STORM: Hardware-Aware Tiny Transformer Co-Design for Low-Power Inertial Human Activity Recognition

electronics

MDPI

Article

STORM: Hardware-Aware Tiny Transformer Co-Design for Low-Power Inertial Human Activity Recognition

Alessandro Varaldi [1,*], Claudio Genta [2], Alberto Manzone [2] and Marco Vacca [1]

[1] Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy; marco.vacca@polito.it
[2] Ideas & Motion, 12062 Cherasco, Italy; claudio.genta@ideasandmotion.com (C.G.); alberto.manzone@ideasandmotion.com (A.M.)
* Correspondence: alessandro.varaldi@polito.it

Abstract

Human Activity Recognition (HAR) from inertial sensors must run continuously on battery-powered wearables under tight latency, memory, and energy budgets. While tiny Transformers can be effective on inertial time series, end-to-end co-design across quantized inference and heterogeneous low-power platforms remains underexplored. We present STORM (Small Transformer for On-node Recognition of Motion), a deployment-oriented 19.7k-parameter 1D Transformer co-designed with X-HEEP, an open-source low-power single-core RISC-V SoC, and a tightly coupled streaming CGRA for nonlinear primitives (e.g., softmax). We build a cross-source 8-class benchmark by harmonizing 3 public datasets under a stringent, deployment-aligned protocol that exposes both cross-subject and cross-source shift. Using 1.280 s windows with 0.640 s stride, the protocol models continuous on-node HAR under cross-dataset generalization. After quantization-aware training and INT8 C inference export, STORM achieves 0.799/0.801 accuracy/macro-F1 on this benchmark. Deployed on an FPGA prototype of X-HEEP with the streaming CGRA backend, STORM requires 67.4 ms per inference at 100 MHz, while activity-based power analysis estimates a total inference energy of 632.4 µJ, satisfying the stride-driven real-time constraint. These results support the practical viability of compact attention-based HAR on low-power wearable-class embedded platforms.

Keywords: Human Activity Recognition (HAR); wearable devices; edge AI; TinyML; quantization-aware training; INT8 inference; hardware-software co-design

Academic Editor: Chen Yang

Received: 30 March 2026
Revised: 29 April 2026
Accepted: 30 April 2026
Published: 1 May 2026

1. Introduction

Human Activity Recognition (HAR) from wearable inertial sensors enables continuous services such as fitness coaching, rehabilitation adherence tracking, fall-risk and sedentary-behavior monitoring, and on-site safety supervision. In these settings, the sensing node must run always-on under tight battery, latency, and memory budgets, while preserving privacy by avoiding raw-signal streaming [1]. Under continuous operation, the real-time requirement is directly dictated by the sliding-window inference pipeline: with overlapping windows, the device must reliably produce one prediction per stride interval to sustain on-node monitoring. Meeting this constraint is further complicated by sensor variability, including placement changes, motion artifacts, and realistic sensing faults.

Attention-based models and Transformer encoders have recently shown strong performance on inertial time series by capturing long-range dependencies and focusing on the

most informative portions of the signal [2–4]. At the same time, the embedded HAR literature demonstrated that compact networks can be deployed on ultra-low-power platforms with aggressive quantization and careful kernel optimization [5]. Despite these advances, most Transformer-based HAR studies are evaluated in isolation, often on single datasets and fixed sensor setups, and primarily report accuracy without characterizing (1) robustness under sensor configuration changes and realistic faults, and (2) end-to-end latency and energy on concrete low-power platforms. Conversely, MCU-focused deployments either conclude that Transformers are unattractive under strict budgets [6] or optimize CPU-only execution without coupling the model design to a heterogeneous architecture that can selectively accelerate critical operators [7]. On the hardware side, coarse-grained reconfigurable arrays (CGRAs) and Decoupled Access-Execute accelerators have been explored for high-throughput streaming and dense loop kernels [8], but their interaction with attention-heavy pipelines, including mixed-precision boundaries and weight movement, under tight SRAM and bandwidth constraints remains underexplored.

A central challenge is that tiny Transformer performance on wearables is governed by coupled, stack-wide choices: windowing and dataset harmonization shape signal statistics, quantization and operator approximations shift class-wise errors, and the platform memory hierarchy determines whether compute or data movement dominates. Optimizing components in isolation can therefore misrepresent real-time feasibility and energy. This motivates end-to-end co-design in which the data pipeline, model, numeric path, and heterogeneous mapping are developed jointly and validated under continuous-inference constraints.

In this work, we pursue this co-design perspective and present STORM (Small Transformer for On-node Recognition of Motion), a deployment-oriented tiny 1D Transformer for inertial HAR co-designed with X-HEEP, an open-source low-power single-core RISC-V SoC [9], and a tightly coupled streaming CGRA inspired by Mage [10]. Our contributions are as follows:

- Tiny Transformer co-designed for deployment: STORM is a 19.7k-parameter 1D Transformer tailored to MCU-class constraints and integer-friendly inference. On the cross-source benchmark, it reaches 0.802/0.804 (FP32) and 0.799/0.801 (exported INT8 C after QAT).
- Cross-source benchmark and robustness: we harmonize 3 public datasets into an 8-class, 6-channel benchmark with subject-independent, source-stratified splits and 1.280 s windows/0.640 s stride, and evaluate realistic sensor degradations (outages, dropout, noise, stuck-at, drift).
- End-to-end RISC-V + CGRA deployment: we deploy on an FPGA prototype of X-HEEP coupled with a streaming CGRA backend, offloading nonlinearities and reporting end-to-end cycles, bottlenecks, and inference-only energy estimates, exposing MHSA as the dominant residual cost after nonlinearity offload.
- Open-source release: code, benchmark generation, QAT, and deployment are available at https://github.com/AlessandroVaraldi/storm (accessed on 1 May 2026).

The remainder of this paper is organized as follows. Section 2 reviews related work on inertial HAR, TinyML, and CGRA-based acceleration. Section 3 describes the cross-source benchmark, sensor and failure models, the STORM architecture, and the deployment flow. Section 4 reports accuracy, robustness, on-board performance measurements, and activity-based energy estimates for the deployed configuration. Section 5 contextualizes these results within the broader design space of Tiny Transformer deployments and discusses open limitations. Section 6 concludes and outlines future directions.

2. Background and Related Work

Research on inertial sensor-based HAR has progressively moved from hand-crafted pipelines to end-to-end deep models that learn temporal representations directly from raw wearable signals. Early deep approaches established convolutional and recurrent networks as strong alternatives to manual feature engineering. Convolutional models were shown to capture local motion patterns effectively, while recurrent units improved temporal aggregation over longer activity segments and transitions [11,12]. Later surveys organized this design space and highlighted the factors that most strongly affect performance in wearable HAR, including sensor modality and placement, temporal context, subject variability, and the trade-off between recognition accuracy and computational cost [13–15].

Building on these foundations, the literature increasingly explored architectures that combine efficient local feature extraction with more expressive temporal modeling. In this context, convolutional front-ends remained attractive because they map naturally to sliding-window inertial data, while recurrent and attention-based modules helped capture longer-range dependencies and cross-channel interactions. Representative examples include attention-based multimodal HAR models such as Attend-and-Discriminate and DanHAR, which use attention to emphasize informative channels and time steps in wearable sensor streams [16,17]. Transformer-style encoders extend this direction by replacing recurrence with self-attention. TRASEND combines multimodal attention with user adaptation and improves over CNN–RNN baselines [3], Dirgová Luptáková et al. show that self-attention is effective for inertial HAR [2], and XTinyHAR targets model compactness through knowledge distillation [4]. These works demonstrate the modeling potential of attention for wearable HAR, but they generally focus on recognition performance under dataset-specific settings rather than on deployment under strict embedded constraints.

A separate line of work addresses HAR from a TinyML and embedded-systems perspective. Here the central issue is not only predictive performance, but whether a model remains practical after quantization, memory fitting, scheduling, and low-power execution constraints are taken into account. Early studies already emphasized the need for resource-aware HAR pipelines on wearable devices [18]. More recently, Daghero et al. studied compact 1D CNNs on ultra-low-power RISC-V MCUs with mixed-precision and sub-byte quantization [5]. Lattanzi et al. compared Transformers with convolutional and recurrent baselines on ESP32 and found that attention is not automatically advantageous under strict resource budgets [6]. In contrast, Jung et al. showed that encoder-only Tiny Transformers can become viable on low-power MCUs when the numeric path, scheduling, and memory management are co-optimized [7]. Taken together, these studies suggest that the usefulness of Transformers in embedded HAR depends strongly on the deployment boundary, including the quantization scheme, kernel implementation, peak activation footprint, and measurement methodology.

