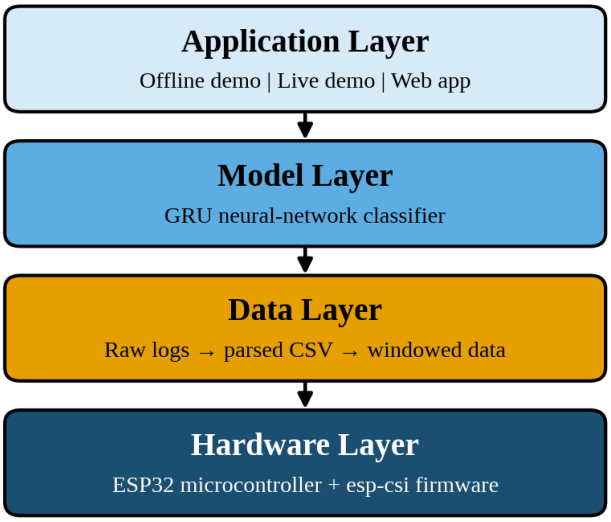


Fig. 1 System architecture: hardware, data, model, and application layers.

A. Hardware and Data Acquisition

Data were captured using a single ESP32-WROOM-32 module running the Espressif esp-csi firmware [4], connected to a 2.4 GHz Wi-Fi hotspot. Raw CSI logs were recorded via ESP-IDF [9] serial monitoring using the built-in logging toggle, which avoids known pyserial DTR/RTS reset issues encountered with direct serial-port scripting. Four subjects performed ten activities (standing/sitting, walking, waving, turning around, picking up an object, lying down, falling, bending, empty room, and door open/close) over repeated trials, yielding 150 raw recordings of approximately 15 to 20 seconds each, collected in a single indoor room. For dynamic activities, subjects repeated the motion continuously throughout the trial rather than performing it once, maximizing the amount of genuinely labeled motion captured per recording. The acquisition setup is summarized in Table II.

TABLE II DATA ACQUISITION SUMMARY

Parameter	Value
Sensing device	ESP32-WROOM-32, single antenna, esp-csi firmware
Network	2.4 GHz Wi-Fi hotspot (constant during all recordings)
Subjects / activities	4 subjects / 10 activity classes
Raw recordings	150 (13 excluded by audit, 137 retained)
Recording length	Approximately 15 to 20 s per recording
Packet rate	Approximately 100 packets/s
Features per packet	62 CSI magnitudes (62 I/Q pairs)
Environment	Single indoor room

B. Data Quality Auditing

Each recording's duration was audited against the physically plausible duration for its activity. Recordings substantially shorter than the time required to complete the labelled motion (for example, under 4 seconds for a full turning motion) were identified as likely truncated and excluded. This process excluded 13 of 150 recordings (9 turning-around, 3 standing-sitting, and 1 lying-down), retaining 137 usable recordings. Section V-C quantifies the measurable benefit of this step. Fig. 2 summarizes the full processing pipeline from raw capture through classification, including the auditing step.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

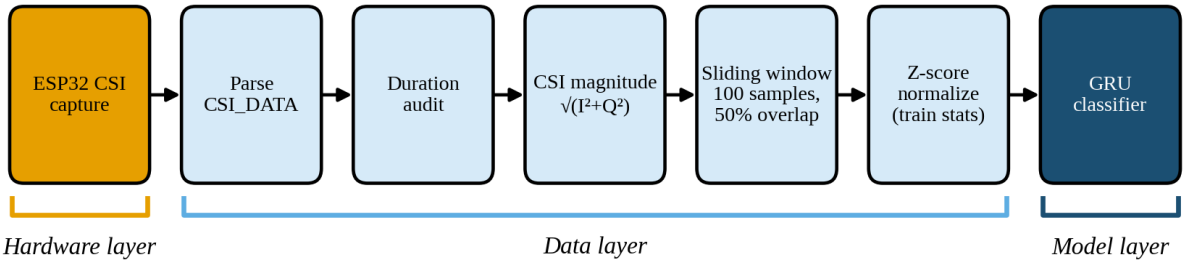


Fig. 2 End-to-end processing pipeline, from raw ESP32 capture to GRU classification.

