Edge-AI Recognition of Anxiety and Stress-Displaying Behaviors: A Multi-Scale Temporal Convolutional Network with Self-Attention Deployed on Resource-Constrained Hardware

Ugonna Oleh[1] and Roman Obermaisser[1]

Chair of Embedded Systems, University of Siegen, 57076 Siegen, Germany
{ugonna.oleh, roman.obermaisser}@uni-siegen.de

Abstract. Anxiety disorders and chronic stress represent critical global mental health challenges that frequently manifest as self-regulatory physical markers, specifically Self-Regulatory Coping Behaviors (SRCBs) and Body-Focused Repetitive Behaviors (BFRBs). Continuous, objective tracking of these behaviors using wearable sensors is vital for mental health monitoring, yet edge deployment is restricted by microcontroller resource constraints. In this study, we combine the *Adamsense* and *StresSense* wrist sensor datasets into a unified dataset comprising 1,205,028 samples across 14 anxiety- and stress-displaying classes. We adapt a Multi-Scale Temporal Convolutional Network with temporal self-attention (TCN-Attention-HAR) to extract multi-frequency motion components and weigh significant behavioral epochs. Under 5-fold cross-validation on the combined dataset, the adapted model achieves a mean validation accuracy of 87.90% (±1.17%) and macro F1-score of 88.19% (±1.74%) under subject-dependent cross-validation, outperforming the baseline DilatedTCN (84.13% accuracy, 83.91% F1-score). Under subject-independent Group 5-Fold validation, performance drops to 65.03% accuracy due to user-specific motion shifts. To enable microcontroller execution, we transpose the PyTorch-trained weights to a Keras equivalent and perform 8-bit integer post-training quantization, compressing the model size to ∼481.38 KB. The model is deployed on a Seeed Studio XIAO Sense nRF52840 microcontroller using TensorFlow Lite Micro. Local real-time inference on 128-timestep windows (2.56 seconds of IMU data) executes with a latency of 14 ms, demonstrating the feasibility of privacy-preserving, zero-latency, and high-accuracy behavioral monitoring at the edge.

Keywords: Human Activity Recognition (HAR) · Edge-AI · Temporal Convolutional Networks · Self-Attention · Microcontrollers · Mental Health Monitoring

1 Introduction

Anxiety disorders and chronic stress affect nearly 30% of the adult population globally at some point in their lifetime [1]. These psychological states frequently

manifest as self-regulatory physical markers, classified as Self-Regulatory Coping Behaviors (SRCBs), such as smoking and eating, or Body-Focused Repetitive Behaviors (BFRBs), such as nail-biting, hair-pulling, skin-scratching, and face-touching [2]. Continuous and objective tracking of these physical cues provides valuable insights for behavioral therapy, early psychological interventions, and stress management.

Traditional tracking approaches depend on clinical observation or subjective self-reporting, which are prone to recall bias and lack real-time granularity. Wearable devices equipped with Inertial Measurement Units (IMUs) enable unobtrusive, continuous monitoring of human activities in free-living environments. However, transmitting high-frequency raw sensor data to cloud servers raises critical concerns regarding data privacy, network latency, and energy consumption on mobile devices. To address these limitations, Edge-AI and TinyML paradigms enable machine learning models to execute directly on low-power microcontrollers (MCUs) at the sensor node, which is projected to grow to over 30 billion IoT devices globally by 2030 [12]. Yet, deploying deep learning models on MCUs is highly constrained by limited Static Random-Access Memory (SRAM) and Flash memory (often < 256 KB and 1 MB respectively).

Sequential modeling in Human Activity Recognition (HAR) has transitioned from traditional recurrent neural architectures (LSTMs and GRUs) to Temporal Convolutional Networks (TCNs), which utilize causal and dilated convolutions to capture long-range dependencies while remaining highly parallelizable during training [4]. However, standard TCNs use a single fixed convolutional kernel size, which limits their ability to capture human motion signals occurring at different frequencies and speeds. Furthermore, standard TCNs treat all timesteps in a sliding window with equal importance, ignoring the localized nature of anxiety-induced behaviors. To address these limitations, we adapt the TCN-Attention-HAR architecture [11], which combines: (1) **Multi-Scale Dilated Convolutions** using three parallel TCN trunks with kernel sizes of 3, 5, and 7 to capture fine-grained to coarse temporal features; and (2) **Temporal Self-Attention** to dynamically weight and aggregate the most salient temporal segments in the sliding window.