From the hardware viewpoint, FPGA and accelerator-based solutions offer tighter control over precision, memory traffic, and operator placement than CPU-only implementations. Ling et al. automate Tiny Transformer deployment on embedded FPGAs through hardware-aware search, quantization-aware training, and RTL generation [19]. In parallel, CGRAs and Decoupled Access-Execute architectures have been investigated to improve the efficiency of dense and streaming kernels under tight on-chip memory constraints. Recent work on Mage demonstrated this approach for TCN inference mapped to streaming 1D convolutions on a tightly coupled CGRA [20]. However, this does not directly answer how similar hardware behaves on attention-heavy pipelines, where tensor reordering, normalization, softmax, and mixed-precision boundaries all contribute to the critical path. For compact Transformers, these effects become especially relevant because SRAM limits constrain model width, buffering, and data movement strategies.

Overall, the literature leaves a gap at the intersection of three research directions: accurate wearable HAR models, resource-constrained TinyML deployment, and accelerator-aware hardware mapping. Prior HAR studies often evaluate attention-based models under dataset-centric conditions, prior TinyML studies frequently focus on CNN-oriented or CPU-only deployments, and prior accelerator studies rarely adopt a robustness-oriented HAR protocol spanning multiple sources and sensing faults. This work targets that intersection by co-designing a compact Transformer with a low-power RISC-V SoC and a streaming CGRA, and by evaluating the resulting pipeline under cross-source, subject-independent splits and controlled sensor degradations. A structured positioning of STORM within this design space is provided in Section 5.

3. Methodology

3.1. Cross-Source HAR Benchmark

We build the cross-source HAR benchmark by harmonizing 3 public inertial datasets: MotionSense [21], UCI HAR [22], and PAMAP2 [23]. The three sources differ in acquisition format, temporal resolution, and activity taxonomy, and are therefore processed independently before being merged into a common benchmark. To enforce a shared sensing interface and avoid dataset-specific modality advantages, all sources are reduced to a fixed 6-channel input composed of tri-axial accelerometer and tri-axial gyroscope signals, representative of a commodity 6-axis IMU.

UCI HAR is distributed as pre-segmented inertial windows at 50 Hz and is converted to the common 6-channel representation using total acceleration and body gyroscope signals. MotionSense provides continuous trial-wise recordings at 50 Hz, from which we extract user-acceleration and rotation-rate channels. PAMAP2 is acquired at 100 Hz as continuous timestamped streams; from this source we retain the hand IMU accelerometer and gyroscope signals, using interpolation to handle missing samples before temporal alignment. Table 1 summarizes the three datasets before harmonization.

To ensure a common temporal resolution, all signals are resampled to 50 Hz prior to window extraction. Sliding windows of duration 1.280 s and stride 0.640 s are then generated, yielding fixed-length inputs with $T = 64$ time steps per window. The selected window length reflects a trade-off between temporal context and embedded cost. A 1.280 s window is long enough to capture short motion primitives, posture segments, and meaningful portions of periodic activities such as gait, while remaining short enough to limit activation footprint, latency, and buffering requirements on the target platform. The selected stride corresponds to a 50% overlap, which is a common compromise in continuous HAR: it reduces boundary effects and improves temporal continuity between adjacent decisions without doubling the inference rate required by non-overlapping windows of the same duration. Under this setting, the evaluation explicitly models continuous on-node HAR, where the classifier must satisfy a stride-driven real-time deadline while exploiting partial context reuse across overlapping windows.

Because the original datasets expose different activity taxonomies, we map them onto the unified 8-class label space reported in Table 2: *walking*, *running*, *upstairs*, *downstairs*, *sitting*, *standing*, *lying*, and *other*. Activities with a direct correspondence are mapped to the first seven classes, while source-specific activities without a one-to-one counterpart are assigned to *other*. This choice makes the benchmark more representative of continuous wearable deployment, where not all observed motion belongs to a closed target taxonomy.

After preprocessing and windowing, the dataset is represented as $X \in \mathbb{R}^{L \times C \times T}$, where L is the number of windows, $C = 6$, and $T = 64$. The train/validation/test split is performed at subject level with no subject overlap across partitions. To preserve source diversity and prevent a single dataset from dominating one split, partitioning is applied

independently within each source and then merged into the final train, validation, and test sets. In the reference configuration, we use 70%/15%/15% train/validation/test partitions.

After splitting, we compute per-channel mean and standard deviation on the training set only and use these statistics to normalize validation and test windows. This mirrors deployment, where normalization parameters are frozen after training and the model must remain stable under subject and source shifts not seen during optimization.

Table 1. Summary of the source datasets before harmonization.

Dataset	Subjects	Native Rate	Used Channels
UCI HAR	30	50 Hz	Total acceleration and body gyroscope
MotionSense	24	50 Hz	User acceleration and rotation rate
PAMAP2	9	100 Hz	Hand IMU accelerometer and gyroscope

Table 2. Cross-source benchmark label space and class distribution.

Class ID	Class Name	# Windows
0	walking	16,096
1	running	5648
2	upstairs	8164
3	downstairs	7039
4	sitting	15,175
5	standing	14,371
6	lying	4941
7	other	16,096

3.2. Sensor Configurations and Failure Models

Beyond the cross-source benchmark, we report two complementary evaluations. First, on the cross-source benchmark we compare two sensing configurations under the same harmonized pipeline: the nominal Acc+Gyro setup with 6 channels, and an Acc-only setup with three accelerometer channels. This comparison isolates the contribution of gyroscope information under the same cross-subject and cross-source generalization setting, and matches the sensor-subset analysis reported in Table 3.

Table 3. Sensor-subset comparison on the cross-source benchmark and single-dataset reference performance without harmonization.

Group	Setting/Dataset	Accuracy	Macro-F1
Cross-source benchmark—sensor subsets			
Cross-source	Acc+Gyro (6ch)	0.802	0.804
Cross-source	Acc-only (3ch)	0.556	0.542
Individual datasets (no harmonization)			
UCI HAR	full	0.929	0.935
MotionSense	full	0.811	0.817
PAMAP2	full	0.887	0.751

Second, we report reference performance on the original datasets without cross-source harmonization, namely UCI HAR, MotionSense, and PAMAP2. In these single-dataset evaluations, each dataset is processed independently using the same temporal preprocessing, subject-wise 70%/15%/15% train/validation/test protocol, and per-channel standardization adopted for the cross-source benchmark, while retaining the dataset-specific full inertial configuration used by the corresponding experiment. These results provide an

upper-bound reference under reduced distribution shift and help quantify the additional difficulty introduced by cross-source harmonization.

To assess robustness under realistic hardware and sensing issues, we inject controlled failures at test time on the cross-source benchmark while keeping the training protocol unchanged. Starting from the same subject-wise splits, failures are applied in the raw sensor domain and the resulting signals are then normalized with the statistics computed on the training split. This ordering preserves a fixed deployment-oriented normalization pipeline while exposing the model to perturbations in the measured signal before standardization. We consider six families of failures: modality outages, single-axis outages, intermittent dropout, additive Gaussian noise, stuck-value faults, and scale drift.

The channel convention used in the fault analysis is the same as in the harmonized 6-channel input: channels 0–2 correspond to the accelerometer axes (a_x, a_y, a_z), while channels 3–5 correspond to the gyroscope axes (g_x, g_y, g_z). Accelerometer-level faults are therefore applied to channels $\{0, 1, 2\}$, gyroscope-level faults to channels $\{3, 4, 5\}$, and full-sensor faults to all six channels. Single-axis outages are evaluated by disabling one channel at a time. Table 4 reports the specific parameter settings used for each fault family.

Table 4. Parameter settings used for the controlled test-time sensor-failure analysis. Faults are injected before applying the training-split normalization.

Fault Family	Affected Channels	Severity	Rule
Modality outage	Acc: 0–2; Gyro: 3–5; All: 0–5	Full window	Selected channels are set to zero.
Single-axis outage	One channel in 0–5	Full window	Each axis is disabled independently.
Intermittent dropout	Acc: 0–2 or Gyro: 3–5	30% timesteps	A Bernoulli temporal mask is shared by the selected modality channels.
Gaussian noise	Acc: 0–2 or Gyro: 3–5	$\sigma = 0.5$	Zero-mean noise is added before normalization, with an amplitude corresponding to $\sigma = 0.5$ in normalized-channel units.
Stuck-value fault	Acc: 0–2 or Gyro: 3–5	$\tau \sim \mathcal{U}(0.25T, 0.75T)$	Samples after τ repeat the last valid value.
Scale drift	Acc: 0–2 or Gyro: 3–5	$1.0\times \rightarrow 2.0\times$	A linear gain ramp is applied over the window.

The selected perturbation levels are intended as controlled stress-test conditions rather than as a fitted statistical model of a specific commercial IMU. In particular, modality and axis outages represent hard sensor or acquisition failures; intermittent dropout approximates packet loss or missed samples; additive Gaussian noise models degraded sensing and noise amplification; stuck-value faults emulate a frozen sensor output; and scale drift represents a progressive calibration or gain error within the window. For stochastic perturbations, namely intermittent dropout, Gaussian noise, and stuck-value faults, the random selection is performed independently across test windows using a fixed random seed for reproducibility. Deterministic perturbations, such as hard outages and scale drift, are applied identically across all windows of the corresponding test condition.