C. CSI Magnitude Extraction

Each raw CSI packet contains 128 values stored as interleaved in-phase (I) and quadrature (Q) components. The first four values are discarded as hardware-flagged invalid, leaving 124 values, or 62 I/Q pairs. The CSI magnitude of subcarrier k is computed as:

$$|H_k| = \sqrt{I_k{}^2 + Q_k{}^2}, \quad k = 1, \ldots, 62 \quad (1)$$

producing 62 real-valued magnitude features per packet. Phase information is not used in this study.

D. Sliding Window Segmentation

At an empirical sampling rate of approximately 100 packets/s, each continuous recording is segmented into overlapping windows of 100 samples (about 1 second), advanced by a stride of 50 samples (50% overlap):

$$w_i = X[t_i : t_i + 100], \quad t_{i+1} = t_i + 50 \quad (2)$$

where X is the T × 62 magnitude matrix of a recording and every window inherits that recording's activity label. To prevent any single long recording from dominating its class, each trial is additionally capped at 22 seconds before windowing, which bounds a recording to at most 43 windows.

E. Normalization

Each of the 62 features is standardized using z-score normalization:

$$X_{\text{norm}} = \frac{X - \mu_{\text{train}}}{\sigma_{\text{train}}} \quad (3)$$

where μ_{train} and σ_{train} are computed exclusively from the training split and applied unchanged to the validation and test data, preventing data leakage.

F. GRU Classifier Architecture

A two-layer GRU processes each window of shape (100, 62). At each timestep t, the GRU computes an update gate, a reset gate, a candidate hidden state, and a final hidden state, as illustrated in Fig. 3:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (4)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (5)$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \quad (6)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (7)$$

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

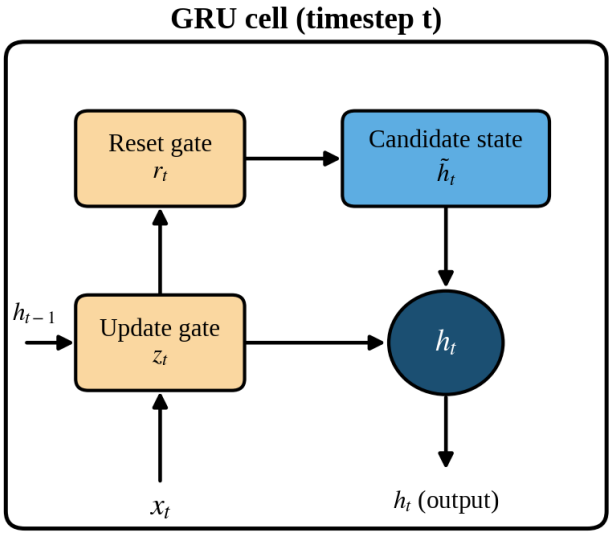


Fig. 3 Internal structure of a single GRU cell at timestep t.

Eq. (7) uses one of two equivalent conventions; the original formulation [7] and the PyTorch implementation [12] swap the roles of z_t and $1 - z_t$. The final hidden state h_{100} (hidden size 64) is passed through a fully connected classifier, Linear (64→64) with ReLU and dropout [10], followed by Linear(64→10), with a SoftMax output over the ten activity classes. The full network has 54,346 trainable parameters, broken down in Table III.

TABLE III MODEL PARAMETER BREAKDOWN

Layer	Output / shape	Parameters
GRU layer 1 (input 62, hidden 64)	(100, 64)	24,576
GRU layer 2 (input 64, hidden 64)	(100, 64)	24,960
Linear 64→64 + ReLU + dropout (0.3)	(64)	4,160
Linear 64→10 + SoftMax	(10)	650
Total		**54,346**

G. Training Procedure

The network is trained with a class-weighted cross-entropy loss to compensate for class imbalance across the ten activities. The weight of class c is inversely proportional to its frequency in the training split:

$$w_c = \frac{N}{K \cdot n_c} \qquad (8)$$

where N = 2,239 is the number of training windows, K = 10 is the number of classes, and n_c is the number of training windows of class c. The loss over a mini-batch of B windows, with true class c_i and predicted probability $\hat{y}_{i,c}$, is (up to a batch-normalization constant):

$$L = -\sum_{i=1}^{B} w_{c_i} \log \hat{y}_{i,c_i} \qquad (9)$$

Table V lists the resulting weights. Rarer classes such as turning_around and picking_object receive proportionally higher weight, which discourages the model from favoring frequent classes such as empty_room. The network is implemented in PyTorch [12] and trained with the Adam optimizer [11] under the configuration in Table IV; the learning rate is halved when validation macro F1 plateaus, and training stops early when validation macro F1 does not improve for 8 epochs. Training converged after 37 epochs.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