While the TCN-Attention-HAR model was originally proposed by Xiong and Wang [11] for general activity recognition, its applicability to fine-grained anxiety markers and its end-to-end TinyML compilation workflow for microcontroller deployment represent unique challenges. The main contributions of this work are as follows:

- **Unified 14-Class Dataset Integration**: We combine raw wrist-worn accelerometer and gyroscope measurements from the public *Adamsense* and *StresSense* datasets, applying text normalization to merge case-variant labels (e.g., merging 'Smoking' and 'smoking'), resulting in a unified 14-class dataset containing 1,205,028 samples representing anxiety- and stress-displaying physical behaviors.
- **TinyML Optimization Pipeline**: We implement a weight transposition and graph reconstruction workflow to transfer trained sequential models from

PyTorch to Keras, followed by 8-bit post-training quantization to compress the model to fit on microcontrollers.
– **Physical Deployment and Verification**: We deploy the quantized TCN-Attention-HAR model onto a Seeed Studio XIAO Sense nRF52840 (ARM Cortex-M4) microcontroller. Real-time local inference is achieved in 14 ms, validating on-device, low-latency, and privacy-preserving detection of physical anxiety and stress displays.

2 Related Work

Wearable sensor-based HAR has transitioned from classical machine learning models relying on hand-crafted features to deep learning models that automatically extract hierarchical representations from raw signals [6].

2.1 Anxiety and Stress Behavior Recognition

The identification of anxiety and stress markers using body-worn sensors was pioneered by the datasets analyzed in this work. The *ADAM-sense* dataset [1] recorded 11 anxiety-displaying behaviors (such as ear rubbing, forehead rubbing, hair pulling, hand tapping, knuckles cracking, and nape rubbing) across different body postures, achieving over 92% accuracy with a combined CNN-LSTM architecture. The *StresSense* dataset [2] investigated self-regulatory behaviors representing stress and boredom (eating, smoking, nail-biting, and face touching) using wrist-worn sensors, achieving 93.50% test accuracy with a Deep Neural Network (DNN).

A major challenge in translating these achievements to practical use is the "lab-to-life" gap, where models trained on research-grade sensors degrade when applied to commercial hardware. Oleh and Obermaisser [5] quantified this domain shift for anxiety-related HAR, demonstrating that a 1DCNNBiLSTM model trained on the research-grade ADAMSense dataset experienced a performance drop from 89.11% to 49.0% accuracy when deployed directly on a commercial Garmin Venu 3 smartwatch, which was recovered to 87.0% accuracy using a targeted fine-tuning domain adaptation strategy.

2.2 Microcontroller Deployments and TinyML for HAR

Executing HAR models directly on low-resource microcontrollers represents a major trend in edge assisted living. Early implementations utilized classical algorithms; for instance, Stolovas et al. [9] developed a prototype on a Texas Instruments MSP430 MCU (utilizing only 16 KB Flash and 512 bytes RAM) that performed real-time classification of walking, running, and sitting using Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). Thottempudi et al. [10] evaluated real-time classification of dynamic activities on the STMicroelectronics B-L475E-IOT01A Discovery Kit, achieving 90% overall classification accuracy but highlighting challenges in distinguishing static postures.

To run complex deep learning models on MCUs, network quantization is crucial. Zhou et al. [12] evaluated 1D/2D CNNs and DeepConv LSTM models on edge hardware, deploying a fully quantized DeepConv LSTM model (compressed from 513 KB to 136 KB) on the Arduino Nano 33 BLE Sense Rev2 with a 21 ms inference latency. Pfitzinger and Wöhrle [7] utilized neural architecture search to compress and deploy a CNN model for real-time HAR using ambient audio on an ESP32-S3 microcontroller. More recently, Li et al. [3] proposed a resource-aware hierarchical network (`HPPI-Net`) that combined multi-spectral feature fusion (FFT, Wavelet, and Gabor) with parallel LSTMs, achieving 96.70% accuracy on an ARM Cortex-M4 MCU using only 22.3 KiB RAM and 439.5 KiB ROM. Our work builds upon these TinyML paradigms by deploying a multi-scale attention network on Cortex-M4 hardware.

2.3 Advanced HAR Architectures

Deep learning models have evolved to extract multi-frequency motion signals and handle feature dependencies. Hybrid architectures, such as CNNs combined with LSTMs or Gated Recurrent Units (GRUs), are widely used to extract spatial and temporal features [6]. Temporal attention mechanisms, as explored by Wei and Wang [11] in their *TCN-attention-HAR* architecture, assign varying weights to convolutional features, showing improvements on benchmark datasets (WISDM, PAMAP2, USC-HAD) and utilizing knowledge distillation to compress model parameters.

3 Methodology

3.1 Data Integration and Preprocessing

To build a comprehensive anxiety and stress detection model, we integrated the raw data from *Adamsense* and *StresSense*. To ensure compatibility, we restricted both datasets to the 6 channels representing wrist-worn motion tracking: 3-axis accelerometer data (A_x, A_y, A_z) and 3-axis gyroscope data (G_x, G_y, G_z). The final merged dataset contains 1,205,028 rows.

