Dual-Stage Pruning for Human Activity Recognition on Microcontrollers

Quang Thong Phan[1] and Kristof Van Laerhoven[1]

University of Siegen, Siegen, Germany
quang.phan@student.uni-siegen.de, kvl@eti.uni-siegen.de

Abstract. Deploying neural networks for Human Activity Recognition on wearable devices is often constrained by battery capacity and computational power. While conventional compression techniques mitigate these constraints, they typically operate on raw time-domain data, which overlooks the low-frequency nature of human movement. Furthermore, as published inference latency are frequently tied to specific physical hardware setups and undocumented toolchains, validating it remains a significant barrier. To address these challenges, we propose a dual-stage pruning strategy optimized for frequency- and wavelet-domain HAR pipelines. Utilizing Gumbel-Softmax reparameterization, our methodology systematically eliminates uninformative transform bins at the input level before selectively pruning convolutional channels. By heavily compressing models, we leave vital memory headroom for other firmware components (OS, future updates, etc), and drastically reduce execution time to conserve battery life. To guarantee verifiable cloud-based hardware validation without requiring physical board access, we profile our optimized models using the ST Edge AI testbench. Our pruned models achieve over a 50% reduction in memory footprint and inference latency cut to less than a third, while maintaining predictive performance comparable to the baseline.

Keywords: human activity recognition · digital signal processing · embedded systems · cloud-based validation

1 Introduction

Human Activity Recognition (HAR) on wearable devices faces significant challenges due to the small battery and processing constraints of low-power microcontrollers (MCUs). To deploy neural network models within these strict limits, the field of Tiny Machine Learning (TinyML) heavily relies on compression techniques, such as weight quantization, weight clustering, and network pruning to minimize computational complexity and memory usage.

While these approaches effectively reduce model size, we may overlook the potential of optimizing for the HAR domain. The current HAR methods mostly use raw time-domain data, which is sensitive to hardware noise and ignores the fact that human movement is limited to specific frequency ranges. Shifting the

HAR pipelines to the frequency or wavelet domains allows for more effective pruning by isolating the essential motion information. In addition to being an understudied approach, the field faces an ongoing challenge regarding hardware validation. Research frequently reports impressive on-device latency reductions, but these claims are often tied to specific local hardware setups. Consequently, verifying published inference speedups can be difficult without obtaining the exact physical development board. Furthermore, while some unpruned models may technically fit within the theoretical constraints of a microcontroller, further compression is critical in real-world deployments. Minimizing RAM and ROM usage leaves vital headroom for the Real-Time Operating System (RTOS), wireless stacks, and firmware updates. Crucially, accelerating inference speed directly translates to reduced active-mode power consumption, allowing the MCU to return to sleep faster and significantly extending battery life.

In response to these challenges, we propose a domain-specific, dual-stage pruning strategy optimized for frequency- and wavelet-domain HAR. By leveraging Gumbel-Softmax reparameterization, our method eliminates uninformative transform bins at the input level, then selectively prunes intermediate convolutional network channels. To ensure our methodology can be reliably validated by the ubiquitous computing community, this paper presents three key contributions:

- We benchmark the Discrete Cosine Transform (DCT), Fast Fourier Transform (FFT) as frequency-domain representations, alongside the Integer Haar Wavelet (IHW) as a wavelet-domain representation, evaluating the trade-offs between these transformed inputs.
- Our pruned models achieve over a 50% reduction in memory footprint and a more than 3x acceleration in inference latency, while maintaining predictive performance comparable to the baseline.
- To facilitate verifiable cloud-based hardware validation, we profile our compressed models using low-power MCUs from the ST Edge AI public testbench [1]. Consequently, our efficiency metrics can be independently validated without requiring physical access to the target hardware.

2 Related Work

This section examines the hardware constraints that drive neural networks in wearable HAR, reviews current methodologies to highlight the gap in learnable pruning for frequency- and wavelet-domain inputs, and discusses how cloud-based benchmarking resolves the ongoing hardware reproducibility bottleneck.

System Constraints in Wearable HAR. Wearable sensing devices differ significantly from smartphones and servers, as they depend on low-power microcontrollers with severe memory and energy restrictions [29]. Operating at clock speeds up to 200 MHz, these platforms are typically limited to a few hundred kilobytes of SRAM and a few megabytes of Flash [27]. Furthermore, the

[1] ST Edge AI Cloud: https://stedgeai-dc.st.com [Last access June 2026]

use of small rechargeable batteries [28] requires thorough power management to support always-on monitoring capabilities. Therefore, the design of wearable systems requires a careful trade-off between energy usage, computational demand, and sensing fidelity [33,9].

To manage these strict hardware constraints without compromising performance, TinyML specializes in adapting neural networks for low-power microcontrollers. Previous research has investigated various model compression strategies, such as low-bit weight quantization [7,37] and lightweight architectures [38,4], to minimize computational complexity and memory footprints. Alongside these improvements, specialized inference engines like ExecuTorch [26] and TensorFlow Lite for Microcontrollers [8] facilitate practical deployment by supplying memory-efficient frameworks and integer-computational kernels tailored for edge devices.

Pruning Techniques and Methodologies. Pruning is essential for deploying neural networks on resource-constrained devices. It removes less essential components without significantly degrading performance [6]. Current literature generally explores two main structural paradigms. *Unstructured pruning* removes individual weights or connections, preserving the overall architecture and layer dimensions [30]. Typically, it zeroes out connections with the smallest absolute magnitudes [34]. This magnitude-based approach often serves as a preparatory step for further compression techniques, such as quantization [13,11]. *Structured pruning*, on the other hand, removes entire channels or filters to accelerate inference. Variation ranges from eliminating near-zero convolutional channels [16,24], dynamic channel selectivity [15], and pruning 2D FFT feature maps to preserve essential low-frequency movement energy [32].

Identifying these redundant components relies on either criteria-based or learnable methodologies. *Criteria-based methods* evaluate post-training parameter importance, often enhancing iterative magnitude pruning with scheduling functions like polynomial decay [19,1], though simple global magnitude pruning remains a highly competitive baseline [12]. *Learnable methods*, alternatively, integrate the search directly into training. Recent HAR approaches utilize input-dependent probabilities to adaptively fuse channels [20], differentiable channel-wise scaling combined with Neural Architecture Search [38], and sparse group Lasso mechanisms with attention [39].

Within this landscape of learnable methods, Gumbel-Softmax reparameterization stands out by enabling the optimization of discrete pruning decisions through continuous gradient descent. While this optimization strategy has been previously explored for channel pruning [14], its application remains largely confined to vision benchmarks (e.g., CIFAR-10, ImageNet) and heavy ResNet architectures. To bridge this gap between high-resource vision tasks and wearable sensing, our work significantly extends the Gumbel-Softmax paradigm into the HAR domain. Rather than merely adapting this paradigm to prune convolutional channels, we introduce a novel application at the input level: applying Gumbel masks to selectively drop uninformative frequency and wavelet bins. By coupling these input- and channel-level reductions, we formulate a dual-stage

Table 1. A comparison of state-of-the-art pruning techniques in HAR alongside our proposed method.