TABLE IV TRAINING CONFIGURATION

Setting	Value
Optimizer	Adam, initial learning rate 0.001
Learning-rate schedule	Halved on validation-F1 plateau
Batch size	32
Dropout	0.3
Loss	Class-weighted cross-entropy
Early stopping	Patience 8 epochs (validation macro F1)
Epochs to convergence	37
Input window	100 × 62, stride 50

H. Live Inference

For the live demonstration, the trained model is wrapped in an inference loop that keeps a rolling buffer (a double-ended queue) of the most recent CSI packets, forms 100-sample windows with the same preprocessing and training-set normalization statistics and smooths the per-window predictions by majority voting to suppress transient misclassifications. The same model is also exposed through offline and web-based interfaces (Fig. 1).

IV. EXPERIMENTAL SETUP

A. Dataset

After cleaning and windowing, the dataset comprises 3,239 labelled windows across 10 activity classes, an average of about 24 windows per retained recording. Fig. 4 shows the class composition; the empty_room class is the largest (506 windows) and turning_around the smallest (197 windows).

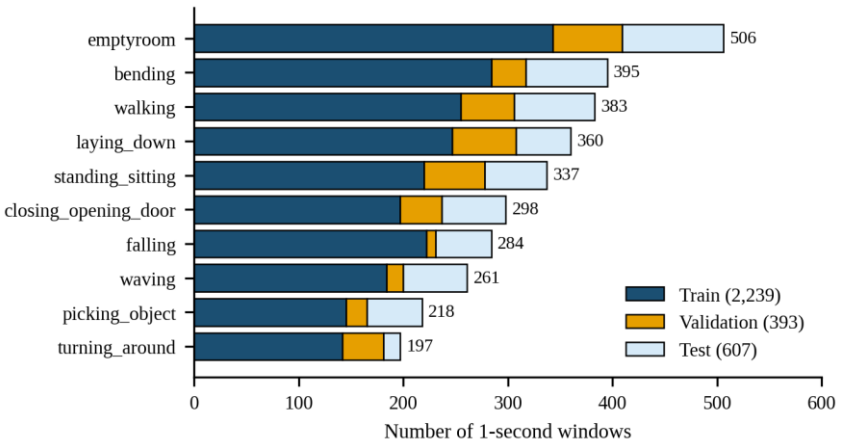


Fig. 4 Number of windows per activity, split into training, validation, and test sets.

B. Train / Validation / Test Split

To prevent data leakage from overlapping windows, the dataset is split at the recording level rather than the window level: all windows from a given recording are assigned entirely to one split. This yields 2,239 training, 393 validation, and 607 test windows, with no source recording appearing in more than one split. Table V reports the per-activity window counts together with the class weights from Eq. (8).

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

TABLE V WINDOW COUNTS PER ACTIVITY AND CLASS WEIGHTS

Activity	Train	Val	Test	Total	Weight
bending	284	33	78	395	0.7884
closing_opening_door	197	40	61	298	1.1365
emptyroom	343	66	97	506	0.6528
falling	222	9	53	284	1.0086
laying_down	247	61	52	360	0.9065
picking_object	145	20	53	218	1.5441
standing_sitting	220	58	59	337	1.0177
turning_around	142	39	16	197	1.5768
walking	255	51	77	383	0.8780
waving	184	16	61	261	1.2168
Total	**2,239**	**393**	**607**	**3,239**	

C. Evaluation Metrics

Performance is reported as overall accuracy and as macro-averaged precision, recall, and F1-score. For class c, precision $P_c = TP_c / (TP_c + FP_c)$ and recall $R_c = TP_c / (TP_c + FN_c)$; the F1-score and its macro average over the K classes are:

$$F1_c = \frac{2P_cR_c}{P_c+R_c} \qquad (10)$$

$$\text{Macro F1} = \frac{1}{K}\sum_{c=1}^{K} F1_c \qquad (11)$$

Because the classes are imbalanced, macro F1 weights every class equally and is therefore used for learning-rate scheduling, early stopping, and model selection on the validation set.