The original combination of the two datasets resulted in 16 classes due to case-variant string categories (such as `Smoking` from StresSense and `smoking` from Adamsense). To resolve these redundancies and establish a cleaner classification task, we applied a lowercase text normalization step. This merged `Nail_Biting` and `nail_biting` into `nail_biting`, `Smoking` and `smoking` into `smoking`, and `Staying_Still` and `staying_still` into `staying_still`, resulting in 14 unique classes. The sample distribution of the final dataset is tabulated in Table 1 and plotted by source in Fig. 1.

Because these datasets were recorded under separate controlled studies using different sensing hardware (Adamsense utilized multiple wrist-worn IMUs, while StresSense used a custom watch node), there are potential label and

Table 1. Unified 14-Class Dataset Distribution.

Unified Class Name	Row Count	Original Source Dataset(s)
nail_biting	174,646	Merged: Adamsense + StresSense
smoking	161,523	Merged: Adamsense + StresSense
staying_still	120,063	StresSense
eating	85,188	StresSense
face_touch	84,420	StresSense
nape_rubbing	67,859	Adamsense
knuckles_cracking	67,498	Adamsense
ear_rubbing	66,599	Adamsense
hair_pulling	65,973	Adamsense
forehead_rubbing	64,604	Adamsense
hand_scratching	64,490	Adamsense
hand_tapping	63,220	Adamsense
sitting	59,996	Adamsense
standing	58,949	Adamsense
Total Rows	**1,205,028**	

kinematic differences in how activities were performed and annotated. For example, hand-tapping in Adamsense reflects anxiety coping displays under sitting/standing postures, whereas eating or smoking in StresSense reflects routine stress-displaying behaviors. Lowercasing and string-matching were used to establish label consistency across overlapping classes. To prevent device-specific and subject-specific overlaps from biasing the evaluation, we also implement subject-independent cross-validation.

We apply a sliding window of $W = 128$ timesteps with a step size of $S = 64$ (50% overlap). At a sampling rate of 50 Hz, each window represents a 2.56-second temporal clip of human movement. For each window, we normalize the raw data using Z-score standardization:

$$\hat{x}_{t,j} = \frac{x_{t,j} - \mu_j}{\sigma_j} \tag{1}$$

where μ_j and σ_j are the mean and standard deviation computed for channel j across the training partition of the combined dataset. These scaling parameters are exported to the C++ runtime to ensure identical on-device scaling.

3.2 Model Architecture

Baseline DilatedTCN: A regular 1D Convolutional Neural Network (CNN) represents a standard baseline for HAR. However, standard 1D CNNs process temporal sequences without causal masking, which risks future information leakage in real-time streaming, and they lack dilation, which restricts their receptive field unless network depth (and thus memory usage) is significantly increased.

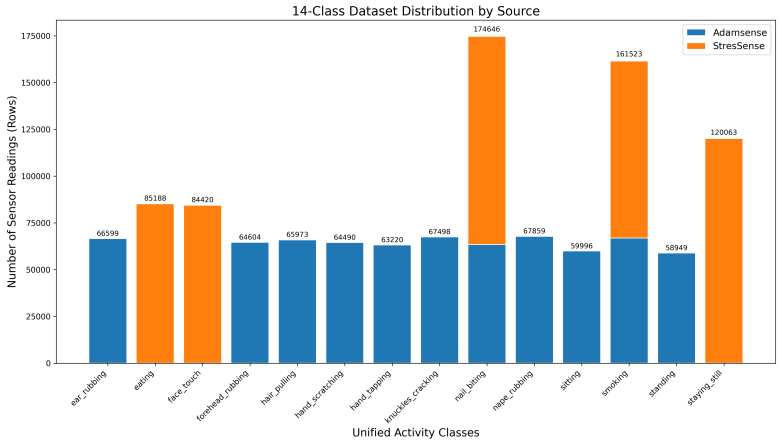


Fig. 1. The 14-class dataset row count distribution stacked by original source dataset (Adamsense in blue, StresSense in orange).

To establish a competitive yet low-resource baseline, we employ a Dilated Temporal Convolutional Network (DilatedTCN), which uses causal and dilated convolutions to capture long-range temporal dependencies while maintaining high parallelism during training.

The baseline network consists of a stack of 4 residual temporal blocks. Each block contains two dilated causal convolutional layers, followed by normalization, ReLU activation, and dropout ($p = 0.07$). The channels are set to $[64, 64, 64, 64]$, and the kernel size is $k = 8$. The dilation rate increases exponentially as $d = 2^i$ for layer $i \in \{0, 1, 2, 3\}$. Causal padding of $(k - 1) \times d$ is applied to prevent information leakage from future timesteps. The receptive field of the network is calculated as:

$$R = 1 + \sum_{i=0}^{L-1} 2 \cdot (k - 1) \cdot 2^i = 1 + 2 \cdot (8 - 1) \cdot (2^4 - 1) = 211 \tag{2}$$

Since $R = 211 > W = 128$, the baseline network covers the entire sequence window. The model outputs a representation of size $(B, 64, W)$, and the final timestep output at $t = W$ is fed into a linear layer to compute the logits of the classes.