References	Pruning Setting	Pruning Method	Input Representation (Domain)
[35]	Unstructured (Model weights)	Learnable	Time
[13,30,11,34,1]	Unstructured (Model weights)	Criteria-Based	Time
[12,16,24,10]	Structured (Channels, Filters, Layers)	Criteria-Based	Time
[36,21,22,15,20,39,38]	Structured (Channels, Filters, Layers, Sensor Channels)	Learnable	Time
[32]	Structured (Feature Maps)	Criteria-based	Frequency
Our work	**Structured (Channels, Input Bins)**	**Learnable**	**Frequency/Wavelet**

pruning methodology specifically tailored for the strict constraints of low-power systems.

Cloud-based hardware validation. Standardized hardware validation remains a bottleneck in ubiquitous computing research. Specifically, metrics such as inference latency and memory utilization are fundamentally dictated by the target hardware architecture, specific compiler flags, and the availability of hardware-accelerated floating-point units [18]. Because these performance metrics are deeply coupled to the physical setup, merely publishing model source code and naming the microcontroller makes independent verification difficult unless researchers acquire the exact development board [27]. Consequently, cloud-based testbenches offer a practical way to decouple initial hardware validation from physical board possession.

3 Proposed Methods

This section introduces a dual-stage pruning methodology that leverages Gumbel-Softmax reparameterization to learn the optimal subset of input transform bins and intermediate convolutional channels for hardware-efficient HAR. Crucially, our dual-stage pipeline is trained sequentially: the model is first optimized at the input level (Stage 1) and retrained to convergence before any convolutional channel pruning (Stage 2) is initiated.

3.1 Input Bins Pruning

Human movements are typically concentrated within a narrow low-frequency band, with 99% of the energy in human gait falling below 15 Hz [3]. When raw signals are transformed into the frequency or wavelet domains, each transform bin isolates a specific frequency component. Therefore, we can treat them as independent discrete variables. To systematically evaluate and select the most informative of these variables, we employ the Gumbel-Softmax trick [17]. This reparameterization forms the foundation of our input pruning stage (Figure 1, Stage 1.1), where a learnable mask is introduced immediately prior to the input layer. Specifically, this mask is applied across the n input bins, modeling each

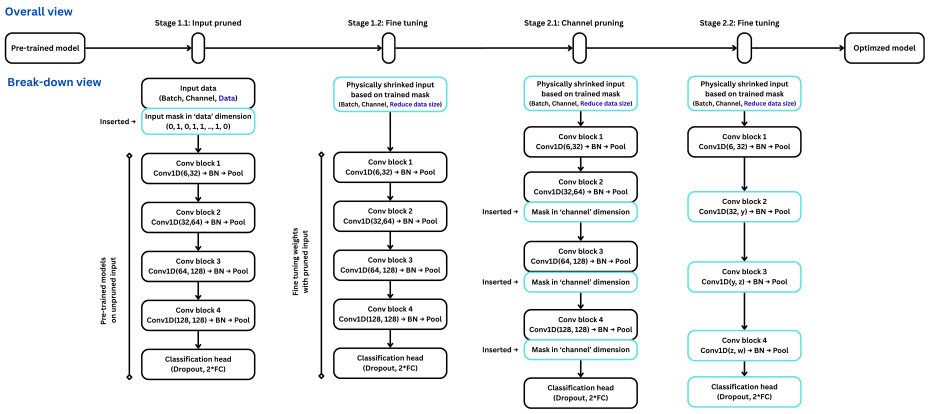

Fig. 1. Dual-stage pruning breakdown from pre-trained model to the optimized model for edge devices.

bin as a categorical variable with two possible states: active (1) or inactive (0). Consequently, the underlying learnable parameters form a weight matrix of shape $(n, 2)$, representing the logits for the active and inactive probabilities of each respective bin.

To discard irrelevant bins, we must encourage sparsity within this weight matrix. We achieve this by applying an L_1 penalty directly to the mask fraction $\mathcal{M}$, defined as the ratio of active bins to the total number of input bins n. The overall training loss function is thus formulated as:

$$\mathcal{L}_{total} = \mathcal{L}_{CE} + \lambda \cdot \mathcal{M} \tag{1}$$

where $\mathcal{L}_{CE}$ is the Cross-Entropy Loss and λ is a tunable sparsity coefficient. By increasing λ, the network aggressively penalizes active bins, trading predictive accuracy for a more compressed input size.

To maintain end-to-end differentiability while enforcing strict binary choices, the masking layer is trained using the Straight-Through Gumbel-Softmax Estimator [17]. First, stochastic Gumbel noise $g_{i,k}$ is injected into the unnormalized logits $\log(\pi_{i,k})$, enabling the continuous exploration of various masking combinations across training samples. These noised logits are then relaxed into continuous probabilities $y_{i,k}^{soft}$ via a softmax function modulated by a temperature parameter τ:

$$y_{i,k}^{soft} = \frac{\exp((\log(\pi_{i,k}) + g_{i,k})/\tau)}{\sum_{j=0}^{1} \exp((\log(\pi_{i,j}) + g_{i,j})/\tau)} \tag{2}$$

During the forward pass of the training phase, the model strictly enforces discrete states by applying an *argmax* operation to y^{soft}, yielding a hard binary mask y^{hard} (where each mask value is definitively 1 or 0). This ensures the input space is compressed. However, because the *argmax* operation is mathematically non-differentiable, the Estimator employs a straight-through trick during the

backward pass. It bypasses the discrete *argmax* step entirely, routing the gradients directly backward through the continuous y^{soft} probabilities. Note that the temperature parameter τ must be carefully selected: higher temperatures smooth the distribution to encourage stochastic exploration, but simultaneously attenuate the learning signal by making the soft approximation less representative of the hard forward pass. Afterwards, the zeroed input bins are then permanently discarded, and the network undergoes a retraining phase (Figure 4, Stage 1.2) utilizing a reduced learning rate (typically scaled by a factor of 0.1) to recover any minor accuracy drops.

In the inference phase, Gumbel noise is omitted, and probabilities are computed by applying a standard softmax directly to the learned logits. A final *argmax* operation is then applied to generate a static, binary mask. By multiplying this fixed mask with the original input, the network deterministically prunes uninformative bins, yielding a computationally efficient input representation optimized for edge hardware.

3.2 Convolutional Channels Pruning

Following these same principles above, the next stage executes channel pruning by introducing a learnable binary mask across the channel dimension of the feature maps (Figure 4, Stage 2.1 and 2.2). Modeled identically as categorical variables, the underlying mask weights form a matrix of shape $(c, 2)$, representing the unnormalized log-probabilities (logits) for the active (1) and inactive (0) states of each of the c channels. Note that rather than applying sparsity uniformly across the entire network, we employ the pruning mechanism exclusively to the network's intermediate layer. By explicitly preserving the initial feature extraction and final classification layers, this method yields significant advantages in the classification results.

4 Implementation

To validate our approach, the subsequent section describes the experimental methodology, including the selected benchmark datasets, the evaluation process and the reproducibility.

4.1 Datasets

To validate our dual-stage pruning approach, two datasets are being used. First, we leverage the WEAR dataset [5], which captures 50Hz tri-axial accelerometer signals from 24 subjects performing body-weight exercises. For this dataset, we group the 18 fine-grained activities into 8 distinct super-classes to emulate the constraints of wearable constraints. Second, we use the UCI-HAR dataset [2], which includes both 50Hz tri-axial gyroscope and accelerometer data from 30 participants performing 6 standard activities of daily living. Details are described in Table 2.

Table 2. Detailed configuration of each dataset utilized in this study. Note that the WEAR dataset's original 18 fine-grained activities are categorized into 8 fundamental activities.