This fault suite is not meant to exhaustively cover all possible IMU failure modes. Rather, it provides a reproducible set of severity levels for comparing the relative sensitivity of the deployed HAR pipeline to accelerometer and gyroscope degradations under the same cross-source, subject-independent evaluation protocol.

3.3. Evaluation Metrics

We report overall accuracy and macro-averaged F1 score. Accuracy measures the fraction of correctly classified windows over the whole test set:

$$\text{Accuracy} = \frac{\sum_{c=1}^{K} TP_c}{N}, \tag{1}$$

where TP_c is the number of true positives for class c, K is the number of classes, and N is the total number of evaluated windows.

Since the considered HAR benchmarks are not perfectly class-balanced, we also report macro-F1, which gives equal importance to all classes. For each class c, precision, recall, and F1 are defined as:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \qquad R_c = \frac{TP_c}{TP_c + FN_c}, \tag{2}$$

$$F1_c = \frac{2P_cR_c}{P_c + R_c}. \tag{3}$$

The macro-F1 score is then computed as:

$$\text{Macro-F1} = \frac{1}{K}\sum_{c=1}^{K} F1_c. \tag{4}$$

Accuracy provides a compact measure of global performance, whereas macro-F1 better reflects balanced recognition quality across both frequent and underrepresented activities. For this reason, macro-F1 is used as the main class-balanced metric throughout the paper.

3.4. STORM Architecture

The core model in this work is STORM (Figure 1), a compact 1D Transformer tailored to inertial HAR and to the constraints of the X-HEEP system-on-chip. It is designed to capture temporal dependencies in multi-axis inertial signals over short-to-medium windows (1.280 s) that cover multiple gait cycles, posture changes and short transitions, while using only building blocks that map efficiently to integer 1D-convolution and matrix-multiplication primitives. Across the experiments considered in this work, the STORM backbone is kept unchanged, while the input and output interfaces follow the sensing configuration and label space of the specific evaluation setting.

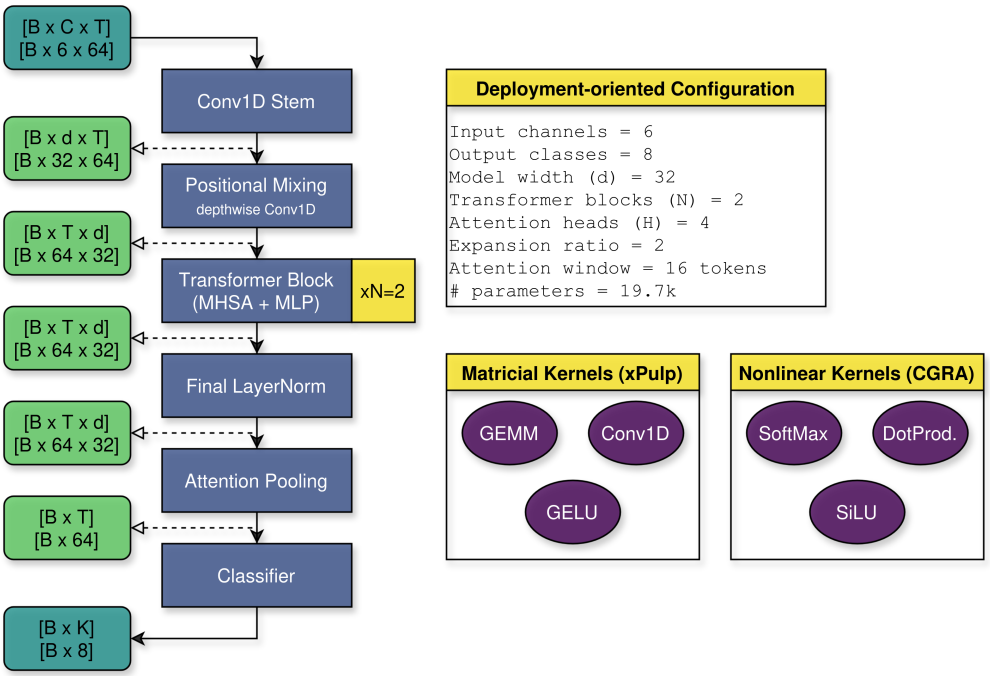

Figure 1. STORM architecture. A convolutional stem and positional mixing feed a stack of N Transformer blocks with MHSA, followed by attention pooling and a linear classifier. The deployment-oriented configuration is chosen from the design-space sweep over width, depth, and attention heads, and from targeted ablations on attention window, pooling, positional mixing, and LayerNorm.

STORM takes as input fixed-length windows of inertial data with shape $[B, C, T]$, where B is the batch size, C is the number of input channels, and T is the number of time steps. A lightweight Conv1D stem applies temporal convolution and SiLU activation, expanding the channels to the internal model dimension d and acting as a local temporal feature extractor. A depthwise 1D convolutional positional mixing stage adds channel-wise temporal filters residually to the token representations, providing a convolutional positional encoding compatible with quantization-aware training (QAT).

The main body of the network is a stack of N identical Transformer blocks with pre-activation Layer Normalization, a multi-head self-attention (MHSA) module, and a feed-forward MLP. MHSA is implemented as scaled dot-product attention with a softmax over the temporal dimension. The MLP is a two-layer feed-forward network with GELU activation. After the last block, token embeddings are normalized and aggregated over time via attention pooling. A small scoring network with a GELU nonlinearity produces a scalar weight for each time step; a softmax over time yields attention weights used to compute a weighted average of the token embeddings, resulting in a fixed-dimensional feature vector for the window. A dedicated Layer Normalization is applied to the pooled feature vector before a final linear layer maps it to K activity logits, where K depends on the label space of the considered evaluation setting.

3.5. Deployment-Oriented Co-Design and Quantization

We target a single-node wearable scenario with a 6-axis IMU (accelerometer + gyroscope), running continuous human activity recognition on the cross-source benchmark under strict memory, compute, and energy constraints. The deployment-oriented STORM configuration is driven by two coupled considerations. First, the streaming CGRA is most effective when offloading control- and math-heavy nonlinear primitives, notably SiLU, GELU, and softmax, which otherwise dominate the residual latency once matrix kernels are optimized on the core. This motivates an architecture that keeps the main linear algebra (Conv1D, dot-products and linear layers) friendly to SIMD/DSP execution, while exposing well-defined nonlinearity stages that can be executed on the CGRA with low orchestration overhead. Second, the on-chip SRAM budget of the X-HEEP system bounds the peak activation footprint and the buffering required by overlapping windows in continuous inference, strongly constraining width and depth. As a result, we adopt a compact STORM instance as the deployment baseline, sized so that both model parameters and worst-case activations fit within the on-chip SRAM budget while retaining GEMM shapes that efficiently utilize the Xpulp SIMD/DSP datapath and keep CGRA offload overhead low.

A key co-design choice in the deployment-oriented STORM variant is the use of local windowed attention with window size 16 tokens, instead of full-window global attention over all 64 time steps. This restriction is introduced specifically for the deployment-oriented configuration to reduce MHSA latency, simplify data movement, and lower the orchestration cost on the platform. The choice reflects the target embedded constraints rather than an architectural preference in isolation, and trades part of the long-range temporal modeling capacity for improved real-time feasibility.

To enable an almost integer-only inference pipeline, we employ a deployment-oriented QAT flow based on symmetric INT8 fake-quantization. We apply input fake-quantization (INT8) with a percentile-based scale and activate lightweight EMA-based fake-quant hooks on selected module outputs, while using standard time-domain augmentation (amplitude rescaling, additive jitter, mild time warping and short time masking) and class-balanced sampling; macro-F1 on validation subjects is used for model selection.

The deployment-oriented STORM is exported from PyTorch to a self-contained C header for inference on X-HEEP. Starting from the best QAT-tuned checkpoint, the exporter

reconstructs the architecture and runs a short calibration pass on a subset of validation windows to estimate per-layer activation ranges and the input range. These statistics define symmetric INT8 activation scales. We adopt uniform symmetric INT8 quantization with zero-point fixed to zero and 32-bit accumulators: activations are quantized per tensor using the calibrated scales, while weights are quantized per output channel and biases are stored as 32-bit integers. For each convolutional and linear layer, the exporter precomputes an integer multiplier and shift implementing the rescaling between the accumulator and the next-layer activation scale as a pure integer operation with saturation. To exploit SIMD/DSP extensions, all INT8 weights are offline-packed to match hand-optimized kernels on the RISC-V core. Nonlinearities and softmax are implemented via fixed-point lookup tables. LayerNorm uses an integer-affine kernel to reduce per-channel floating-point cost. The inverse standard deviation $1/\sqrt{\sigma^2+\epsilon}$ is obtained from a log-uniform lookup table in SRAM indexed via a fast IEEE754-based $\ln(\cdot)$ approximation, while the affine step is applied in fixed-point (Q14) with pre-quantized per-channel γ and β and an int32/int64 inner loop with rounding and int8 saturation.