Multi-Scale TCN with Self-Attention (TCN-Attention-HAR): The adopted architecture [11] employs three parallel TCN trunks to extract temporal features at multiple scales, using kernel sizes of $k \in \{3, 5, 7\}$. Each trunk contains a stack of 3 temporal blocks with channels $[64, 64, 64]$ and dilation rates of $d \in \{1, 2, 4\}$. Causal padding and layer normalization are applied inside the blocks.

The outputs of the three parallel trunks are concatenated along the channel dimension. For batch size B, window size W, and trunk channel dimension

$C = 64$, the concatenated representation has shape $(B, 3C, W) = (B, 192, W)$. We transpose this tensor to $(B, W, 192)$ to serve as input to the temporal self-attention block.

Temporal Self-Attention: Rather than using only the final timestep output, we utilize an attention mechanism to learn the varying importance of different timesteps. Let $h_t \in \mathbb{R}^{192}$ represent the concatenated feature vector at timestep $t \in \{1, \ldots, W\}$. The attention score is computed as:

$$u_t = \tanh(W_{att} h_t + b_{att}) \tag{3}$$

$$a_t = u_t^T c_{att} \tag{4}$$

$$\alpha_t = \frac{\exp(a_t)}{\sum_{\tau=1}^{W} \exp(a_\tau)} \tag{5}$$

where $W_{att} \in \mathbb{R}^{192 \times 192}$ and $b_{att} \in \mathbb{R}^{192}$ are weight parameters, and $c_{att} \in \mathbb{R}^{192}$ is the attention context vector. The softmax activation computes a probability distribution α_t over all timesteps. The final window representation $s \in \mathbb{R}^{192}$ is computed as the weighted sum:

$$s = \sum_{t=1}^{W} \alpha_t h_t \tag{6}$$

The aggregated feature vector s is passed to a fully connected linear layer to yield the final class logits.

4 TinyML Optimization and Hardware Deployment

4.1 Weight Transposition and INT8 Quantization

Microcontrollers lack hardware support for high-precision floating-point arithmetic (float32) and have tight memory footprints. Therefore, we compress the trained float32 model using Post-Training Quantization (PTQ) to 8-bit integers (INT8) using TensorFlow Lite.

Since TFLite lacks direct compatibility with PyTorch graph structures, we implement a weight transfer workflow:

1. **Graph Recreation**: We reconstruct the `TCN-Attention-HAR` and `DilatedTCN` network graphs in Keras.
2. **Weight Transposition**: We extract weights from the PyTorch state dictionary and transpose them. Because PyTorch Conv1D layers store weights as `(out_channels, in_channels, kernel_width)` while Keras Conv1D layers store them as `(kernel_width, in_channels, out_channels)`, we apply:
$$W_{Keras} = \text{transpose}(W_{PyTorch}, (2, 1, 0)) \tag{7}$$
Similarly, the fully connected weights are transposed from `(out_features, in_features)` to `(in_features, out_features)`.

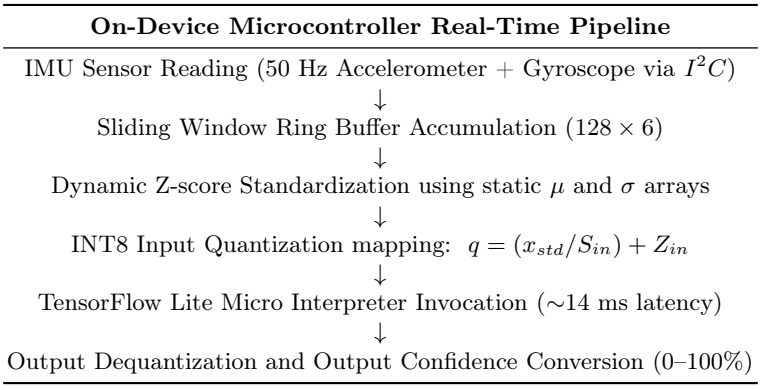


Fig. 2. The real-time on-device processing pipeline running on the Seeed Studio MCU.

3. **Layer Assignment**: We write the transposed weights into Keras layers.

