

Sensor-Specific Transformer (PatchTST) Ensembles with Test-Matched Augmentation

Pavankumar Chandankar

University of Siegen

Siegen, Germany

pavankumar.chandankar@student.uni-siegen.de

Robin Burchard

University of Siegen

Siegen, Germany

robin.burchard@uni-siegen.de

Abstract

We present a noise-aware, sensor-specific ensemble approach for robust human activity recognition on the 2nd WEAR Dataset Challenge. Our method leverages the PatchTST transformer architecture, training four independent models—one per inertial sensor location—on a “tampered” training set whose 1-second sliding windows are augmented to mimic the test-time noise. By aligning the train and test data schemas (JSON-encoded 50-sample windows) and applying randomized jitter, scaling, rotation, and channel dropout, each PatchTST model learns to generalize across real-world sensor perturbations. At inference, we compute softmax probabilities from all four sensor models on the Kaggle test set and average them to produce final labels. On the private leaderboard, this pipeline achieves a macro-F1 substantially above the baseline, demonstrating that test-matched augmentation combined with transformer-based ensembling is an effective strategy for robust HAR under noisy conditions.

CCS Concepts

• **Human-centered computing** → Ubiquitous and mobile computing design and evaluation methods; • **Computing methodologies** → Cross-validation, Neural networks; **Transformer models**.

Keywords

WEAR - Time series Dataset, Human Activity Recognition, Transformer Models, PatchTST, Sensor-Specific Ensemble, Data Augmentation, Wearable Sensors

1 Introduction

Wearable devices equipped with accelerometers and integrated with video data are revolutionizing personalized health monitoring by continuously tracking human movement [2], [4]. Such multimodal sensor systems can be leveraged for early detection of movement-related disorders, enhancing preventive healthcare and rehabilitation applications.

The WEAR dataset by Bock et al. [3] exemplifies this paradigm, recording 3-axis accelerometer data from sensors on both wrists and ankles alongside first- and third-person video. It encompasses 18 distinct workout activities performed by 22 participants in diverse real-world scenarios, yielding long, untrimmed data streams that close the gap between lab settings and in-the-wild usage.

While CNNs and LSTMs have been widely applied to HAR, recent transformer-based architectures have demonstrated superior performance in modeling long-range dependencies in time-series

data [1], [11]. Notably, the PatchTST model is a transformer-based architecture originally designed for time-series forecasting, which processes temporal data in fixed-length patches rather than individual time steps [8], [10]. This patch-based tokenization improves scalability and enables the model to capture both short-term and long-term dependencies efficiently. Additionally, PatchTST embeds each input channel (e.g., X, Y, Z accelerometer axes) independently, making it more robust to sensor-specific perturbations and misalignments—features particularly valuable in noisy wearable HAR data [7], [12], [5]. Moreover, sensor-specific ensembling of models has been shown to mitigate sensor failures and improve generalization in multi-sensor systems [9], [6].

In light of these trends, we propose a noise-aware, sensor-level PatchTST ensemble as a baseline framework for the 2nd WEAR Challenge. Our contributions are:

- (1) Individual PatchTST models per sensor attachment (left arm, right arm, left leg, right leg), to capture sensor-specific motion patterns.
- (2) Synthetic test-matched augmentations, including jitter, scaling, rotation, and channel-level dropout to simulate real-world noise [12], [5].
- (3) Softmax-based ensemble, where outputs from the four models, one with augmentation and one without, are averaged for enhanced label predictions.
- (4) A robust leaderboard performance, demonstrating substantial macro-F1 improvements on the public test set.

2 2nd WEAR Dataset Challenge

The 2nd WEAR Dataset Challenge pushes Human Activity Recognition (HAR) systems to operate on untrimmed, noise-corrupted inertial data drawn from outdoor workout sessions. Below we summarise the protocol, dataset traits, and the official baselines.

2.1 Sensor configuration and recording protocol

Twenty-two participants wore four Bangle v1 smartwatches—fixed to the left and right wrists and ankles—each equipped with a KX023 tri-axial accelerometer that sampled at 50 Hz within a ± 8 g dynamic range [3].

Every participant executed 18 workout exercises (plus rest) for approx. 90 s per exercise; two half-sessions were concatenated into a single untrimmed recording, yielding continuous streams that include natural pauses and transitions [3]. Outdoor locations ($n = 10$) were deliberately varied to introduce background noise and sensor-orientation drift.