3.6. Hardware Platform and CGRA Integration

Cycle count, latency, and throughput are measured on an AMD ZCU104 board (Zynq UltraScale+ XCZU7EV), where the heterogeneous RISC-V system is implemented in programmable logic. Inference energy, instead, is not directly measured on silicon: it is estimated through activity-based power analysis targeting a commercial 65 nm standard-cell flow. Specifically, post-synthesis simulation is run in QuestaSim to collect switching activity over the inference interval, and the resulting activity is back-annotated in PrimePower to obtain dynamic and leakage power. The reported energy numbers therefore represent full-system inference estimates for the deployed hardware configuration, including the CPU, memories, DMA, interconnect, and the CGRA, rather than direct board-level or post-silicon measurements.

X-HEEP is configured with 8 tightly coupled SRAM banks (256 kB total) and a lightweight peripheral subsystem. Computation runs on a cv32e40p RISC-V core with single-precision FPU and Xpulp SIMD/DSP extensions, used to implement the main INT8 linear kernels of STORM, namely Conv1D, dot-products, and linear layers.

The hardware backend couples X-HEEP with a tightly coupled streaming CGRA used for integer approximations of nonlinear activation functions. In this work, the CGRA is used as a nonlinear-function offload engine rather than as a general matrix accelerator. It targets recurring patterns in activation kernels, such as multiply-adds, shifts, comparisons, clipping, and division-like steps, which appear in SiLU, GELU, and softmax approximations. Data are exchanged with the X-HEEP memory system through DMA-managed streams, allowing nonlinear kernels to be executed with limited core intervention while leaving the main INT8 linear-algebra path on the RISC-V core.

In STORM, the streaming CGRA offloads the nonlinear stages of the inference pipeline, specifically SiLU, GELU, and softmax. These operations have limited arithmetic intensity but are costly in software due to control flow and transcendental approximations. Linear algebra remains on the cv32e40p, where Xpulp SIMD/DSP extensions efficiently handle INT8 matrix and convolution kernels. Data transfers between X-HEEP SRAM banks and the CGRA use DMA, enabling low-overhead streaming execution with minimal core intervention.

Memory and layout choices are co-designed with this execution model. In the deployed configuration, model parameters are kept resident in on-chip SRAM during inference, avoiding external weight transfers in the steady-state execution path. Activations adopt a channel-major $[C, T]$ layout to simplify banking and DMA transfers; a lightweight

per-token gather is applied only where required by dot-product kernels. The software framework also supports storing model parameters in external flash and transferring them through QSPI when larger configurations exceed the available SRAM budget.

3.7. Energy Analysis

Inference energy is estimated through activity-based analysis rather than measured directly on silicon. We explicitly distinguish measured and estimated quantities: cycle counts and latency are obtained from the FPGA prototype, whereas average power is estimated from post-synthesis switching activity targeting a commercial 65 nm standard-cell flow. Starting from the synthesized implementation of the complete system, we run post-synthesis simulation in QuestaSim over the steady-state interval corresponding to one full STORM inference. The resulting switching activity is back-annotated in PrimePower to estimate average dynamic and leakage power for the full system, including the cv32e40p core, on-chip SRAM banks, DMA subsystem, interconnect, and the CGRA.

Clock gating is enabled in the synthesized design and is therefore included in the activity-based power estimate. As a result, idle or inactive portions of the system contribute according to their gated switching activity and leakage, rather than being modeled through a purely cycle-proportional or always-active approximation. This is important for the heterogeneous execution path, where the RISC-V core, DMA, SRAM banks, and the CGRA are not active with the same intensity during all kernel phases.

The analysis boundary is restricted to steady-state inference. Boot, program loading, memory initialization, and peripheral bring-up are excluded so as to isolate the per-window cost of continuous HAR. In the deployed configuration, all model parameters remain resident in on-chip SRAM during inference; consequently, the reported energy figures do not include external weight-transfer overhead and reflect the steady-state execution path of the deployed model. The reported inference energy is therefore obtained by combining the estimated average full-system power with the measured inference latency:

$$E_{\text{inf}} = P_{\text{avg}}\, T_{\text{inf}}. \tag{5}$$

To isolate the contribution of the CGRA, we also evaluate a no-CGRA execution path in which the same nonlinear kernels are executed on the RISC-V core instead of being offloaded. This comparison uses the same model, input window, memory-resident weights, clock frequency, and activity-based power-estimation methodology. Therefore, the difference between the two configurations reflects the effect of nonlinear-kernel offload under the same steady-state inference boundary.

In addition to the end-to-end estimate, we derive a kernel-level energy profile using the same execution trace employed for cycle counting. For each kernel type k, we identify the corresponding activity window, extract the average power over that interval, and multiply it by the measured kernel latency. The total energy contribution of kernel k is then obtained by accumulating over all its invocations:

$$E_k = N_k\, P_k\, T_k, \tag{6}$$

where N_k is the number of invocations, P_k is the average power over the kernel execution window, and T_k is the kernel latency per invocation. This procedure yields a kernel-resolved energy attribution that is consistent with the full-system activity-based estimate, rather than a post-hoc proportional allocation based only on cycle share.

4. Results

4.1. Architectural Trade-Offs, Baselines, and Quantization

To justify the deployed STORM design under the same cross-source protocol, we analyze its main architectural knobs, evaluate the sensitivity to the temporal windowing parameters, compare it with lightweight matched-footprint baselines, and then quantify the effect of the quantized deployment path. Unless otherwise noted, all comparisons use the same input interface, preprocessing, subject-wise splits, and training setup adopted for STORM.

Table 5 reports the main design-space sweep over model width d, encoder depth N, and number of attention heads H. The results indicate that width improves recognition quality at a steep parameter cost, depth brings smaller gains, and multi-head decomposition is the most important architectural knob at fixed footprint. On this basis, the selected STORM configuration uses $d = 32$, $N = 2$, and $H = 4$.

Table 5. Main design-space exploration of STORM on the cross-source benchmark. The deployment-oriented STORM version is indicated in bold.

Variant	d	N	H	Params	Acc.	F1	QAT Acc.	QAT F1
STORM-base	32	2	2	19,753	0.768	0.767	0.771	0.770
STORM-w64	64	2	2	72,201	0.780	0.790	0.776	0.794
STORM-w16	16	2	2	5817	0.744	0.757	0.746	0.760
STORM-d4	32	4	2	36,841	0.775	0.793	0.771	0.790
STORM-d1	32	1	2	11,209	0.764	0.781	0.771	0.787
STORM-h4	32	2	4	19,753	0.802	0.804	0.802	0.804
STORM-h1	32	2	1	19,753	0.690	0.693	0.688	0.691

Since the temporal window length and stride affect both recognition quality and deployment cost, we also perform a sensitivity analysis around the selected windowing configuration. In this analysis, the overlap is kept fixed at 50%, while the window duration is varied. As reported in Table 6, shortening the window to 0.640 s reduces the available temporal context and degrades accuracy and macro-F1. Increasing the window to 2.560 s recovers most of the performance, but does not improve over the selected 1.280 s configuration and would increase the number of tokens processed per inference. This has direct implications for latency and memory, since the attention path and activation buffers scale with the temporal length. The adopted 1.280 s window with 0.640 s stride therefore provides the best trade-off among the tested settings: it achieves the highest recognition quality while keeping the token count fixed to $T = 64$ and preserving a stride-driven real-time deadline compatible with the deployed hardware.

Table 6. Sensitivity analysis with respect to window length. The stride is set to 50% of the window duration in all cases.

Window [s]	Stride [s]	Timesteps T	Accuracy	Macro-F1
0.640	0.320	32	0.755	0.758
1.280	0.640	64	0.802	0.804
2.560	1.280	128	0.796	0.798

Table 7 complements the main design sweep with targeted ablations around the selected configuration. These results clarify why the deployed model combines local attention with window size 16, attention pooling, positional mixing, and the integer-affine LayerNorm path. Global attention and floating-point LayerNorm improve accuracy,

but they relax the deployment assumptions that motivate the proposed design. Conversely, shrinking or enlarging the local attention window, replacing attention pooling with average pooling, or removing positional mixing degrades the overall trade-off. Taken together, Tables 5–7 show that the final STORM instance is not the best isolated accuracy point under unconstrained settings, but the most balanced operating point among the explored solutions once temporal context, model size, quantization compatibility, latency, and deployment-oriented cost are considered.

Table 8 compares the selected STORM model with lightweight baselines at comparable parameter counts. Under the same protocol, STORM provides the strongest recognition performance, while the alternative model families expose different trade-offs between predictive quality and software-side latency. This comparison supports the choice of a compact Transformer in the present setting, without implying that Transformers are universally preferable for embedded HAR.