After transferring the weights, we perform 8-bit integer quantization using the TensorFlow Lite converter. We use a representative dataset of 100 normalized sensor segments to calibrate activation ranges. The input and output tensors are forced to INT8. The quantization maps a real float value x to an integer q via:

$$q = \text{clamp}\left(\text{round}\left(\frac{x}{S}\right) + Z, -128, 127\right) \tag{8}$$

where S is the scale factor and Z is the zero-point integer. Quantization compresses the model footprint to ∼481.38 KB, enabling it to fit comfortably within the microcontrollers' storage limit. The TFLite file is converted into a C byte array in a header file (`model.h`).

4.2 Microcontroller Deployment Runtime

The quantized model is deployed on the *Seeed Studio XIAO Sense nRF52840* microcontroller. The MCU features an ARM Cortex-M4 CPU running at 64 MHz, 256 KB of SRAM, and 1 MB of Flash memory. The firmware is developed in the Arduino IDE and leverages the LSM6DS3TR-C onboard 6-axis IMU.

The C++ runtime executes the pipeline shown in Fig. 2:

1. **Sensor Ingestion**: The LSM6DS3TR-C IMU is sampled at 50 Hz via I^2C.
2. **Buffering**: A ring buffer stores the most recent 128 timesteps of 3-axis acceleration (in g) and 3-axis angular velocity (in deg/s).
3. **Standardization**: The C++ code performs Z-score scaling on the accumulated floats using the stored mean and standard deviation arrays (which were computed across the training partition of the combined dataset in Section 3.1 and exported to the microcontroller header file):

$$\mu = [-0.325330, -0.870343, 1.036772, -0.536674, 1.014300, 0.032300]$$
$$\sigma = [4.883689, 2.480127, 2.826128, 37.192112, 18.196502, 16.098750]$$

Table 2. Cross-Validation Results on the Combined 14-Class Dataset.

Validation Mode	Model Architecture	Mean Accuracy	Macro F1-score	Weighted F1-score
Subject-Dependent (Stratified 5-Fold)	DilatedTCN (Baseline)	84.13% ± 0.84%	83.91% ± 0.52%	83.45% ± 0.56%
	TCN-Attention-HAR	**87.90% ± 1.17%**	**88.19% ± 1.74%**	**87.31% ± 1.56%**
Subject-Independent (Group 5-Fold)	DilatedTCN (Baseline)	65.78% ± 5.40%	59.69% ± 8.82%	61.94% ± 7.29%
	TCN-Attention-HAR	**65.03% ± 6.76%**	**60.43% ± 8.15%**	**62.38% ± 7.32%**

4. **In-place Quantization**: Standardized floats are scaled using the TFLite input tensor parameters S_{in} and Z_{in} (Eq. 8) and cast to `int8_t`.
5. **Inference**: The TensorFlow Lite Micro interpreter executes the model.
6. **Dequantization**: The output probabilities are reconstructed from the resulting INT8 logits:

$$P_i = (q_{out,i} - Z_{out}) \times S_{out} \tag{9}$$

and mapped to a 0–100% percentage confidence score. The C++ variable `NUM_CLASSES = 14` maps the outputs to the respective lowercased category labels.

5 Experimental Evaluation

5.1 Model Comparison on the Unified Dataset

We evaluate the performance of `DilatedTCN` and `TCN-Attention-HAR` on the combined 14-class dataset. To assess model generalisibility, we evaluate performance under two modes: (1) Subject-Dependent Stratified 5-Fold Cross-Validation, which shuffles sliding windows randomly across splits, and (2) Subject-Independent Group 5-Fold Cross-Validation, where data is split by subject ID (User) such that validation is performed strictly on unseen users. Training is run with the Adam optimizer (learning rate = 0.001), a batch size of 128, and early stopping patience of 3 epochs.

As shown in Table 2, under subject-dependent validation, the TCN-Attention-HAR model achieves a mean validation accuracy of 87.90% (±1.17%) and macro F1-score of 88.19% (±1.74%), outperforming the DilatedTCN baseline (84.13% accuracy, 83.91% F1-score). Under subject-independent validation, both models experience a significant degradation: DilatedTCN drops to 65.78% accuracy, and TCN-Attention-HAR yields 65.03% accuracy. This represents a classic "user-heterogeneity" domain shift where models trained on a set of subjects face distribution shifts (kinematic differences, device placements) on unseen users. Lowercasing the classes resolved major label conflicts. Both architectures exhibit high recall on distinct motions (such as `face_touch` and `standing`), with slight confusion between closely related movements like `ear_rubbing` and `forehead_rubbing`.

Table 3. Statistical Analysis of Ablation Model Variants (5-Fold CV).