Dataset	Activity	Classes	Channels	Window (s)	Overlap
WEAR	Sport	8	6 (Left+Right Acc)	2.00	50%
UCI-HAR	Daily living	6	6 (Acc+Gyro)	2.56	50%

4.2 Experimental Setup

We evaluate our proposed strategies using a typical lightweight Convolutional Neural Network tailored for inertial time-series data. By employing Depthwise Separable Convolutions, we decouple spatial and channel-wise feature extraction, greatly reducing model complexity. This yields a highly compact architecture comprising approximately 37k parameters, as illustrated in Figure 2.

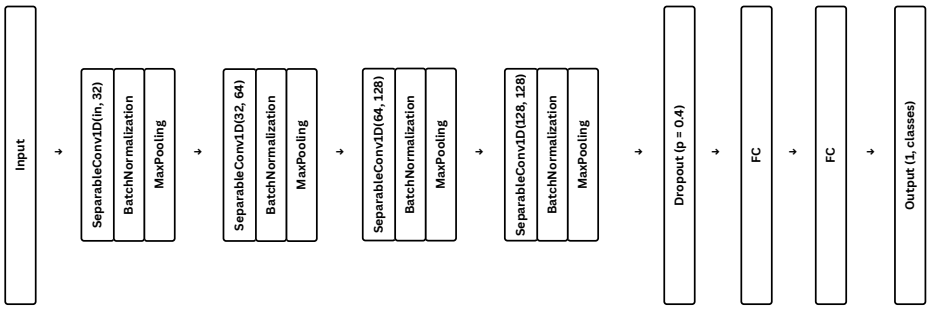

Fig. 2. Overview of the lightweight CNN architecture for evaluation.

Data preprocessing strategies are adapted based on the specific input domain. For time-domain (TD) configurations, the input signals will be fed directly into the network. Conversely, for frequency- and wavelet-domain configurations, the raw sensor data is transformed via FFT, DCT, and IHW, respectively. To preserve their spectral characteristics, these transformed representations are also fed directly into the network without further normalization.

During the training phase, the network is optimized using Adam with a learning rate of 1×10^{-3}, a weight decay of 1×10^{-6}, and a batch size of 64. This optimization process minimizes a weighted cross-entropy loss function. Additionally, the temperature parameter τ of the Gumbel-Softmax reparameterization is initialized at 10 and linearly decayed to 0.01 upon network convergence. Moreover, we evaluate the model's robustness against inter-subject variability by employing Leave-One-Subject-Out (LOSO) cross-validation.

Deploying the trained model into the embedded systems involves two primary phases: implementing the on-device data preprocessing pipeline, and converting

Table 3. Estimated CPU cycle costs and processing times for proposed preprocessing methods on an MCU running at 84MHz (Window size $N = 128$).

Preprocessing Technique	CPU cycle cost (1D sensor)	CPU cycle cost (of 6D sensor)	Processing time (of 6D sensor)
TD (Time Domain)	0 cycles	0 cycles	0 μs
FFT	16,153 cycles	96,918 cycles	1,153 μs
DCT	9,950 cycles	56,700 cycles	675 μs
IHW	710 cycles	4,260 cycles	51 μs

the model into a TFLite format. For the first phase, the computational cost of the preprocessing pipeline depends heavily on the chosen domain. Specifically, switching to the frequency or wavelet domain introduces additional preprocessing overhead, as illustrated in Figure 3. The estimated CPU cycle counts required to process a single 1D sensor channel for these transforms are detailed in 3. In practice, a system utilizing a 6D sensor configuration (e.g., a 3D accelerometer paired with a 3D gyroscope) will multiply these cycle costs by six. Despite this multiplication, the total estimated preprocessing time peaks at around 1 ms, making its impact on the system's inference latency negligible, as shown in Table 3.

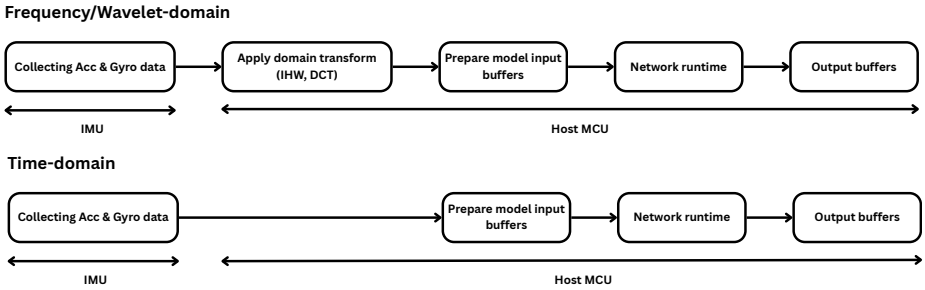

Fig. 3. The on-device processing pipelines for all domains. Notable is that the frequency or wavelet domain (top) introduces additional processing overhead.

4.3 Cloud-based Hardware Validation

To execute our cloud-based hardware validation, we leverage a pipeline that bridges local model development and physical hardware evaluation, as illustrated in Figure 4. First, the trained PyTorch models are converted to the TensorFlow utilizing `onnx2tf` and subjected to post-training quantization (8-bit integer weights and 16-bit integer activations). Second, to bypass the need for physical hardware access, these quantized models are programmatically submitted to the

publicly accessible ST Edge AI Cloud via its API. Third, the platform will generates the corresponding C-code firmware and compiles it into binary files. Lastly, the platform flashes the binary onto the target physical device for profiling.

For our benchmarking, we target an STM32F401 MCU operating at 84 MHz, equipped with a hardware Floating-Point Unit, 512 KB of flash memory, and 96 KB of SRAM. Once deployed on this specific MCU, the cloud testbench profiles the model and returns efficiency metrics. We systematically evaluate three key metrics without applying further hardware-specific memory optimizations: (1) inference latency, which measures the end-to-end processing time required for a single input sample; (2) ROM utilization, representing the non-volatile memory required to store static model parameters (predominantly weights and biases); and (3) RAM utilization (the activation buffer), which measures the dynamic heap allocated by the C-runtime engine exclusively for intermediate feature maps and input/output tensors [31].

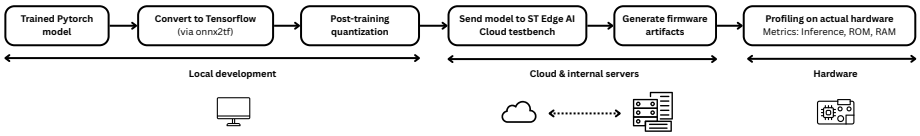

Fig. 4. Overview of the hardware benchmarking pipeline, enabling enabling automated cloud-based validation and profiling of quantized models without requiring physical hardware access.

5 Result Analysis

The following section details our experimental findings, offering an in-depth analysis of the isolated impacts of each pruning stage, alongside their integration to achieve optimal efficiency. All the results are displayed in the Table 4.

5.1 Isolated Pruning

Evaluating the first stage, we find that isolated input pruning (IP) successfully maintains the baseline performance, even when keeping only 40% to 50% of the input bins. For instance, on the UCI-HAR dataset, the DCT-IP configuration retains a mere 40.6% of the input size yet achieves a macro F1-score of 91.36%, slightly outperforming the unpruned DCT of 90.60%. A comparable trend emerges in the WEAR dataset. This sustained performance suggests that by actively stripping away high-frequency noise in the transformed domains, the pruning mechanism effectively forces the model to focus exclusively on the most informative frequency bands.