Table 7. Targeted ablations around the selected STORM configuration. The selected model uses local attention with window size 16, positional mixing, attention pooling, and integer-affine LayerNorm. The deployment-oriented STORM version is indicated in bold.

Variant	Params	Acc.	F1
STORM	19,753	0.802	0.804
Global attention	19,753	0.821	0.833
Local attention, window 32	19,753	0.775	0.783
Local attention, window 8	19,753	0.758	0.764
Floating-point LayerNorm	19,753	0.817	0.823
Average pooling	18,664	0.725	0.727
Positional mixing off	19,625	0.753	0.760

Table 8. Comparison with lightweight baselines under the same cross-source protocol. The deployment-oriented STORM version is indicated in bold.

Model	Params	Acc.	Macro-F1
STORM	19,753	0.802	0.804
1D-CNN	19,064	0.747	0.753
LSTM	19,609	0.760	0.768
GRU	19,527	0.740	0.754
Hybrid	19,185	0.733	0.749

Finally, Table 9 reports the effect of the quantized deployment path. The small gap between the FP32 reference and the exported INT8 inference indicates that the training, export, and integer execution paths remain well aligned. Figure 2 shows that residual errors are mainly concentrated among semantically related classes. The strongest confusion occurs between *standing* and *sitting*, which is expected because both are low-motion postural activities and may differ mainly in sensor orientation or placement. Locomotion classes also show some overlap: *walking* is partly confused with *upstairs*, while *downstairs* is confused with *walking*, *upstairs*, and *other*. This suggests that short windows can make vertical-motion activities harder to separate. The *other* class is also involved in several errors, consistently with its heterogeneous composition. Conversely, *running* and *lying* are the most stable classes, reflecting their distinctive inertial signatures. Overall, the confusion matrix indicates that the remaining errors are mainly due to class ambiguity and cross-source variability rather than to quantization.

Table 9. Impact of quantization on the cross-source benchmark for the selected STORM configuration.

Model	Accuracy	Macro-F1
STORM, FP32 reference	0.802	0.804
STORM, INT8 deployment	0.799	0.801

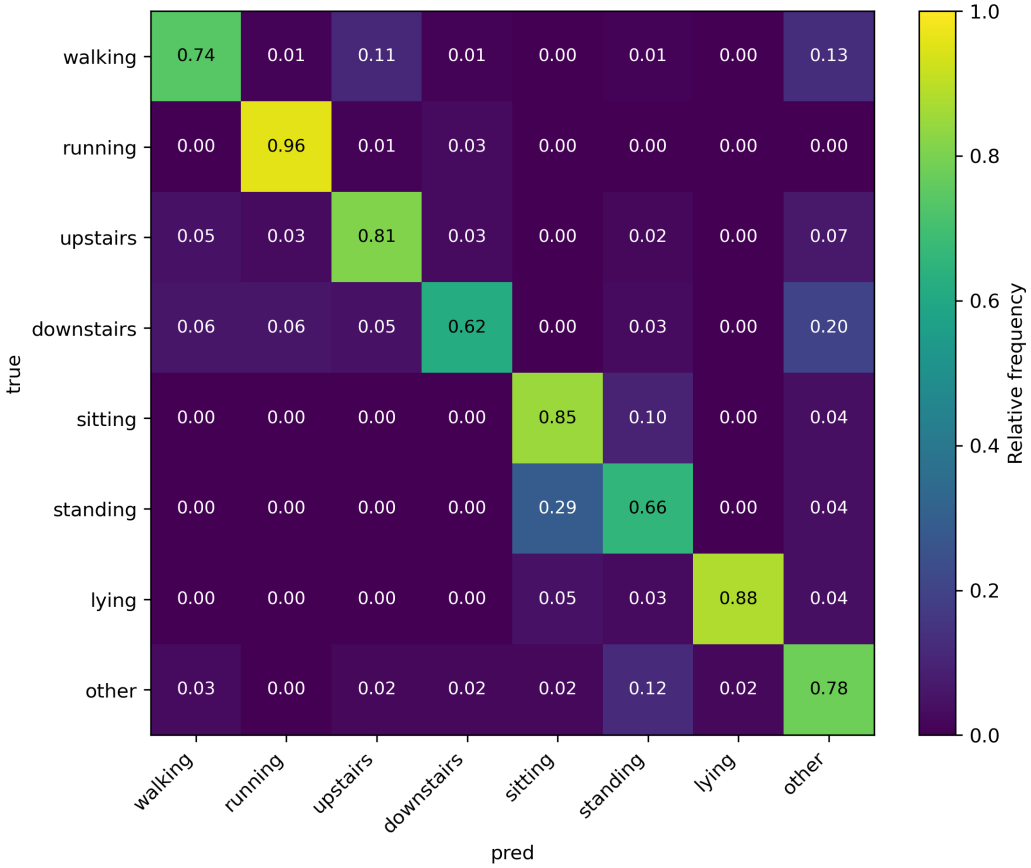

Figure 2. Row-normalized confusion matrix for the INT8 deployment model on the cross-source benchmark.

4.2. Effect of Sensor Configurations and Failures

Tables 3 and 10 summarize the impact of sensor subsets and controlled test-time failures. On the cross-source benchmark, the nominal 6-channel Acc+Gyro setting is the clear best-performing configuration, while accelerometer-only variants incur a substantial degradation, indicating that gyroscope dynamics provide complementary cues under cross-subject and cross-source heterogeneity. In contrast, performance on the original single-dataset evaluations remains markedly higher, confirming that the cross-source protocol intentionally stresses generalization across acquisition conditions rather than optimizing for closed-dataset accuracy. The PAMAP2 reference experiment exhibits a wider gap between accuracy and macro-F1 than the other datasets. This indicates that the aggregate score is supported by strong performance on high-support classes, while one or more low-support classes remain harder to classify. In this setting, macro-F1 is the more informative summary because it weights all classes equally and therefore better reflects class-wise balance.

For completeness, removing the *other* class from the cross-source label space increases performance to 0.850/0.824 accuracy/macro-F1. This gain is expected because the task becomes a stricter closed-set classification problem. Nevertheless, we retain the 8-class setting as the main benchmark because, in deployment, the node must avoid systematically

relabeling activities outside the target set as one of the monitored classes; in this sense, the *other* class acts as a practical background/rejection category.

Table 10. Cross-source 6-channel robustness under selected sensor failures (test-time injection).

Failure Test	Accuracy	Macro-F1
Modality outages		
Gyro total failure (gyro off)	0.603	0.615
Acc total failure (acc off)	0.471	0.396
All sensors failure	0.173	0.037
Single-axis outages (examples)		
Acc axis failure (X)	0.686	0.675
Acc axis failure (Z)	0.597	0.576
Gyro axis failure (Y)	0.634	0.657
Gyro axis failure (Z)	0.756	0.772
Noise/intermittency/drift (examples)		
Gyro noise $\sigma = 0.5$	0.478	0.491
Acc noise $\sigma = 0.5$	0.314	0.313
Gyro intermittent dropout 30%	0.716	0.726
Acc intermittent dropout 30%	0.487	0.475
Gyro stuck (freeze)	0.636	0.643
Acc stuck (freeze)	0.714	0.722
Acc scale drift 2.0×	0.478	0.494
Gyro scale drift 2.0×	0.705	0.692

The failure injections in Table 10 expose a consistent modality asymmetry that is actionable for co-design. Disabling the gyroscope largely recovers the accelerometer-only regime, whereas removing the accelerometer causes a much sharper collapse, identifying the accelerometer as the safety-critical modality and the gyroscope as a secondary but valuable contributor. The same qualitative trend holds across realistic non-ideal behaviors, including additive noise, intermittent dropout, freeze/stuck faults, and scale drift: perturbations affecting acceleration measurements typically induce larger accuracy and macro-F1 losses than comparable gyroscope perturbations, while partial dropouts are tolerated better than sustained corruption.

Practically, this suggests prioritizing accelerometer reliability (mechanical mounting, calibration, and redundancy when feasible) and treating the gyroscope as a performance enhancer rather than a single point of safety. It also motivates training-time fault augmentation aligned with deployed normalization and low-overhead runtime checks to detect intermittency, stuck sensors, and drift before they silently degrade HAR quality.

4.3. FPGA Implementation and Resource Utilization

The complete X-HEEP system integrating the cv32e40p core with FPU and DSP extensions, together with the streaming CGRA, was synthesized and implemented on an AMD ZCU104 FPGA development board as a prototyping platform. The purpose of this implementation is to validate the complete hardware–software stack and to measure cycle count, latency, throughput, and integration overheads of the deployed STORM inference pipeline in a realistic execution environment. The FPGA platform is thus used for functional and timing validation, while energy is analyzed separately through activity-based power estimation on a 65 nm technology target. Resource utilization on the FPGA is driven primarily by logic and on-chip memory, whereas DSP usage remains limited. To contextualize

the hardware cost of the prototype, Table 11 reports the post-route resource utilization of the full configuration.