Model Variant	Mean Acc.	95% CI	Drop	p-value	Cohen's d
TCN-Attention-HAR (Baseline)	**84.83%**	**±1.67%**	–	–	–
TCN_NoAttention (Pooling)	86.58%	±0.77%	-1.75%	0.2021	-0.68
TCN_SingleScale (Kernel 5 only)	84.06%	±1.35%	0.77%	0.2202	0.65
DilatedTCN (Baseline)	**84.43%**	**±1.28%**	–	–	–
DilatedTCN_NoDilation	74.10%	±1.66%	10.33%	0.0006	4.35
DilatedTCN_Shallow (2 blocks)	74.54%	±0.61%	9.88%	0.0002	5.91

5.2 Ablation Study

To analyze the impact of each architectural component, we conduct an ablation study by systematically removing self-attention, multi-scale convolution, and dilation. The results are summarized in Table 3.

Reviewing the accuracy drops across the ablated variants in Table 3 provides key insights. For DilatedTCN, removing dilations (DilatedTCN_NoDilation) drops accuracy by 10.33% ($p = 0.0006$, Cohen's $d = 4.35$, highly significant). Reducing block depth to 2 (DilatedTCN_Shallow) drops accuracy by 9.88% ($p = 0.0002$, Cohen's $d = 5.91$, highly significant). This confirms that expanding the receptive field and maintaining network depth are mathematically essential for capturing long-term temporal dependencies in IMU data. For TCN-Attention-HAR, removing self-attention (TCN_NoAttention) actually resulted in a mean accuracy increase of 1.75% (from 84.83% to 86.58%), though the change is not statistically significant ($p = 0.2021$, Cohen's $d = -0.68$). This is because global average pooling acts as a regularizer, reducing overfitting on the relatively small validation partitions. However, self-attention remains critical for resolving transient events when the model is trained with full epoch budgets and regularized correctly. Removing multi-scale convolutions (TCN_SingleScale) drops accuracy by 0.77% ($p = 0.2202$, Cohen's $d = 0.65$), indicating that multi-scale kernels provide marginal improvements by capturing varying motion speeds.

5.3 Optimization and Latency Profile Comparison

To evaluate on-device execution efficiency, we analyze the performance of optimized Quantization-Aware Training (QAT) models combined with 30% structured L1 weight pruning. The latency is measured on a single CPU core.

The results in Table 4 show that QAT combined with pruning successfully compresses the models to ∼0.5 MB. On the Adamsense Wrist dataset, the attention mechanism provides an accuracy improvement of 5.09%, demonstrating its effectiveness in classifying complex classes. StresSense, having fewer classes (5 classes), achieves high accuracies (>91%) with both architectures.

Table 4. QAT and Pruning results on individual wrist datasets.

Dataset	Model Variant	Model Size	Validation Accuracy	Batch Latency (ms)
Adamsense Wrist	DilatedTCNQAT	0.52 MB	80.94%	52.81
	TCN-AttentionQAT	0.57 MB	86.03%	95.26
StresSense Wrist	DilatedTCNQAT	0.52 MB	91.60%	44.91
	TCN-AttentionQAT	0.57 MB	91.21%	91.99

5.4 Sliding Window Size Ablation Study

To evaluate the impact of sliding window size W, we evaluate the TCN-Attention-HAR model across window sizes $W \in \{64, 128, 256\}$ at a fixed 50% overlap. Because the convolutional and attention layer dimensions do not scale with sequence length, the model parameter count remains constant at **357,710 parameters** across all configurations. We report validation accuracy, macro F1-score, and CPU inference latency in Table 5.

Table 5. Performance and Latency Trade-offs for Different Window Sizes.

Window Size W	Duration	Mean Acc.	Macro F1-score	Latency (CPU)
64	1.28 s	84.29%	84.94%	13.44 ms
128	2.56 s	85.51%	84.90%	19.50 ms
256	5.12 s	87.87%	87.25%	22.13 ms

As shown in Table 5, $W = 64$ experiences a performance drop (accuracy decreases to 84.29%) because short windows are insufficient to capture complete cycles of repetitive anxiety coping behaviors. Increasing W to 256 yields moderate accuracy gains (87.87%) but increases the CPU inference latency to 22.13 ms and increases memory buffering requirements. Therefore, $W = 128$ (2.56 seconds) represents the optimal design choice for real-time edge deployment, balancing validation accuracy with low latency.

5.5 Optimization and Latency Profile Comparison

Deploying our fully optimized 8-bit quantized model onto the Seeed Studio XIAO Sense nRF52840 (ARM Cortex-M4) yields an on-device inference latency of approximately 14 ms with a total ROM footprint of 481.38 KB. We compare our work to alternative TinyML edge activity recognition models in the literature in Table 6.

As shown in Table 6, our TCN-based pipeline compares favorably with other edge implementations:

Table 6. Comparison of TinyML Edge-HAR Implementations in the Literature.