Alongside these accuracy improvements, input pruning drastically reduces inference latency and RAM usage. For the UCI-HAR dataset, transitioning from

the unpruned DCT to the balanced DCT-IP configuration accelerates inference by nearly 63% (dropping from 104.46 ms to 38.72 ms) and cuts RAM allocation by more than half (from 28.24 KB to 12.81 KB). However, despite these substantial runtime efficiencies, the static ROM requirement remains stubbornly fixed at 161.74 KB, as the downstream network architecture is unaltered.

To validate the physical intuition behind our learnable input pruning, we visualize the learned binary mask mapped back to the discrete frequency and wavelet bins. Figure 5 presents the input pruning masks for the UCI-HAR (DCT-IP) and the WEAR dataset (IHW-IP) across all validation subjects. The Gumbel-Softmax reparameterization successfully learns to preserve the fundamental movement components while automatically discarding higher frequency bins. For the DCT mask on UCI-HAR, active bins are heavily concentrated in the low-frequency range (bins 0–25), aligning with the biomechanical reality that human movement energy is concentrated in lower frequencies. Similarly, the IHW mask on WEAR consistently preserves the left half of the coefficients, corresponding to the macro-level movement shapes. This confirms that our methodology systematically removes hardware noise and irrelevant high-frequency signals without relying on manually cutoff thresholds.

The second stage of our methodology, channel pruning (CP), specifically addresses this ROM limitation. Similar to input feature reduction, we can selectively drop convolution channels without degrading predictive capacity. This phenomenon is observable across both datasets. Specifically, for the WEAR dataset, the unpruned FFT configuration yields an F1-score of 68.17%; however, applying a balanced channel pruning ratio that retains only 47% of the model's parameters actually elevates the F1-score to 69.05%.

Interestingly, while parameter reduction heavily optimizes ROM footprint and inference speed, it leaves overall RAM usage largely unaffected. In C-runtime embedded inference engines (such as the ST Edge AI runtime in this case), dynamic memory is allocated via a single, pre-calculated tensor arena. The size of this arena must accommodate the maximum memory bottleneck across the entire network, typically the largest intermediate activation buffer during a single layer's execution. Although channel pruning reduces the channel depth of intermediate activation buffers, it does not reduce the peak RAM requirement unless the specific layer acting as the network's absolute memory bottleneck is proportionally shrunk. Since the initial input buffers and unpruned layers remain constant and dictate this peak bottleneck, the overall runtime arena size stays fixed. In contrast, the ROM footprint (which stores the static weight matrices) and the inference latency (which is driven by the volume of multiply-accumulate operations) scale more directly with the degree of channel reduction.

5.2 Dual Pruning

Dual pruning synergistically combines the advantages of both pruning stages to yield a highly optimized, deployment-ready model. In the following paragraphs, we evaluate the efficacy of our dual-pruning (DP) strategy across two main datasets.

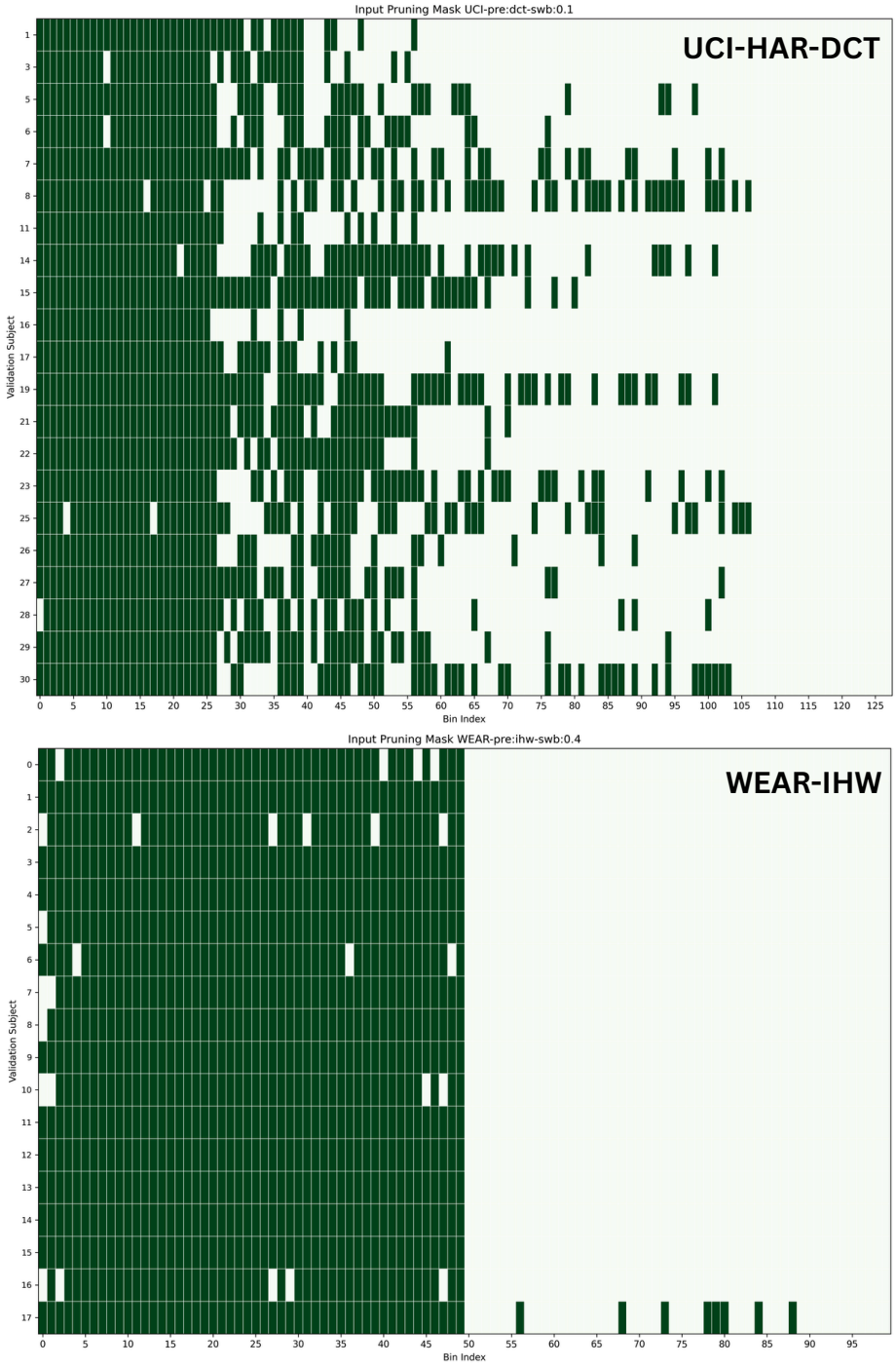

Fig. 5. Visualization of the learned input pruning masks for the DCT on UCI-HAR (top) and the IHW on WEAR (bottom). Dark green indicates a preserved bin, while light green indicates a pruned bin. The model consistently learns to retain low-frequency components and discard high-frequency noise.