Table 11. Post-route FPGA resource utilization on the AMD ZCU104 board (Zynq UltraScale+ XCZU7EV).

Item	X-HEEP + CGRA
System clock	100 MHz
LUTs	43.62%
FFs	8.94%
DSPs	3.30%
BRAMs	25.64%
I/O buffers	13.06%

4.4. On-Board Latency and Energy Breakdown

Using the methodology described in Section 3.7, we evaluate the deployed STORM configuration on the complete system by combining cycle counting on FPGA with activity-based full-system power analysis. The reported results refer to steady-state inference and include software orchestration, DMA exchanges with the CGRA, and on-chip data movement. Since all model parameters remain resident in on-chip SRAM during inference, the reported figures exclude external weight-transfer overhead and therefore capture the steady-state execution cost of continuous HAR.

The optimized implementation completes one inference in 6,739,790 cycles. At 100 MHz, this corresponds to a latency of 67.4 ms and a throughput of 14.84 inferences/s. Since the application operates on 1.280 s windows with 0.640 s stride, real-time feasibility is dictated by the stride deadline rather than by the full window duration. The minimum clock frequency required to sustain continuous operation is therefore 10.5 MHz, leaving a comfortable timing margin at the measured operating point. Table 12 summarizes the corresponding end-to-end latency, throughput, and effective MMAC-eq/s. Under the same operating conditions, the average full-system power during inference is 9.38 mW, which yields a total inference energy of 632.4 µJ. The corresponding workload is 6.125 MMAC-eq per inference, resulting in an energy cost of 0.103 nJ/MAC-eq.

Table 12. End-to-end latency, throughput, and effective MMAC/s.

Working Freq.	Deadline/Latency [s]	Throughput [inf/s]	Eff. [MMAC-eq/s]
Real-time bound	0.640	1.56	9.57
100 MHz	0.067	14.84	90.88

Table 13 reports the joint kernel-level breakdown in terms of cycles and energy, where repeated kernels from the two encoder blocks are aggregated by kernel type to keep the presentation compact. The dominant single contributor is the MHSA QKV projection, which alone accounts for 23.86% of total cycles and 20.43% of total inference energy. When combined with score computation, softmax, context accumulation, output projection, and the LayerNorm kernels inside the attention blocks, the MHSA path reaches 59.75% of total execution time and 56.61% of total inference energy. LayerNorm remains the main secondary contributor, with all normalization kernels together accounting for 25.28% of total cycles and 24.75% of total energy.

By contrast, the kernels offloaded to the CGRA, namely SiLU, GELU, and softmax, contribute only 0.56% of total cycles and about 1.08% of total inference energy. This indicates that the current mapping is effective at suppressing the latency and energy impact of transcendental and control-heavy stages, and that the dominant optimization opportunity

has shifted toward the integer attention datapath itself, in particular QKV generation, context accumulation, and the associated normalization and data-reordering costs.

To quantify the benefit of CGRA acceleration, we also evaluated an otherwise identical execution path in which the nonlinear kernels are executed without CGRA offload. In this configuration, one inference requires 7,154,337 cycles and an estimated total energy of 661 µJ. In comparison, the streaming CGRA acceleration saves 414,547 cycles, corresponding to about 4.15 ms at 100 MHz, and reduces the inference energy by approximately 28.6 µJ. This confirms that offloading SiLU, GELU, and softmax lowers the end-to-end execution cost while preserving the same model and deployment pipeline. The remaining cost is mainly associated with the attention and normalization path, indicating where future hardware mapping efforts should be focused.

Table 13. Joint kernel-level latency and energy breakdown for one STORM inference in the optimized Xpulp + CGRA configuration. Repeated kernels from the two encoder blocks are aggregated by kernel type.

Kernel	Cycles	Share [%]	Energy [µJ]	Share [%]
ConvStem_conv	285,760	4.24	25.2	3.98
ConvStem_SiLU	4026	0.06	0.836	0.13
Positional_Mix	133,120	1.98	10.7	1.69
MHSA_LN	559,368	8.30	44.2	6.99
MHSA_QKV	1,608,288	23.86	129.2	20.43
MHSA_SCORE	528,384	7.84	42.6	6.74
MHSA_SOFTMAX	20,680	0.31	3.82	0.60
MHSA_CTX	773,888	11.48	83.6	13.22
MHSA_OUTPROJ	536,256	7.96	54.6	8.63
MLP_LN	555,948	8.25	55.8	8.82
MLP_FC1	493,440	7.32	55.0	8.70
MLP_GELU	5702	0.08	1.33	0.21
MLP_FC2	492,800	7.31	52.4	8.29
Final_LN	278,951	4.14	26.8	4.24
AttnPool_LN	278,328	4.13	26.7	4.22
AttnPool_FC1	127,168	1.89	13.5	2.13
AttnPool_GELU	1860	0.03	0.368	0.06
AttnPool_FC2	11,264	0.17	1.28	0.20
AttnPool_SOFTMAX	5159	0.08	0.457	0.07
AttnPool_SUM	7885	0.12	0.952	0.15
Classifier_LN	30,946	0.46	3.00	0.47
Classifier_FC	569	0.01	0.060	0.01
Total	6,739,790	100.00	632.4	100.00

As a more realistic system-level estimate, we also account for representative always-on sensing and background overheads. Assuming an always-on sensor power budget of 1.00 mW, an additional idle/background budget of 0.25 mW, and regulator efficiency 0.90, the average compute-related power over one stride is 0.99 mW, while the corresponding average full-system power rises to 2.24 mW. For a 150 mAh Li-ion/LiPo cell at 3.7 V, this results in an estimated continuous operating lifetime of 9.3 days at a stride of 0.640 s. This figure should be interpreted as an illustrative system-level estimate rather than a device-specific measurement, since the actual lifetime will depend on the selected IMU, its operating mode and output data rate, the interrupt and buffering policy, firmware duty cycling, memory-access patterns outside the measured inference interval, and the effectiveness of system-level power management.

5. Positioning of STORM in the Embedded HAR Design Space

Direct numerical comparison with prior work must be interpreted with care because embedded HAR studies differ in sensing modality, class taxonomy, preprocessing, windowing policy, subject split protocol, quantization scheme, hardware target, and measurement boundary. STORM is therefore positioned primarily by the deployment setting it addresses: a microcontroller-scale Transformer, an integer-oriented export path, a heterogeneous low-power execution target, and a cross-source evaluation protocol that explicitly includes distribution shift and sensing perturbations.

At the model level, previous Transformer-based HAR studies have shown that self-attention can be effective for inertial time series and that compact encoder architectures can achieve strong recognition performance [2–4]. These works mainly advance the algorithmic side of the problem under conventional dataset-specific evaluation settings. STORM belongs to the same broad family of attention-based models, but it is developed under a different objective: preserving usefulness after quantization, C export, and execution on a resource-constrained embedded platform, rather than maximizing closed-dataset accuracy alone.

At the same time, the Transformer choice should not be interpreted as an a priori rejection of CNNs or recurrent models. In TinyML-oriented HAR, compact CNNs remain particularly strong because they map efficiently to streaming local operations, have favorable memory behavior, and often avoid the intermediate-buffer pressure associated with attention and residual paths. This is also reflected by MCU-focused studies showing that lightweight CNNs and recurrent networks can outperform or be more practical than Transformer variants under strict CPU-only constraints [5,6]. STORM is positioned with this limitation in mind: it does not claim that Transformers are universally preferable, but identifies a narrower operating regime in which a tiny attention-based model can be competitive when architecture, quantization, memory layout, and hardware mapping are co-designed. This is why the experimental evaluation includes matched-footprint CNN, recurrent, and hybrid baselines under the same cross-source protocol, rather than treating the Transformer architecture as self-justifying.

The most relevant comparison is with recent works that target deployment of small Transformer models on embedded hardware. Jung et al. focus on kernel- and schedule-level optimization of encoder-only Transformers for low-power MCUs [7], while Ling et al. address automated FPGA-oriented deployment through hardware-aware search, quantization-aware training, and RTL generation [19]. STORM differs from both in emphasis. Its main contribution is not automation or a generic optimization framework, but a constrained end-to-end co-design in which dataset protocol, model structure, quantization path, memory layout, and heterogeneous mapping are aligned to a specific low-power RISC-V platform. This makes it possible to characterize not only whether deployment is feasible, but also which operators remain dominant once nonlinear stages are offloaded.

The evaluation setting further differentiates STORM from much of the existing literature. Many prior works are assessed on single datasets, fixed sensor placements, and nominal sensing conditions, which is appropriate when the target is architectural validation or kernel efficiency. By contrast, STORM is evaluated under cross-source harmonization, subject-independent splits, and controlled sensor-failure scenarios. In this sense, the contribution is methodological as much as architectural: the benchmark and protocol are intentionally chosen to expose the interaction between dataset shift, sensor degradation, quantization, and hardware-aware model design.