V. RESULTS AND DISCUSSION

A. Overall Performance

On the 607-window held-out test set, the model correctly classified 393 windows, giving the metrics reported in Table VI. This is well above the 10% expected from random guessing over ten classes and the 16.0% obtained by always predicting the majority class (empty_room, 97 of 607 test windows). Macro recall exceeds macro precision, which is consistent with class-weighted training shifting predictions toward the rarer classes.

TABLE VI OVERALL TEST-SET PERFORMANCE

Metric	Value
Accuracy	64.74%
Macro precision	65.40%
Macro recall	68.25%
Macro F1-score	65.38%

B. Per-Activity Performance

Table VII and Fig. 5 report per-class F1-scores. High-motion, kinematically distinctive activities (picking up an object, turning around, bending, waving, falling) achieved F1-scores above 0.75, whereas low-motion activities (standing/sitting transitions and door open/close) scored below 0.45. Walking, although a dynamic activity, scored only 0.51, so its confusion pattern merits closer inspection.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

TABLE VII PER-ACTIVITY F1-SCORE ON THE TEST SET

Activity	Test windows	F1
picking_object	53	0.92
turning_around	16	0.89
bending	78	0.85
waving	61	0.80
falling	53	0.77
emptyroom	97	0.63
walking	77	0.51
laying_down	52	0.50
standing_sitting	59	0.44
closing_opening_door	61	0.20

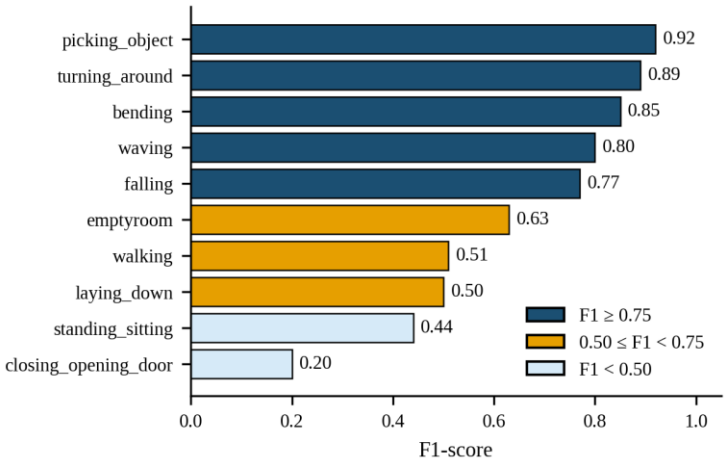


Fig. 5 Per-activity F1-score on the held-out test set, sorted in descending order.

The per-class scores show the weakest results concentrated among the low-motion classes (standing_sitting, closing_opening_door, laying_down, empty_room), consistent with the hypothesis that single-antenna CSI amplitude alone provides insufficient resolution to separate subtly different low-motion signatures. We interpret this as a sensor-resolution limitation rather than a pipeline defect, since the high-motion classes are consistently well separated, although this explanation was not isolated experimentally. Note also that some classes have small test support (for example, 16 windows for turning_around), so their F1-scores carry wider uncertainty than the table suggests.

C. Effect of Data Cleaning

Before the duration-based data-quality audit described in Section III-B, the identical pipeline achieved 60.70% accuracy and 59.19% macro F1 on its own (differently sized) test split. After excluding the 13 truncated recordings, macro F1 improved to 65.38%, most notably for turning_around (F1 0.64 to 0.89) and picking_object (F1 0.68 to 0.92), as shown in Table VIII and Fig. 6. Turning_around accounts for 9 of the 13 excluded recordings. Picking_object lost no recordings, so its gain is likely indirect (for example, fewer truncated turning trials to be confused with); a before-and-after confusion analysis would be needed to confirm this.