For the WEAR dataset, IHW configuration emerges as the most competitive alternative to the time-domain baseline. Notably, IHW-DP approach achieves a superior F1-score of 71.87%, slightly outperforming the time-domain at 71.80%. Additionally, IHW-DP demonstrates exceptional cross-subject stability. Under LOSO validation scheme, its F1-scores maintain a tight bound between 71.20% and 72.50%, whereas the time-domain equivalent exhibits significant variance, fluctuating from 68.67% to 74.93%. This stability highlights a key advantage of wavelets over FFT and DCT: by preserving temporal resolution, wavelets effectively capture the transient energy bursts characteristic of outdoor sports movements. Beyond predictive robustness, IHW-DP is highly resource-efficient. The dual-pruning methodology yields a 3x reduction in inference latency (from 80.91 ms to 23.10 ms) and approximately halves the memory overhead, reducing the RAM requirement from 22.90 KB to 11.22 KB and the ROM footprint from 162.27 KB to 86.02 KB.

Conversely, for the UCI-HAR dataset, the DCT emerges as the optimal alternative. As illustrated in Table 4, applying dual pruning to the DCT yields a massive efficiency boost at the cost of a negligible 2.4% drop in F1-score, from 92.58% to 90.28%. This configuration achieves a 3.6x reduction in inference latency, dropping from 104.46 ms to an average of 24.72 ms, with performance ranging strictly between 18.14 ms and 31.30 ms. Furthermore, this dual-pruning approach successfully minimizes memory utilization, cutting the RAM requirement from 22.24 KB to 11.22 KB and the ROM footprint from 161.74 KB to 86.02 KB.

Finally, in scenarios where minimizing absolute latency and power consumption is the strict priority, the FFT provides unparalleled acceleration. Because the FFT output is symmetric, preprocessing only needs to process the positive half of the spectrum. Coupled with dual pruning, the FFT approach operates up to 2x faster than the other spectral methods evaluated in this study. For the WEAR dataset, this extreme efficiency comes with a trade-off: a 4% drop in F1-score is offset by a 5x speedup in inference latency, dropping from 80.91 ms down to an average of 12.53 ms.

6 Limitations

While our dual-stage pruning strategy demonstrates significant efficiency gains under the constraints of limited processing power and battery life, there are a few acknowledged limitations in our experimental setup. First, our current dual-stage methodology is architecturally dependent on Convolutional Neural Networks (CNNs). Both Stage 1 (input pruning) and Stage 2 (channel pruning) are specifically optimized for the structural properties of convolutional layers and their corresponding feature maps. Extending this learnable pruning framework to other emerging wearable architectures, such as Recurrent Neural Networks or Transformers, remains an area for future work. Second, our Gumbel-Softmax formulation treats discrete frequency and wavelet bins as independent categorical variables, effectively ignoring the innate harmonic structures of human move-

Table 4. Predictive performance and hardware efficiency of pruning configurations on both datasets, profiled on STM32F401 running at 84 MHz. Compression is quantified by input and parameter kept ratios (pruned relative to original). Suffixes indicate the pruning strategy: *no-suffix* (no pruning), *-IP* (input pruning, only stage 1), *-CP* (channel pruning, only stage 2) and *-DP* (dual pruning, applied both stage 1 and 2). Bold values indicate the best performance per dataset in each metric column. Detailed implementation is located in the github repository: `https://github.com/thong-phn/dual-pruning-har`

Dataset	Pre-processing	Input kept ratio	Model param. kept ratio	Float metrics (F32)		Quantized metrics (W8A16)		Efficiency		
				Accuracy (%)	F1 score (%)	Accuracy (%)	F1 score (%)	Inference (ms)	RAM (KB)	ROM (KB)
WEAR	TD	1.000	1.000	80.94 ± 2.13	71.80 ± 3.13	80.92 ± 2.31	72.07 ± 2.33	80.91 ± 0.01	22.87 ± 0.00	162.27 ± 0.00
	FFT	1.000	1.000	78.19 ± 1.89	68.17 ± 1.24	77.34 ± 2.79	67.74 ± 1.62	40.54 ± 0.00	13.14 ± 0.00	162.26 ± 0.00
	FFT-IP	0.503	1.000	78.16 ± 1.55	67.12 ± 1.86	78.39 ± 1.68	68.08 ± 1.42	22.26 ± 6.77	8.70 ± 1.78	160.99 ± 0.69
	FFT-CP	1.000	0.470	79.38 ± 0.66	69.05 ± 0.84	78.87 ± 0.79	68.77 ± 0.88	24.08 ± 1.96	11.31 ± 0.23	**84.24 ± 8.17**
	FFT-DP	0.503	0.485	78.40 ± 1.28	68.05 ± 1.36	78.23 ± 1.33	67.93 ± 1.32	**12.53 ± 3.35**	**7.20 ± 1.06**	85.15 ± 11.38
	DCT	1.000	1.000	77.78 ± 3.81	68.11 ± 1.91	77.41 ± 4.12	67.92 ± 2.14	80.90 ± 0.01	22.87 ± 0.00	162.27 ± 0.00
	DCT-IP	0.423	1.000	78.55 ± 1.69	66.71 ± 2.28	78.07 ± 1.96	68.04 ± 1.27	38.72 ± 8.49	12.81 ± 1.99	161.74 ± 0.25
	DCT-CP	1.000	0.498	79.40 ± 0.73	69.20 ± 0.61	79.12 ± 0.91	69.06 ± 0.59	47.38 ± 5.45	18.81 ± 0.34	88.02 ± 11.80
	DCT-DP	0.424	0.500	79.03 ± 0.98	68.36 ± 1.07	79.00 ± 1.00	68.35 ± 1.07	24.72 ± 6.57	11.22 ± 1.79	86.02 ± 12.22
	IHW	1.000	1.000	81.09 ± 2.24	72.07 ± 1.26	81.08 ± 2.26	72.07 ± 1.28	80.91 ± 0.01	22.87 ± 0.00	162.27 ± 0.00
	IHW-IP	0.490	1.000	81.30 ± 1.28	70.31 ± 4.46	81.27 ± 1.25	71.82 ± 1.05	42.45 ± 11.09	13.64 ± 2.78	161.54 ± 0.70
	IHW-CP	1.000	0.555	**81.52 ± 0.95**	**72.40 ± 0.81**	**81.83 ± 0.92**	**72.61 ± 0.77**	47.73 ± 3.35	18.67 ± 0.10	95.67 ± 5.43
	IHW-DP	0.490	0.523	81.36 ± 0.70	71.87 ± 0.65	81.36 ± 0.69	71.85 ± 0.65	23.10 ± 3.65	10.84 ± 0.96	91.74 ± 10.32
UCI-HAR	TD	1.000	1.000	**92.58 ± 1.18**	**92.60 ± 1.21**	**92.58 ± 1.19**	**92.60 ± 1.22**	104.46 ± 0.04	28.24 ± 0.00	161.74 ± 0.00
	FFT	1.000	1.000	88.83 ± 5.88	88.83 ± 6.46	86.39 ± 5.59	86.39 ± 6.20	52.35 ± 0.01	15.96 ± 0.00	161.72 ± 0.00
	FFT-IP	0.400	1.000	89.47 ± 1.83	89.91 ± 1.84	88.15 ± 1.65	88.24 ± 1.69	22.26 ± 6.77	8.70 ± 1.78	160.99 ± 0.69
	FFT-CP	1.000	0.476	88.52 ± 3.75	89.49 ± 3.89	86.73 ± 3.57	86.85 ± 3.69	31.65 ± 4.41	13.56 ± 0.31	84.44 ± 13.37
	FFT-DP	0.400	0.462	89.45 ± 1.44	89.58 ± 1.41	88.27 ± 1.40	88.41 ± 1.39	**14.06 ± 4.62**	**7.77 ± 1.53**	81.79 ± 10.20
	DCT	1.000	1.000	90.48 ± 1.87	90.60 ± 1.82	90.28 ± 1.83	90.41 ± 1.77	104.46 ± 0.04	28.24 ± 0.00	161.74 ± 0.00
	DCT-IP	0.406	1.000	91.12 ± 0.97	91.36 ± 0.94	91.04 ± 0.88	91.14 ± 0.85	38.72 ± 8.49	12.81 ± 1.99	161.74 ± 1.25
	DCT-CP	1.000	0.485	90.18 ± 1.45	90.82 ± 1.34	89.95 ± 1.45	90.11 ± 1.41	62.25 ± 9.40	23.63 ± 1.04	85.90 ± 13.10
	DCT-DP	0.406	0.486	90.28 ± 1.41	90.40 ± 1.36	90.15 ± 1.41	90.28 ± 1.37	24.72 ± 6.58	11.22 ± 1.79	86.02 ± 12.22
	IHW	1.000	1.000	88.77 ± 5.51	88.81 ± 5.94	88.77 ± 5.63	88.81 ± 6.04	104.46 ± 0.04	28.24 ± 0.00	161.74 ± 0.00
	IHW-IP	0.430	1.000	87.61 ± 5.14	88.78 ± 5.20	87.38 ± 5.02	87.59 ± 5.10	42.45 ± 11.09	13.64 ± 2.78	161.54 ± 0.70
	IHW-CP	1.000	0.434	87.40 ± 5.49	87.71 ± 5.33	87.19 ± 5.51	87.45 ± 5.52	55.17 ± 12.70	23.35 ± 1.10	78.26 ± 17.65
	IHW-DP	0.430	0.407	88.59 ± 3.17	88.86 ± 3.14	88.56 ± 3.05	88.85 ± 3.03	22.16 ± 5.77	11.63 ± 2.20	**74.19 ± 13.34**