A complementary HAR direction relies on WiFi channel state information (CSI), where activities are inferred from human-induced perturbations of the wireless channel rather than from body-worn inertial signals. CSI-based systems enable device-free sensing and

can reuse commodity WiFi infrastructure, reducing user burden and potentially lowering deployment cost in instrumented indoor environments [24]. However, they target a different deployment regime from STORM: their performance depends on room layout, transmitter–receiver geometry, multipath conditions, interference, and the number and position of occupants. Inertial HAR requires one wearable node per user, but provides body-centric measurements that are independent of the surrounding radio infrastructure and naturally compatible with on-node processing. From a privacy perspective, both approaches avoid camera-based monitoring: CSI avoids visual sensing but still performs ambient monitoring of a space, whereas the inertial setting considered here keeps raw motion data local to the wearable node. Scalability also differs: WiFi CSI can cover a monitored area without instrumenting each subject, but may require denser infrastructure or recalibration across rooms and occupants; wearable inertial HAR scales with users at the cost of per-node battery, memory, and latency constraints. Therefore, CSI-based HAR and the present inertial approach should be viewed as complementary rather than interchangeable. Table 14 summarizes the positioning of STORM against representative accuracy-focused, MCU-level, and FPGA-level Tiny Transformer deployment works.

Table 14. Positioning of STORM within the Tiny Transformer deployment design space. Energy †: activity-based simulation (65 nm). n/r = not reported.

Model	Dataset	Acc.	Params	Freq.	Lat.	Energy	Platform
Accuracy-focused (application processor, no MCU deployment)							
Lamaakal et al. [4]	UTD-MHAD	98.7%	0.62 M	1.5 GHz	3.1 ms	1.8 mJ	RPi 4B
	MM-Fit	98.6%	0.62 M	1.5 GHz	3.0 ms	1.8 mJ	RPi 4B
1D-CNN on single-core RISC-V MCU (same ISA class as STORM)							
Daghero et al. [5]	UniMiB-SHAR	86.2%	∼23.2 kB	205.1 MHz	16.2 ms	61.59 µJ	PULPissimo (1×RV32+Xpulp)
	UCI HAPT	85.6%	∼7.5 kB		6.11 ms	23.27 µJ	
	WISDM	98.8%	∼6.2 kB		4.19 ms	15.94 µJ	
	WALK	95.7%	∼1.7 kB		0.18 ms	0.67 µJ	
Tiny Transformer on multi-core RISC-V MCU (different domains)							
Jung [7]	EEG	n/r	n/r	370 MHz	9.42 ms	460 µJ	GAP9 (9×RV32)
	Radar	n/r	n/r	370 MHz	5.49 ms	315 µJ	
	ECG	n/r	n/r	370 MHz	2.85 ms	120 µJ	
Busia et al. [25]	MIT-BIH	99.0%	∼6.0k	370 MHz	4.28 ms	90 µJ	GAP9 (9×RV32)
Tiny Transformer on embedded FPGA (dedicated accelerator)							
Ling et al. [19]	WISDM	83.9%	∼20.1k	100 MHz	12.04 ms	855 µJ	Spartan-7 XC7S15
	UCIHAR	82.4%	∼1.0k		1.034 ms	67 µJ	
Tiny Transformer on constrained MCU (transfer costs included)							
Lattanzi et al. [6]	WISDM	73.4%	n/r	240 MHz	≤120.0 ms	n/r	ESP32 (2×Xtensa)
	RealWorld	85.6%	n/r				
	Watch_HAR	81.1%	n/r				
This work							
STORM	Cross-source	79.9%	∼19.7k	100 MHz	67.4 ms	632.4 µJ †	X-HEEP+CGRA (1×RV32+Xpulp)

Overall, STORM occupies a distinct point in the embedded HAR design space. Prior HAR Transformer studies clarify the modeling potential of attention, while prior TinyML studies clarify why CNNs and recurrent models remain highly competitive on constrained MCUs. STORM contributes at their intersection by showing that a compact Transformer can remain practical under a specific co-designed stack: cross-source evaluation,

integer-oriented deployment, heterogeneous RISC-V + CGRA execution, and hardware-level profiling. The resulting analysis also shows that, after nonlinearity offload, multi-head self-attention remains the dominant residual bottleneck and the main target for future optimization.

6. Conclusions and Limitations

This work presented STORM, a deployment-oriented tiny 1D Transformer for inertial Human Activity Recognition co-designed with the X-HEEP RISC-V single-core SoC and the streaming CGRA. We evaluated the full stack end-to-end under a deployment-aligned protocol: a cross-source 8-class benchmark built from 3 public accelerometer+gyroscope datasets with source-stratified, subject-independent splits and continuous windowing with 1.280 s duration and 0.640 s stride, complemented by single-dataset variants with richer sensing modalities and controlled sensor-failure models.

Under this deliberately stringent setting, STORM reaches 0.802/0.804 accuracy/macro-F1 in FP32 on the cross-source benchmark with a microcontroller-class footprint. With QAT and an exported INT8 C inference pipeline using an integer-affine LayerNorm, performance remains close to the FP32 reference (0.799/0.801), showing that the deployment-oriented quantized path preserves most of the original model quality. The sensor study further indicates that robustness is a first-order deployment variable, with clear benefits from multi-sensor configurations and non-negligible sensitivity to realistic sensing faults.

On the deployed X-HEEP+CGRA platform, STORM completes one inference in 6,739,790 cycles, corresponding to 67.4 ms at 100 MHz, and therefore satisfies the continuous-inference deadline imposed by a 0.640 s stride. The cycle breakdown shows that nonlinear offload is effective, while MHSA remains the dominant residual cost in the present mapping. This reflects a structural trade-off of the selected operating point: SRAM constraints keep the model compact, but also limit the extent to which attention costs can be amortized. Fused attention kernels and dedicated CGRA mappings for the MHSA path therefore appear as the most promising directions for further optimization.

The present results are specific to inertial HAR and to the selected hardware-software stack, so they should not be interpreted as directly transferable to arbitrary time-series problems or final silicon implementations. At the same time, the broader methodology proposed here, namely cross-source harmonization, deployment-oriented model design, quantization-aware training, integer export, and hardware-level profiling, is applicable beyond HAR to embedded time-series tasks with similar robustness and resource constraints. Likewise, the FPGA prototype provides a useful validation of timing, integration, and bottleneck structure, but it does not replace ASIC- or device-level characterization.

A further limitation is the fixed-window continuous-inference assumption adopted in this work. This choice makes latency and energy constraints explicit and reproducible, but it does not cover event-driven regimes in which computation is triggered only when signal conditions justify it. In that direction, event-driven deployments and spike-based sequence models, including spike-driven Transformer variants [26], may offer a useful path toward lower average power in always-on settings. Scalability is another relevant constraint: increasing the number of classes or the sampling rate would place additional pressure on activation memory, attention cost, and energy per decision, with the MHSA path expected to remain the main bottleneck.

Finally, the current energy analysis is based on activity-driven estimation under controlled assumptions. A natural extension is to combine direct hardware measurements with more detailed physical modeling, including hybrid FEM-AI approaches [27] for electrothermal analysis, to refine future energy estimates under sustained execution. Future work will therefore focus on extending the CGRA mapping to the computational core of

MHSA, validating the energy model on real hardware, and exploring more scalable or event-driven deployment regimes under tighter long-term energy constraints.

Author Contributions: Conceptualization, A.V.; methodology, A.V.; software, A.V.; validation, A.V.; formal analysis, A.V.; investigation, A.V.; data curation, A.V.; writing—original draft preparation, A.V.; visualization, A.V.; supervision, C.G., A.M. and M.V.; writing—review and editing, M.V. All authors have read and agreed to the published version of the manuscript.

Funding: This publication is part of the project PNRR-NGEU which has received funding from the MUR—DM 117/2023.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets analyzed in this study are publicly available from the original sources: MotionSense [21], UCI HAR [22], and PAMAP2 [23]. The code used for benchmark generation, preprocessing, model training, quantization-aware training, export, and deployment is available at https://github.com/AlessandroVaraldi/storm (accessed on 29 April 2026). No new proprietary datasets were generated in this study.

Acknowledgments: The authors would like to thank Alessio Naclerio for his support with the CGRA-based acceleration flow and hardware integration. The authors also thank Maurizio Martina and his research group for kindly providing access to the AMD ZCU104 board. The authors acknowledge the use of ChatGPT 5.4, a generative AI tool developed by OpenAI, as a writing-support assistant for language editing, grammar correction, and improvement of textual clarity. The authors reviewed and edited all AI-assisted text and take full responsibility for the content of the manuscript.