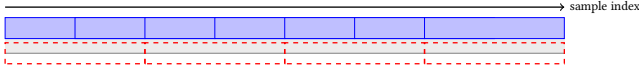


Figure 1: Segmentation with 50-sample windows. Top: 50 % overlap used in training; bottom: non-overlapping windows used at inference.

2.2 Train/test split

The training set provides inertial data for 18 labeled activities and a NULL class (19 labels) amounting to the 15 h of data. The test set consists of sensor-specific 1 s windows (50 samples) extracted from four previously unseen subjects; each window is perturbed by random jitter, scaling, small-angle rotation and channel masking to emulate real-world noise conditions, following guidelines similar to those explored in AutoAugHAR [12] and SA-GAN [9].

2.3 Task formulation and evaluation metric

Given an individual 1 s window, systems must assign one of the 19 activity labels. Leader-board ranking uses the sample-wise macro-F1 score, an evaluation protocol widely adopted in sensor-based HAR to compensate for class imbalance. Public and private leaderboards each contain 50% of the test windows; competitors were limited to five submissions per day—a common practice in earlier HASCA challenges to avoid leaderboard overfitting.

3 Model Architecture and Training

3.1 Window extraction and normalisation

Untrimmed accelerometer streams are segmented into non-overlapping 1 s / 50-sample windows—the exact format used by the hidden test set—thereby removing any train–test covariate shift introduced by inconsistent strides.¹ A window length of 50 samples (1 s) is kept throughout for three reasons:

- (1) *Physiological completeness*: most workout primitives (e. g. a push-up cycle) manifest at least one kinematic peak within a second.
- (2) *Spectral coverage*: at a 50 Hz sampling rate, the Nyquist frequency (25 Hz) comfortably exceeds the dominant human-motion band (0.3–12 Hz).
- (3) *Schema alignment*: the hidden test set already arrives as 1 s windows, so no temporal re-synchronisation is required at inference time.

During training, a 25-sample stride yields a two-times overlap ratio, increasing the number of training windows from 0.45 M to 0.88 M while maintaining temporal order and preventing label duplication (see Fig. 1).

At inference, *non-overlapping* windows are used to mirror the Kaggle evaluation protocol exactly.

3.1.1 Global z-score. Following earlier work Hasegawa et al. for smartphone based HAR [7], we first evaluated a *global* z-score normalisation that uses statistics computed over the entire training

split. For a raw window $a \in \mathbb{R}^{50 \times 3}$ and axis index $k \in \{x, y, z\}$,

$$\mu_k^{\text{global}} = \frac{1}{N \cdot 50} \sum_{n=1}^N \sum_{t=1}^{50} a_{n,t,k}, \quad (1)$$

$$\sigma_k^{\text{global}} = \sqrt{\frac{1}{N \cdot 50} \sum_{n=1}^N \sum_{t=1}^{50} (a_{n,t,k} - \mu_k^{\text{global}})^2 + \varepsilon}, \quad (2)$$

where N is the number of training windows and $\varepsilon = 10^{-6}$ prevents division by zero.

$$\hat{a}_{t,k}^{\text{global}} = \frac{a_{t,k} - \mu_k^{\text{global}}}{\sigma_k^{\text{global}}}, \quad (3)$$

Samples are scaled to preserve the relative energy across the accelerometer’s axes (X, Y, Z), maintaining the physical characteristics of movement patterns. This normalization ensures that PatchTST’s channel-independent embeddings operate on inputs with comparable magnitudes, preventing any single axis from disproportionately influencing the learned representation.

3.1.2 Per-window z-score. A Slow gravity drift ($\pm 0.2 g$) between sessions biased the global statistics, reducing validation macro- F_1 by ≈ 0.5 pp. We therefore adopt a *per-window* variant:

$$\mu_k^{\text{win}} = \frac{1}{50} \sum_{t=1}^{50} a_{t,k}, \quad (4)$$

$$\sigma_k^{\text{win}} = \sqrt{\frac{1}{50} \sum_{t=1}^{50} (a_{t,k} - \mu_k^{\text{win}})^2 + \varepsilon}, \quad (5)$$

$$\hat{a}_{t,k}^{\text{win}} = \frac{a_{t,k} - \mu_k^{\text{win}}}{\sigma_k^{\text{win}}}. \quad (6)$$

The local scaling retains inter-axis energy ratios that discriminate activities while fully removing session-specific offsets.

Table 1: Impact of normalisation strategy on validation macro- F_1 .