TABLE VIII EFFECT OF THE DATA-QUALITY AUDIT

Metric	Before	After	Change
Accuracy	60.70%	64.74%	+4.04 pts
Macro F1-score	59.19%	65.38%	+6.19 pts
F1, turning_around	0.64	0.89	+0.25
F1, picking_object	0.68	0.92	+0.24

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

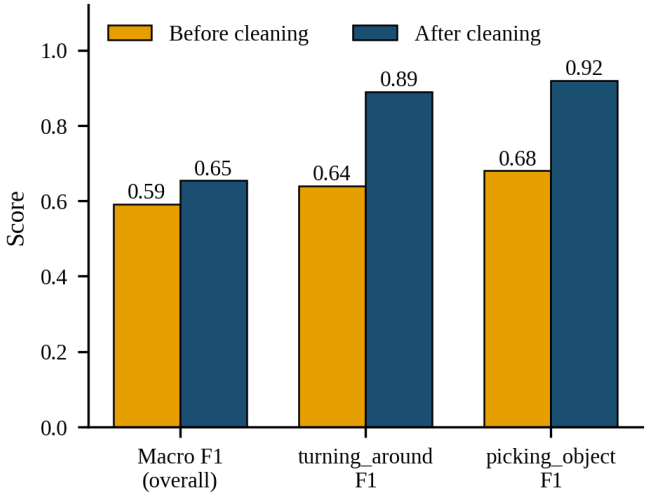


Fig. 6 Effect of the data-quality audit on macro F1 and on the two most-affected classes.

Because the two evaluations used differently sized test splits, this comparison is indicative rather than a controlled ablation. The consistent gains nevertheless suggest that systematic data-quality auditing is a primary determinant of model performance in CSI-HAR pipelines, not a peripheral step.

VI. LIMITATIONS

A. Cross-Environment Generalization

Testing in a different room caused misclassifications despite unchanged Wi-Fi hardware, indicating environment dependent features [2]. Domain-independent features [13] may improve generalization.

B. Single-Subject Assumption

The model assumes one subject per recording and showed unstable predictions when multiple subjects were present.S

VII. CONCLUSION AND FUTURE WORK

This paper presented a complete, reproducible Wi-Fi CSI-based HAR pipeline built entirely on a single low-cost ESP32 microcontroller, achieving 64.74% accuracy and 65.38% macro F1 across ten activity classes on held-out, recording-level test data. A systematic data-quality audit was associated with a substantial improvement in model performance, and a controlled live-deployment experiment provided first-hand evidence of the cross-environment generalization limitation widely reported in the literature. Future work will focus on: (i) multi-environment data collection to improve room invariance; (ii) expanding subject diversity to enable statistically meaningful subject-independent evaluation; (iii) incorporating CSI phase information alongside amplitude; and (iv) empirical benchmarking against LSTM, 1D-CNN, and classical machine-learning baselines on this dataset.

REFERENCES

[1]. 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, 2017.

[2]. Y. Ma, G. Zhou, and S. Wang, “WiFi sensing with channel state information: A survey,” *ACM Comput. Surv.*, vol. 52, no. 3, 2019.

[3]. D. Halperin, W. Hu, A. Sheth, and D. Wetherall, “Tool release: Gathering 802.11n traces with channel state information,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, p. 53, 2011.

[4]. Espressif Systems, “esp-csi: Channel State Information tool for ESP32,” GitHub repository. [Online]. Available: https://github.com/espressif/esp-csi

[5]. W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu, “Understanding and modeling of WiFi signal based human activity recognition,” in *Proc. ACM MobiCom*, 2015.

[6]. S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[7]. K. Cho, B. van Merriënboer, C. Gulcehre, et al., “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in *Proc. EMNLP*, 2014.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 10, October 2026

DOI: 10.17148/IJARCCE.2026.151098

[8]. J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv:1412.3555, 2014.

[9]. Espressif Systems, "ESP-IDF programming guide." [Online]. Available: https://docs.espressif.com/projects/esp-idf/

[10]. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[11]. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[12]. A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 32, 2019.

[13]. Y. Zheng et al., "Zero-effort cross-domain gesture recognition with Wi-Fi," in *Proc. ACM MobiSys*, 2019, pp. 313–325.

 This work is licensed under a Creative Commons Attribution 4.0 International License