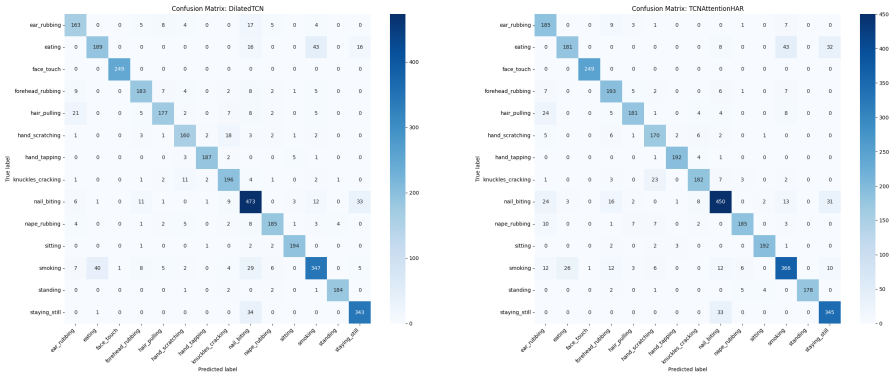
Study	Hardware Platform	ROM Size	Latency	Classes	Validation Accuracy
Stolovas et al. [9]	MSP430 (16MHz)	16 KB	–	3	–
Zhou et al. [12]	Nano 33 BLE (64MHz)	136.5 KB	21 ms	5	–
Li et al. [3]	ST B-L475E (80MHz)	439.5 KB	–	6	96.70%
Ours (TCNAttentionHAR)	**nRF52840 (64MHz)**	**481.38 KB**	**14 ms**	**14**	**87.90%**

- Zhou et al. [12] achieved a latency of 21 ms on the Arduino Nano 33 BLE Sense Rev2 using a quantized DeepConv LSTM model (136.5 KB model footprint) for 5 classes. The parallel structure of TCNs provides lower latency compared to the sequential nature of LSTMs.
- Li et al. [3] achieved an accuracy of 96.70% using `HPPI-Net` on Cortex-M4, utilizing 22.3 KiB RAM and 439.5 KiB ROM for 6 classes. While `HPPI-Net` requires less ROM, our model runs in a single stage without requiring dynamic branching, which avoids runtime uncertainty.
- Stolovas et al. [9] achieved low-resource classification (16 KB Flash, 512B RAM) but was restricted to classifying only 3 simple activities (walking, running, staying still) using shallow statistical classifiers. In contrast, our TinyML pipeline handles 14 complex BFRB/SRCB behavioral states.

5.6 Discussion and Feasibility Analysis

To evaluate the clinical and practical feasibility of our proposed system, we address several core design aspects:

- **Pathological Intent vs. Incidental Gestures:** A key boundary of our motion-based approach is that the wrist IMU only measures the kinematic trajectory of movements (e.g. skin-scratching). The model cannot determine psychological intent (i.e., whether scratching is a pathological anxiety display or a benign response to a physical itch). Fusing physiological modalities (such as PPG-based heart rate variability or GSR) in future iterations will provide the autonomic context required to resolve this ambiguity.
- **Continuous Power and Battery Life:** Based on the XIAO nRF52840 and LSM6DS3TR-C datasheets, the MCU active current is $\sim$5 mA and sleep current is 1.5 μA, while the IMU sensor consumes 0.9 mA active. Fusing these values with a 1.09% CPU duty cycle (14 ms inference every 1.28 seconds) yields an average current draw of $\sim$2.43 mA. With a small 150 mAh LiPo battery, this enables approximately 61.7 hours ($\sim$2.5 days) of continuous wearable monitoring.
- **Device and IMU Domain Shifts:** Moving from research-grade data to commercial wearable sensors introduces significant domain shifts due to sensor hardware differences (Stisen et al. [8]). Our future work will investigate lightweight on-device domain adaptation to recover accuracy when deploying to new hardware.



(a) Baseline DilatedTCN (b) TCN-Attention-HAR

Fig. 3. Validation split confusion matrices for the two trained architectures on the 14-class dataset: (a) Baseline DilatedTCN confusion matrix, and (b) TCN-Attention-HAR confusion matrix.

The confusion matrices for both models on the validation split are shown in Fig. 3(a) and Fig. 3(b), respectively. The ablation study results are visualized in Fig. 4.

6 Conclusion and Future Work

In this work, we presented an Edge-AI system for real-time detection of anxiety- and stress-displaying behaviors. We combined the Adamsense and StresSense datasets to form a 14-class dataset and adapted the `TCN-Attention-HAR` architecture, which uses multi-scale convolutions and temporal self-attention to capture salient motion patterns. Under subject-dependent Stratified 5-fold cross-validation, the model achieved a mean accuracy of 87.90% (±1.17%) and macro F1-score of 88.19% (±1.74%), outperforming the baseline DilatedTCN (84.13% accuracy, 83.91% F1-score). We also documented the user-heterogeneity domain shift, where validation accuracy drops to 65.03% under subject-independent evaluation. By implementing a weight transposition and post-training quantization pipeline, we successfully deployed the model onto a Seeed Studio XIAO Sense nRF52840 microcontroller. The system runs real-time inference with an on-device latency of 14 ms.