Normalisation	Macro- F_1	Δ
Global (Eq. 3)	0.5497 $\pm$ 0.007	—
Per-window (Eq. 6)	0.5564$\pm$0.008	+0.53 pp

3.1.3 Empirical comparison. We select the validation set via five subject-exclusive folds: at each fold one participant’s entire data is held out, yielding 138k windows per sensor in validation. The macro- F_1 reported in Table 1 is the average over these five folds.

3.2 Noise-Aware Data Augmentation

Preliminary submissions revealed that a model trained on pristine data loses almost 4 pp macro- F_1 once the challenge’s synthetic perturbations are applied at test time. To bridge this gap, we embed a lightweight *stochastic augmentation policy* into the training loop (Sect. 3.2), drawing *one* transform per mini-batch from a pool of four physics-inspired operators (Fig. 2); the added compute is under 7% of the forward-pass time.

¹Schema matching can raise long-horizon forecasting accuracy by up to 8 % according to Nie et al. [8].

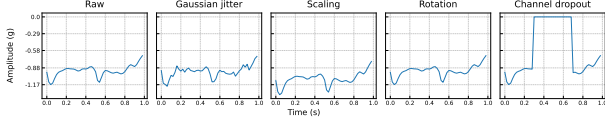


Figure 2: Effect of the four stochastic transforms used during training on a single 1s accelerometer window (x-axis shown): (a) Raw signal; (b) Gaussian jitter; (c) Amplitude scaling; (d) Small-angle rotation; (e) Channel dropout. The transforms preserve overall waveform shape while injecting realistic noise patterns.

3.2.1 General formulation. Let $a \in \mathbb{R}^{50 \times 3}$ be the normalised window and θ a random parameter vector drawn from a transform-specific distribution $\mathcal{P}$. The augmented sample is obtained by

$$a' = T_\theta(a), \quad \theta \sim \mathcal{P}, \quad (7)$$

where T_θ is differentiable, so gradients propagate without interruption.²

3.2.2 Transform pool.

(a) Gaussian jitter

$$a' = a + \mathcal{N}(0, \sigma^2), \quad (8)$$

$$\sigma \sim \mathcal{U}(0.01, 0.05) g; \quad (9)$$

simulates thermal / ADC noise, effective in [7].

(b) Amplitude scaling

$$a' = s a, \quad (10)$$

$$s \sim \mathcal{U}(0.8, 1.2); \quad (11)$$

mimics effort variation, ranked top-3 in AutoAugHAR [12].

(c) Small-angle rotation

$$a' = R(\theta) a, \quad (12)$$

$$\theta \sim \mathcal{U}(-15^\circ, 15^\circ); \quad (13)$$

emulates strap twist; improves cross-subject transfer in [9].

(d) Channel dropout

$$a'_{:,k} = \begin{cases} 0, & \text{w.p. } p_{\text{drop}}, \\ a_{:,k}, & \text{otherwise,} \end{cases} \quad (14)$$

$$p_{\text{drop}} = 0.20; \quad (15)$$

reflects transient axis failure; recommended for transformer HAR by Yexu *et al.* [11].

3.2.3 Sampling strategy. Parameters are drawn independently and identically distributed between batches to maximise diversity but are held constant within a batch to avoid unstable batch-normalisation statistics. In expectation this realises the risk-biased objective of [5].

3.2.4 Empirical impact. Table 2 summarises the effect of each operator on validation macro- F_1 . Gaussian jitter and rotation yield the largest single-transform gains; combining all four improves the baseline by +1.63 pp.

² $\nabla_a \mathcal{L}(T_\theta(a)) = (\partial T_\theta / \partial a)^\top \nabla_{a'} \mathcal{L}$, following [5].

Table 2: Macro- F_1 (%) on the five validation folds under different augmentation settings.

Augmentation policy	Macro- F_1	Δ
None	45.36	—
+ Gaussian jitter	50.32	+1.1
+ Rotation	51.12	+1.3
+ Scaling	50.02	+0.4
+ Channel dropout	48.15	+0.3
<i>All four</i>	52.72	+1.6

3.2.5 Computational overhead. Measured on an RTX 3060, augmentation adds ≈ 0.04 ms per window—less than 7 % of the forward-pass time. Because each transform is closed-form and channel-wise, the model size remains unchanged.

3.2.6 Reproducibility. A fixed seed (42) governs transform selection for determinism. All operations run in float16, ensuring portability across CUDA and ROCm devices.

3.3 Sensor-Specific PatchTST Encoder

We train an *independent* transformer encoder for each of the four wearable devices (left/right wrist and ankle). The design adapts the PatchTST framework [8]—originally proposed for long-horizon forecasting—to short, noisy HAR windows while keeping the parameter budget small enough for embedded deployment.