Conflicts of Interest: Authors Claudio Genta and Alberto Manzone are employed by Ideas & Motion S.r.l., Italy. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HAR	Human Activity Recognition
STORM	Small Transformer for On-node Recognition of Motion
IMU	Inertial Measurement Unit
TinyML	Tiny Machine Learning
MCU	Microcontroller Unit
SoC	System-on-Chip
CGRA	Coarse-Grained Reconfigurable Array
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit
MHSA	Multi-Head Self-Attention
MLP	Multi-Layer Perceptron
QAT	Quantization-Aware Training
LUT	Lookup Table
DMA	Direct Memory Access
SRAM	Static Random-Access Memory
QSPI	Quad Serial Peripheral Interface
FPGA	Field-Programmable Gate Array
ASIC	Application-Specific Integrated Circuit

References

1. Ramanujam, E.; Perumal, T.; Padmavathi, S. Human Activity Recognition With Smartphone and Wearable Sensors Using Deep Learning Techniques: A Review. *IEEE Sens. J.* **2021**, *21*, 13029–13040. https://doi.org/10.1109/JSEN.2021.3069927.
2. Dirgová Luptáková, I.; Kubovčík, M.; Pospíchal, J. Wearable Sensor-Based Human Activity Recognition with Transformer Model. *Sensors* **2022**, *22*, 1911. https://doi.org/10.3390/s22051911.
3. Buffelli, D.; Vandin, F. Attention-Based Deep Learning Framework for Human Activity Recognition With User Adaptation. *IEEE Sens. J.* **2021**, *21*, 13474–13483. https://doi.org/10.1109/JSEN.2021.3067690.
4. Lamaakal, I.; Yahyati, C.; Maleh, Y.; El Makkaoui, K.; et al. XTinyHAR: A Tiny Inertial Transformer for Human Activity Recognition via Multimodal Knowledge Distillation and Explainable AI. *Sci. Rep.* **2025**, *15*, 42335. https://doi.org/10.1038/s41598-025-26297-2.
5. Daghero, F.; Burrello, A.; Xie, C.; Castellano, M.; Gandolfi, L.; Calimera, A.; Macii, E.; Poncino, M.; Pagliari, D.J. Human Activity Recognition on Microcontrollers with Quantized and Adaptive Deep Neural Networks. *ACM Trans. Embed. Comput. Syst.* **2022**, *21*, 46. https://doi.org/10.1145/3542819.
6. Lattanzi, E.; Calisti, L.; Contoli, C. Are Transformers a Useful Tool for Tiny Devices in Human Activity Recognition? In Proceedings of the 8th International Conference on Advances in Artificial Intelligence (ICAAI), London, UK, 17–19 October 2024; pp. 339–344. https://doi.org/10.1145/3704137.3704171.
7. Jung, V.J.B.; Burrello, A.; Scherer, M.; Conti, F.; Benini, L. Optimizing the Deployment of Tiny Transformers on Low-Power MCUs. *IEEE Trans. Comput.* **2025**, *74*, 526–541. https://doi.org/10.1109/TC.2024.3500360.
8. Wijerathne, D.; Li, Z.; Karunarathne, M.; Pathania, A.; Mitra, T. CASCADE: High Throughput Data Streaming via Decoupled Access-Execute CGRA. *ACM Trans. Embed. Comput. Syst.* **2019**, *18*, 50:1–50:26. https://doi.org/10.1145/3358177.
9. Machetti, S.; Schiavone, P.D.; Ansaloni, G.; Peón-Quirós, M.; Atienza, D. X-HEEP: An Open-Source, Configurable and Extendible RISC-V Platform for TinyAI Applications. In Proceedings of the 2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Kalamata, Greece, 6–9 July 2025; pp. 1–6. https://doi.org/10.1109/ISVLSI65124.2025.11130281.
10. Naclerio, A.; Riente, F.; Turvani, G.; Vacca, M.; Zamboni, M.; Graziano, M. Mage: A Decoupled Access-Execute CGRA Tailored for Static Control Applications. In Proceedings of the 2025 IEEE International Symposium on Circuits and Systems (ISCAS), London, UK, 25–28 May 2025; pp. 1–5. https://doi.org/10.1109/ISCAS56072.2025.11043426.
11. Hammerla, N.Y.; Halloran, S.; Ploetz, T. Deep, Convolutional, and Recurrent Models for Human Activity Recognition using Wearables. *arXiv* **2016**, https://doi.org/10.48550/arXiv.1604.08880.
12. Ordóñez, F.J.; Roggen, D. Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition. *Sensors* **2016**, *16*, 115. https://doi.org/10.3390/s16010115.
13. Nweke, H.F.; Teh, Y.W.; Al-garadi, M.A.; Alo, U.R. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Syst. Appl.* **2018**, *105*, 233–261. https://doi.org/10.1016/j.eswa.2018.03.056.
14. Chen, K.; Zhang, D.; Yao, L.; Guo, B.; Yu, Z.; Liu, Y. Deep Learning for Sensor-based Human Activity Recognition: Overview, Challenges, and Opportunities. *ACM Comput. Surv.* **2021**, *54*, 77. https://doi.org/10.1145/3447744.
15. Zhang, S.; Li, Y.; Zhang, S.; Shahabi, F.; Xia, S.; Deng, Y.; Alshurafa, N. Deep Learning in Human Activity Recognition with Wearable Sensors: A Review on Advances. *Sensors* **2022**, *22*, 1476. https://doi.org/10.3390/s22041476.
16. Abedin, A.; Ehsanpour, M.; Shi, Q.; Rezatofighi, H.; Ranasinghe, D.C. Attend and Discriminate: Beyond the State-of-the-Art for Human Activity Recognition Using Wearable Sensors. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **2021**, *5*, 1. https://doi.org/10.1145/3448083.
17. Gao, W.; Zhang, L.; Teng, Q.; He, J.; Wu, H. DanHAR: Dual Attention Network for Multimodal Human Activity Recognition Using Wearable Sensors. *Appl. Soft Comput.* **2021**, *111*, 107728. https://doi.org/10.1016/j.asoc.2021.107728.
18. Ravì, D.; Wong, C.; Lo, B.; Yang, G.Z. Deep Learning for Human Activity Recognition: A Resource Efficient Implementation on Low-Power Devices. In Proceedings of the 2016 IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN), San Francisco, CA, USA, 14–17 June 2016; pp. 71–76. https://doi.org/10.1109/BSN.2016.7516235.
19. Ling, T.; Qian, C.; Haßler, L.J.; Schiele, G. Automating Versatile Time-Series Analysis with Tiny Transformers on Embedded FPGAs. In Proceedings of the 2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Kalamata, Greece, 6–9 July 2025; Volume 1, pp. 1–6. https://doi.org/10.1109/ISVLSI65124.2025.11130202.
20. Varaldi, A.; Naclerio, A.; Riente, F.; Zamboni, M.; Graziano, M.; Vacca, M. Optimizing TCN Inference: A Hardware-Software Co-Design Approach with CGRA Acceleration. In Proceedings of the 2025 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Kalamata, Greece, 6–9 July 2025; pp. 1–6. https://doi.org/10.1109/ISVLSI65124.2025.11130330.
21. Malekzadeh, M.; Clegg, R.G.; Cavallaro, A.; Haddadi, H. Mobile Sensor Data Anonymization. In *Proceedings of the International Conference on Internet of Things Design and Implementation, IoTDI '19*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 49–58. https://doi.org/10.1145/3302505.3310068.

22. Anguita, D.; Ghio, A.; Oneto, L.; Parra, X.; Reyes-Ortiz, J.L. A Public Domain Dataset for Human Activity Recognition Using Smartphones. In Proceedings of the European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium, 24–26 April 2013.
23. Reiss, A.; Stricker, D. Introducing a New Benchmarked Dataset for Activity Monitoring. In Proceedings of the 2012 16th Annual International Symposium on Wearable Computers (ISWC), Newcastle, UK, 18–22 June 2012; pp. 108–109.
24. Boudlal, H.; Serrhini, M.; Tahiri, A. Human Activity Monitoring System with Commodity WiFi Infrastructure Using Channel State Information. *Indones. J. Electr. Eng. Comput. Sci.* **2023**, *31*, 763–776. https://doi.org/10.11591/ijeecs.v31.i2.pp763-776.
25. Busia, P.; Scrugli, M.A.; Jung, V.J.B.; Benini, L.; Meloni, P. A Tiny Transformer for Low-Power Arrhythmia Classification on Microcontrollers. *IEEE Trans. Biomed. Circuits Syst.* **2024**, *19*, 142–152. https://doi.org/10.1109/TBCAS.2024.3401858.
26. Yao, M.; Hu, J.; Hu, T.; Xu, Y.; Zhou, Z.; Tian, Y.; Xu, B.; Li, G. Spike-driven Transformer V2: Meta Spiking Neural Network Architecture Inspiring the Design of Next-generation Neuromorphic Chips. *arXiv* **2024**, https://doi.org/10.48550/arXiv.2404.03663.
27. Pratticò, D.; Carlo, D.D.; Silipo, G.; Laganà, F. Hybrid FEM-AI Approach for Thermographic Monitoring of Biomedical Electronic Devices. *Computers* **2025**, *14*, 344. https://doi.org/10.3390/computers14090344.