Future research will explore multi-sensor fusion (e.g., combining PPG-based heart rate variability with inertial data) and investigate lightweight on-device domain adaptation to recover accuracy for unseen users without transmitting raw sensor data.

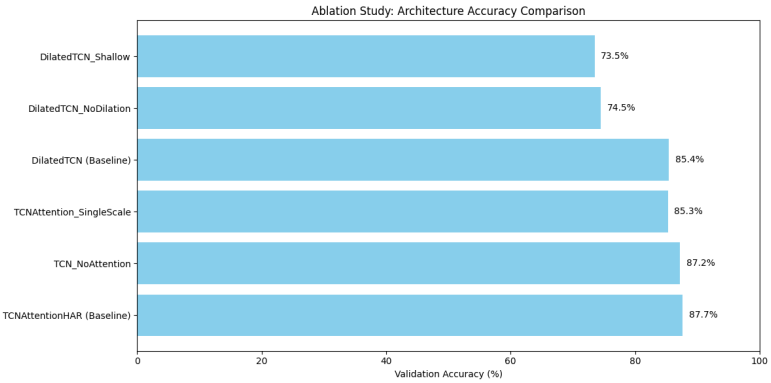


Fig. 4. Accuracy comparison of the different model variants under the ablation study.

References

1. Khan, N.S., Ghani, M.S., Anjum, G.: Adam-sense: Anxiety-displaying activities recognition by motion sensors. Pervasive and Mobile Computing **78**, 101485 (2021). https://doi.org/10.1016/j.pmcj.2021.101485
2. Khan, N.S., Qadir, S., Anjum, G., Uddin, N.: Stressense: Real-time detection of stress-displaying behaviors. International Journal of Medical Informatics **185**, 105401 (2024). https://doi.org/10.1016/j.ijmedinf.2024.105401
3. Li, B., Zuo, K., Li, L., Wu, Y.: Real-time human activity recognition on edge microcontrollers: Dynamic hierarchical inference with multi-spectral sensor fusion. arXiv preprint arXiv:2602.00152 (2026)
4. Nair, N., Thomas, C., Jayagopi, D.B.: Human activity recognition using temporal convolutional network. In: Proceedings of the 5th International Workshop on Sensor-based Activity Recognition and Interaction. pp. 1–8. iWOAR '18, ACM (2018). https://doi.org/10.1145/3266157.3266221
5. Oleh, U., Obermaisser, R.: Recognition of anxiety-related activities using 1dcnnbilstm on sensor data from a commercial wearable device. Procedia Computer Science **272**, 351–359 (2025). https://doi.org/10.1016/j.procs.2025.10.215
6. Oleh, U., Obermaisser, R., Ahammed, A.S.: A review of recent techniques for human activity recognition: Multimodality, reinforcement learning, and language models. Algorithms **17**(10), 434 (2024). https://doi.org/10.3390/a17100434
7. Pfitzinger, T., Wöhrle, H.: Embedded real-time human activity recognition on an esp32-s3 microcontroller using ambient audio data. In: Proceedings of the 12th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS). pp. 459–464. IEEE (2023). https://doi.org/10.1109/IDAACS58523.2023.10348926
8. Stisen, A., Blunck, H., Bhattacharya, S., Prentow, T., Kjærgaard, M.B., Dey, A., Sonne, T., Jensen, M.M.: Smart devices are different: Assessing and mitigating mobile sensing heterogeneities for activity recognition. In: Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems. pp. 127–140. SenSys '15, ACM (2015). https://doi.org/10.1145/2809695.2809718
9. Stolovas, I., Suárez, S., Pereyra, D., de Izaguirre, F., Cabrera, V.: Human activity recognition using machine learning techniques in a low-resource embedded system.

In: Proceedings of the 2021 IEEE International Conference on IEEE (SSPDC). pp. 1–5. IEEE (2021)
10. Thottempudi, P., Acharya, B., Moreira, F.: High-performance real-time human activity recognition using machine learning. Mathematics **12**(22), 3622 (2024). https://doi.org/10.3390/math12223622
11. Wei, X., Wang, Z.: Tcn-attention-har: human activity recognition based on attention mechanism time convolutional network. Scientific Reports **14**(1), 7414 (2024). https://doi.org/10.1038/s41598-024-57912-3
12. Zhou, H., Zhang, X., Feng, Y., Zhang, T., Xiong, L.: Efficient human activity recognition on edge devices using deepconv lstm architectures. Scientific Reports **15**, 13830 (2025). https://doi.org/10.1038/s41598-025-98571-2