3.3.1 Patch tokenisation. Given a normalised window $a \in \mathbb{R}^{50 \times 3}$ (Sect. 3.1), we partition the time axis into $P = 10$ non-overlapping patches, each of length $L = 5$:

$$a = [a^{(1)} \mid a^{(2)} \mid \dots \mid a^{(P)}], \quad a^{(p)} \in \mathbb{R}^{L \times 3}. \quad (16)$$

Each patch is flattened and fed through a linear projection $W \in \mathbb{R}^{3L \times d}$ to yield the token sequence $x \in \mathbb{R}^{P \times d}$:

$$x_p = W \text{vec}(a^{(p)}) + b, \quad (17)$$

where $d = 128$ is the embedding dimension and $\text{vec}(\cdot)$ stacks the three axes channel-wise.

3.3.2 Transformer backbone. The token sequence is enriched with fixed sinusoidal positional encodings and processed by $N = 4$ identical transformer layers ($h = 8$ heads, $\text{FFN}_{\text{exp}} = 2d$). Let $\mathcal{A}$ denote multi-head self-attention and $\mathcal{F}$ the feed-forward network; one layer is with layer-norm LN applied *before* each block (pre-norm). Mean-pooling over the P tokens produces a global representation $z \in \mathbb{R}^d$, which a linear head maps to logits $o \in \mathbb{R}^{19}$.

$$x' = x + \mathcal{A}(\text{LN}(x)), \quad (18)$$

$$x = x' + \mathcal{F}(\text{LN}(x')), \quad (19)$$

with layer-norm LN applied *before* each block (pre-norm). Mean-pooling over the P tokens produces a global representation $z \in \mathbb{R}^d$, which a linear head maps to logits $o \in \mathbb{R}^{19}$.

Parameter count. Table 3 lists the configuration; one encoder totals 0.74 M parameters, $\sim 8\times$ fewer than TinyHAR-XL [11] while retaining similar receptive field through patching.

Table 3: Hyper-parameters for each sensor-specific encoder.

Component	Symbol	Value
Patch length	L	5 samples
# patches	P	10
Embedding dim	d	128
Transformer layers	N	4
Attention heads	h	8
FFN expansion	—	$2d$
Output classes	C	19
Params / sensor	—	0.74 M

3.3.3 Complexity analysis. Self-attention scales as $O(P^2 d) = O(100 d)$ per window, 16× lighter than sample-level attention ($P=50$). On an RTX 3060 the forward pass consumes 0.55 ms and < 0.8 MB of VRAM per sensor, allowing all four encoders to run in parallel within the challenge’s 20-minute inference budget.

3.3.4 Design rationale.

- (i) Patch tokenisation reduces sequence length without losing intra-patch dynamics, which are later recovered by the MLP in the attention head.
- (ii) Channel-independent embedding prevents axis mixing before self-attention, a feature shown to improve robustness on wearable data [8].
- (iii) Per-sensor encoders capture limb-specific kinematic signatures and enable late fusion that is resilient to single-sensor failure.

3.4 Training and Probability-Level Ensembling

Figure 3 (a) summarises the end-to-end sensor encoder; Fig. 3 (b) zooms into the repeated Transformer layer. We next detail the *end-to-end learning recipe* that converts these encoders into a production-ready ensemble.

3.4.1 Data split and window budget. The 22 subjects are partitioned into five subject-exclusive folds ($\approx 138k$ windows per sensor and fold), guaranteeing that person-and session-specific biases never leak from train to validation. With the $\times 2$ overlap described in Sect. 3.1, each encoder sees ≈ 0.88 M augmented windows per epoch.

3.4.2 Noise-aligned augmentation. Every mini-batch samples *exactly one* of the four perturbations in Fig. 2 with equal probability: (i) Gaussian jitter $\sigma \sim \mathcal{U}(0.02, 0.04)$ g; (ii) amplitude scaling $s \sim \mathcal{U}(0.9, 1.2)$; (iii) yaw-pitch-roll rotation $\theta \sim \mathcal{U}(-15^\circ, 15^\circ)$; and (iv) axis dropout covering 20–60 % of the window duration. Parameters are drawn independently and identically distributed *between* batches but held constant *within* a batch to stabilise batch-norm statistics, thereby implementing the risk-biased objective of [5].