ment. Although the network successfully compensates for this by learning robust downstream feature combinations, integrating harmonic constraints directly into the pruning mask could further optimize the representation.

7 Conclusions

In this paper, we presented a reproducible framework for deploying frequency- and wavelet-domain HAR models on severely constrained microcontrollers. By utilizing Gumbel-Softmax reparameterization, we demonstrated that aggressive dual-stage pruning of both transform bins and network channels yields up to a 3x inference speedup and reduces RAM requirements by over 50%, with a negligible impact on classification accuracy. Our evaluation underscores that there is no universal deployment strategy; rather, the optimal configuration is intrinsically tied to specific application constraints, ranging from predictable daily-activity monitoring and complex sports analytics to operation on strictly power-constrained edge devices.

Beyond these performance gains, this work explicitly addresses the ongoing reproducibility crisis. By open-sourcing our preprocessing pipelines and executing hardware-in-the-loop validation exclusively on the public ST Edge AI Developer Cloud, we establish a verifiable baseline that the ubiquitous computing community can seamlessly adopt and extend.

References

1. Agac, S., Durmaz Incel, Ö.: Lightweight deep learning for sensor-based har: Benchmarking optimization strategies. In: Durmaz Incel, Ö., Qin, J., Bieber, G., Kuijper, A. (eds.) Sensor-Based Activity Recognition and Artificial Intelligence. p. 77–98. Springer Nature Switzerland, Cham (2026)
2. Anguita, D., Ghio, A., Oneto, L., Parra, X., Reyes-Ortiz, J.L.: A public domain dataset for human activity recognition using smartphones. In: The European Symposium on Artificial Neural Networks (2013), https://api.semanticscholar.org/CorpusID:6975432
3. Antonsson, E.K., Mann, R.W.: The frequency content of gait. Journal of Biomechanics **18**(1), 39–47 (1985). https://doi.org/https://doi.org/10.1016/0021-9290(85)90043-0
4. Bian, S., Liu, M., Rey, V.F., Geißler, D., Lukowicz, P.: Tinierhar: Towards ultra-lightweight deep learning models for efficient human activity recognition on edge devices. In: Proceedings of the 2025 ACM International Symposium on Wearable Computers. pp. 163–169. ACM, Espoo Finland (Oct 2025). https://doi.org/10.1145/3715071.3750410, https://dl.acm.org/doi/10.1145/3715071.3750410
5. Bock, M., Kuehne, H., Van Laerhoven, K., Moeller, M.: Wear: An outdoor sports dataset for wearable and egocentric activity recognition. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. **8**(4) (Nov 2024). https://doi.org/10.1145/3699776, https://doi.org/10.1145/3699776
6. Cheng, H., Zhang, M., Shi, J.Q.: A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(12), 10558–10578 (Dec 2024). https://doi.org/10.1109/TPAMI.2024.3447085

7. 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 Transactions on Embedded Computing Systems **21**(4), 1–28 (2022). https://doi.org/10.1145/3542819
8. David, R., Duke, J., Jain, A., Reddi, V.J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., Warden, P.: Tensorflow lite micro: Embedded machine learning on tinyml systems (2021), https://arxiv.org/abs/2010.08678
9. Elfouly, T., Alouani, A.: A comprehensive survey on wearable computing for mental and physical health monitoring. Electronics **14**(17), 3443 (Aug 2025). https://doi.org/10.3390/electronics14173443
10. Gkountelos, D., Kokhazadeh, M., Bournas, C., Keramidas, G., Kelefouras, V.: Towards highly compressed cnn models for human activity recognition in wearable devices. In: 2023 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA). p. 83–88. IEEE, Poznan, Poland (Sep 2023). https://doi.org/10.23919/SPA59660.2023.10274441, https://ieeexplore.ieee.org/document/10274441/
11. Gou, H., Huang, M.: Real-time human activity recognition on edge devices using quantized deep neural networks. In: Agaian, S.S., Kong, X. (eds.) Second International Conference on Communication, Information, and Digital Technologies (CIDT 2025). vol. 14064, p. 140643O. SPIE (2026). https://doi.org/10.1117/12.3089273, https://doi.org/10.1117/12.3089273, backup Publisher: International Society for Optics and Photonics
12. Gupta, M., Camci, E., Keneta, V.R., Vaidyanathan, A., Kanodia, R., James, A., Foo, C.S., Wu, M., Lin, J.: Is complexity required for neural network pruning? a case study on global magnitude pruning. In: 2024 IEEE Conference on Artificial Intelligence (CAI). p. 747–754. IEEE, Singapore, Singapore (Jun 2024). https://doi.org/10.1109/CAI59869.2024.00144, https://ieeexplore.ieee.org/document/10605398/
13. Haseeb, A., Cleland, I., Nugent, C., McLaughlin, J.: Efficient and personalized federated learning for human activity recognition on resource-constrained devices. Applied Sciences **16**(2), 700 (Jan 2026). https://doi.org/10.3390/app16020700
14. Herrmann, C., Bowen, R.S., Zabih, R.: Channel selection using gumbel softmax. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M. (eds.) Computer Vision – ECCV 2020. Lecture Notes in Computer Science, vol. 12372, pp. 241–257. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-58583-9_15, https://link.springer.com/10.1007/978-3-030-58583-9_15
15. Huang, W., Zhang, L., Teng, Q., Song, C., He, J.: The convolutional neural networks training with channel-selectivity for human activity recognition based on sensors. IEEE Journal of Biomedical and Health Informatics **25**(10), 3834–3843 (Oct 2021). https://doi.org/10.1109/JBHI.2021.3092396
16. Huang, W., Zhang, L., Wang, S., Wu, H., Song, A.: Deep ensemble learning for human activity recognition using wearable sensors via filter activation. ACM Transactions on Embedded Computing Systems **22**(1), 1–23 (Jan 2023). https://doi.org/10.1145/3551486
17. Jang, E., Gu, S., Poole, B.: Categorical reparametrization with gumbel-softmax. In: Proceedings International Conference on Learning Representations (ICLR) (Apr 2017), https://openreview.net/pdf?id=rkE3y85ee

18. King, T., Zhou, Y., Röddiger, T., Beigl, M.: Micronas for memory and latency constrained hardware aware neural architecture search in time series classification on microcontrollers. Scientific Reports **15**(1), 7575 (Mar 2025). https://doi.org/10.1038/s41598-025-90764-z
19. Kumari, N., Yadagani, A., Behera, B., Semwal, V.B., Mohanty, S.: Human motion activity recognition and pattern analysis using compressed deep neural networks. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization **12**(1), 2331052 (Dec 2024). https://doi.org/10.1080/21681163.2024.2331052
20. Li, Z., Chen, G., Liu, X., Zhang, L., Sun, Q., Wang, K., Wu, H., Song, A.: Beyond 1 × 1 convolutions: A dynamic select-and-fuse channel sampling strategy for on-device human activity recognition. IEEE Internet of Things Journal **13**(7), 14923–14939 (Apr 2026). https://doi.org/10.1109/JIOT.2025.3650552
21. Liang, J., Zhang, L., Bu, C., Cheng, D., Wu, H., Song, A.: An automatic network structure search via channel pruning for accelerating human activity inference on mobile devices. Expert Systems with Applications **238**, 122180 (Mar 2024). https://doi.org/10.1016/j.eswa.2023.122180
22. Liang, J., Zhang, L., Han, C., Bu, C., Wu, H., Song, A.: A collaborative compression scheme for fast activity recognition on mobile devices via global compression ratio decision. IEEE Transactions on Mobile Computing **23**(4), 3259–3273 (Apr 2024). https://doi.org/10.1109/TMC.2023.3271306
23. Lorenser, T.: The dsp capabilities of arm® cortex®-m4 and cortex-m7 processors: Dsp feature set and benchmarks. White paper, ARM Limited (2016)
24. Mahor, V., Pandit, J.: A neural network pruning approach to human activity identification based on deep learning. In: 2022 IEEE International Conference on Current Development in Engineering and Technology (CCET). p. 1–9. IEEE, Bhopal, India (Dec 2022). https://doi.org/10.1109/CCET56606.2022.10079945, https://ieeexplore.ieee.org/document/10079945/
25. Makhoul, J.: A fast cosine transform in one and two dimensions. IEEE Transactions on Acoustics, Speech, and Signal Processing **28**(1), 27–34 (1980). https://doi.org/10.1109/TASSP.1980.1163351
26. Nachin, M., Desai, D., Jia, S.S., Lai, C., Liu, M., Szwejbka, J., Alvarez, R., Ascani, R., Bort, D., Candales, M., Caples, A., Cao, Y., Chen, Z., Chintala, S., Comer, G., Islam, T., Jia, S., Karuturi, T., Khuu, J., Kukkadapu, A., Manlaibaatar, T., Or, A., Patel, K., Pothapragada, S., Qiu, L., Rao, S., Reblitz-Richardson, O., Ren, M., Roy, S., Shoumikhin, A., Wolchok, S., Yang, G., Yi, A., Yuan, M., Zhang, H., Zhang, J., Zhang, J., Zhang, S., Bilgin, C.C.: Executorch – a unified pytorch solution to run ai models on-device (2026), https://arxiv.org/abs/2605.08195
27. Najeh, H., Lohr, C., Leduc, B.: Real-time human activity recognition on embedded equipment: A comparative study. Applied Sciences **14**(6), 2377 (Mar 2024). https://doi.org/10.3390/app14062377
28. Pascale, R., Tudose, D., Ţurcanu, T.: Open hardware and software smartwatch. In: 2024 23rd RoEduNet Conference: Networking in Education and Research (RoEduNet). p. 1–5. IEEE, Bucharest, Romania (2024). https://doi.org/10.1109/RoEduNet64292.2024.10722548, https://ieeexplore.ieee.org/document/10722548/
29. Saha, B., Samanta, R., Ghosh, S., Roy, R.B.: Bandx: An intelligent iot-band for human activity recognition based on tinyml. In: Proceedings of the 24th International Conference on Distributed Computing and Networking. p. 284–285. ACM, Kharagpur India (Jan 2023). https://doi.org/10.1145/3571306.3571415, https://dl.acm.org/doi/10.1145/3571306.3571415

30. Sharma, V., Pau, D., Cano, J.: Efficient tiny machine learning for human activity recognition on low-power edge devices. In: 2024 IEEE 8th Forum on Research and Technologies for Society and Industry Innovation (RTSI). p. 85–90. IEEE, Milano, Italy (Sep 2024). https://doi.org/10.1109/RTSI61910.2024.10761203, https://ieeexplore.ieee.org/document/10761203/
31. STMicroelectronics: ST Edge AI Core Technology Documentation. STMicroelectronics (2026), https://stedgeai-dc.st.com/assets/embedded-docs/index.html, accessed: 2026-05-01
32. Su, J., Shao, H., Deng, X., Jiang, Y.: An investigation of frequency-domain pruning algorithms for accelerating human activity recognition tasks based on sensor data. Computers, Materials & Continua **81**(2), 2219–2242 (2024). https://doi.org/10.32604/cmc.2024.057604
33. Tekin, N., Aris, A., Acar, A., Uluagac, S., Gungor, V.C.: A review of on-device machine learning for iot: An energy perspective. Ad Hoc Networks **153**, 103348 (Feb 2024). https://doi.org/10.1016/j.adhoc.2023.103348
34. Uddin, M.H., Ara, J.M.K., Rahman, M.H., Yang, S.: Neural network pruning: An effective way to reduce the initial network for deep learning based human activity recognition. In: 2021 International Conference on Electronics, Communications and Information Technology (ICECIT). p. 1–4. IEEE, Khulna, Bangladesh (Sep 2021). https://doi.org/10.1109/ICECIT54077.2021.9641226, https://ieeexplore.ieee.org/document/9641226/
35. Yao, M., Cheng, D., Zhang, L., Liu, L., Song, S., Wu, H., Song, A.: A sparse diverse-branch large kernel convolutional neural network for human activity recognition using wearables. Applied Soft Computing **167**, 112444 (Dec 2024). https://doi.org/10.1016/j.asoc.2024.112444
36. Youm, S., Go, S.: Lightweight and efficient csi-based human activity recognition via bayesian optimization-guided architecture search and structured pruning. Applied Sciences **15**(2), 890 (Jan 2025). https://doi.org/10.3390/app15020890
37. Zhou, H., Zhang, X., Feng, Y., Zhang, T., Xiong, L.: Efficient human activity recognition on edge devices using deepconv lstm architectures. Scientific Reports **15**(1), 13830 (Apr 2025). https://doi.org/10.1038/s41598-025-98571-2
38. Zhou, Y., King, T., Huang, Y., Zhao, H., Riedel, T., Röddiger, T., Beigl, M.: Enhancing efficiency in har models: Nas meets pruning. In: 2024 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops). p. 33–38. IEEE, Biarritz, France (Mar 2024). https://doi.org/10.1109/PerComWorkshops59983.2024.10502894, https://ieeexplore.ieee.org/document/10502894/
39. Zhou, Y., Yang, Z., Zhang, X., Wang, Y.: A hybrid attention-based deep neural network for simultaneous multi-sensor pruning and human activity recognition. IEEE Internet of Things Journal **9**(24), 25363–25372 (Dec 2022). https://doi.org/10.1109/JIOT.2022.3196170

A Appendix: Preprocessing Overhead

To evaluate the computational overhead of the different preprocessing methods, we analyzed the required CPU cycle costs with the ARM CMSIS-DSP library [23]. For the WEAR dataset, as the APIs for frequency-domain transformations require the window size to be a power of 2, we had to apply zero-padding with an additional 28 samples ($100+28 = 2^7$). For the UCI-HAR dataset, the original window size is already 128 samples. As the additional zero-padding computation is minimal compared to the total execution time, we simplified our evaluation by focusing on the 128-sample window.

FFT. ARM's CMSIS-DSP complex FFT function, `arm_cfft_f32`, requires RAM data to be formatted as an interleaved array of alternating real and imaginary components (e.g., [Real_0, Imag_0, Real_1, Imag_1, ...]). The implementation is discussed below, and the estimated computational cycle cost for a single 1D sensor channel is summarized in Table 5.

Input: X: Buffer of length N containing raw 1D sensor data
Output: M: Buffer of length $N/2 + 1$ containing positive half-spectrum magnitudes

`/* Step 1: Complex vector construction (Interleaving) */`
for $i \leftarrow 0$ **to** $N - 1$ **do**
  $Z[2i] \leftarrow X[i]$ `// Real component`
  $Z[2i + 1] \leftarrow 0$ `// Imaginary component`
end

`/* Step 2: Fast Fourier Transform using CMSIS-DSP arm_cfft_f32 */`
$Z \leftarrow \text{cFFT}(Z, N)$

`/* Step 3: Magnitude calculation of the positive half */`
for $k \leftarrow 0$ **to** $\frac{N}{2}$ **do**
  $M[k] \leftarrow \sqrt{Z[2k]^2 + Z[2k + 1]^2}$
end

Algorithm 1: FFT Implementation

Table 5. Estimated CPU cycle cost for FFT on 1D sensor input data ($N = 128$). Operations leverage the hardware FPU in the Cortex-M4.

Step	Cycle Cost	Comments
Vector construction	≈ 900	128 samples; ≈ 7 cycles/sample to load real and zero-pad imaginary components.
FFT calculation	$13,823$	Reference value from [23].
Magnitude calculation	$\approx 1,430$	65 bins computed using the hardware FPU (≈ 22 cycles/bin).
Total	$\approx 16,153$	

DCT. As DCT-II algorithm is not natively supported by the ARM CMSIS-DSP library, a custom implementation is necessary. This can be efficiently achieved by leveraging the existing `arm_rfft_fast_f32` function, following the strategy originally proposed by Makhoul [25]. Traditionally, the DCT of an N-point real signal is derived by computing the DFT of its $2N$-point even extension. However,

Makhoul demonstrated that the exact same result can be obtained using only an N-point DFT applied to a reordered version of the original signal, effectively reducing the computational overhead by half.

Table 6. Estimated CPU cycle cost for DCT on 1D sensor input data ($N = 128$). Operations leverage the hardware FPU unit in the Cortex-M4.

Step	Cycle Cost	Comments
Signal Rearrangement	≈ 450	64 iterations of load/store operations and pointer arithmetic.
Real FFT calculation	$7,714$	Reference value from [23].
Phase shift (Lower half)	≈ 910	65 iterations of load/store, MAC, and MUL operations (≈ 14 cycles/bin).
Phase shift (Upper half)	≈ 882	63 iterations of load/store, MSUB, and MUL operations (≈ 14 cycles/bin).
Total	$\approx 9,956$	

Input: X: Buffer of length N containing raw 1D sensor data
Output: C: Buffer of length N containing DCT coefficients

`/* Step 1: Buffer construction */`
for $i \leftarrow 0$ **to** $\frac{N}{2} - 1$ **do**
  $Y[i] \leftarrow X[2i]$ `// Even indices into first half`
  $Y[N-1-i] \leftarrow X[2i+1]$ `// Odd indices (reversed order) into second half`
end

`/* Step 2: Real Fast Fourier Transform */`
$Z \leftarrow \mathrm{rFFT}(Y, N)$ `// Using arm_rfft_fast_f32 from CMSIS-DSP`

`/* Step 3.1: Phase shift (Lower half, even indices) */`
for $k \leftarrow 0$ **to** $\frac{N}{2}$ **do**
  `// Multiply by phase factor` $e^{-j\frac{\pi k}{2N}}$
  $C[k] \leftarrow 2 \cdot \left(\Re(Z[k]) \cos\left(\frac{\pi k}{2N}\right) + \Im(Z[k]) \sin\left(\frac{\pi k}{2N}\right)\right)$
end

`/* Step 3.2: Phase shift (Upper half, odd indices) */`
for $k \leftarrow \frac{N}{2} + 1$ **to** $N - 1$ **do**
  `// Utilize Hermitian symmetry` $Z[k] = Z^*[N-k]$ `for the remaining bins`
  $C[k] \leftarrow 2 \cdot \left(\Re(Z[N-k]) \cos\left(\frac{\pi k}{2N}\right) - \Im(Z[N-k]) \sin\left(\frac{\pi k}{2N}\right)\right)$
end

Algorithm 2: DCT-II Implementation

IHW. Unlike the heavy floating-point operations of the FFT or DCT, the IHW requires no FPU, no multiplications, and no complex array rearrangements. The algorithm inherently processes data in pairs, the table reflects the CPU cycle cost per pair of samples (one even, one odd). In this case, the total sample is either 100 or 128 for 1D sensor, which is even. So there no need to check for even at the end of the algorithm.

Input: X: Buffer of length N containing raw ADC samples (assumed 16-bit integers)
Output: Y: Buffer of length N containing IHW coefficients (16-bit signed integers)

$H \leftarrow \lfloor N/2 \rfloor$ `// Half-length of the buffer`

`/* Compute Approximation (`A`) and Detail (`D`) coefficients */`
for $i \leftarrow 0$ **to** $H - 1$ **do**
  $E \leftarrow X[2i]$ `// Even sample`
  $O \leftarrow X[2i + 1]$ `// Odd sample`
  $D \leftarrow O - E$
  $A \leftarrow E + (D \gg 1)$
  $Y[i] \leftarrow A$
  $Y[H + i] \leftarrow D$
end

Algorithm 3: IHW implementation

Table 7. Estimated CPU cycle cost for IHW on 1D sensor input ($N = 128$). No FPU is required.

Step	Cycle Cost	Comments
Initialization	≈ 4	Computes $H = N/2$ and loads the initial memory pointers.
Calculation loop	704	64 iterations $\times$ 11 cycles per pair (split, subtract, add with in-line shift, and store).
Total	≈ 708	