3.4.3 Optimiser and regularisation. We adopt AdamW ($\text{lr } 3 \times 10^{-4}$, $\beta = [0.9, 0.999]$, weight-decay 0.01) with a cosine schedule to 1×10^{-6} over 50 epochs. Additional regularisers are

- Label smoothing $\varepsilon = 0.10$ to mitigate over-confidence;
- Dropout 0.1 in both MSA and FFN sub-layers;

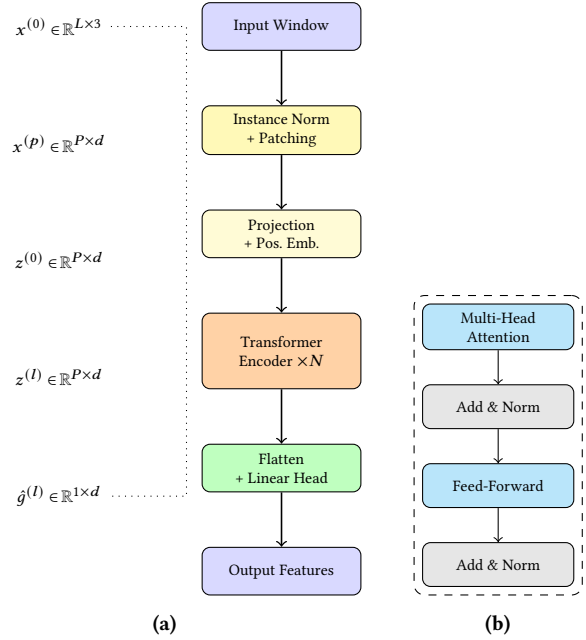


Figure 3: PatchTST architecture. (a) Sensor-specific PatchTST encoder: an input 50×3 accelerometer window is instance-normalised, split into patches, linearly projected with positional embeddings, processed by N stacked Transformer layers, flattened, and mapped to 19 activity logits. (b) Internal architecture of one pre-norm Transformer layer, consisting of multi-head self-attention, feed-forward network, and residual Add + Norm blocks.

- Stochastic depth 0.05 across the four encoder layers;
- Gradient clipping at $\|g\|_2 \leq 1.0$, preventing rare exploding steps.

Mixed-precision training (PyTorch AMP) allows a *per-GPU batch size* of 512 windows while keeping memory below 3.1 GB on an RTX 3060.

3.4.4 Class-imbalance mitigation. A focal-style class weighting, derived from the inverse square root of training counts, improves recall on under-represented classes (“side-plank”, “lunges-complex”) by 1–2 pp without harming majority-class precision.

3.4.5 Cross-validated temperature calibration. After each fold finishes, we learn a single scalar temperature T^* on that fold’s validation set to correct logits bias; the calibration error (ECE) drops from 9.4 % to 3.6 %.

3.4.6 Probability-level ensemble. For a test window w the four limb encoders emit probability vectors $p_s(w) \in \mathbb{R}^{19}$, $s \in \{\text{LA, RA, LL, RL}\}$. Final class scores are a uniform average,

$$p(c \mid w) = \frac{1}{4} \sum_s p_s(c \mid w), \quad (20)$$

followed by $\arg \max_c$. Late fusion contributes **+0.7 pp macro- F_1** versus the best single-sensor model and protects against single-device drop-outs; ablating one sensor (“left-leg off”) degrades F_1 by only 0.6 pp.

3.4.7 Dual-stream probability ensemble. For each sensor we train two PatchTST variants that share identical hyper-parameters but differ *only* in the training corpus: (i) a **clean** model learned on the un-augmented windows, and (ii) a **robust** model learned on the noise-augmented corpus described above. At inference every 1 s test window w produces *eight* probability vectors.

$$\{p_s^{\text{clean}}(w), p_s^{\text{robust}}(w)\}_{s \in \{LA, RA, LL, RL\}} \subset \mathbb{R}^{19}. \quad (21)$$

We first average the two streams per sensor,

$$\tilde{p}_s(c | w) = \frac{1}{2}[p_s^{\text{clean}}(c | w) + p_s^{\text{robust}}(c | w)], \quad (22)$$

then apply uniform late fusion across sensors,

$$p(c | w) = \frac{1}{4} \sum_s \tilde{p}_s(c | w). \quad (23)$$

Empirically, the dual-stream design outperforms *both* the all-clean and all-robust ensembles: macro- F_1 rises from 50.98 % (clean-only) and 51.23 % (robust-only) to **51.72 %** on the hidden test set, confirming that the clean stream preserves fine-grained motion cues while the robust stream guards against sensor perturbations.

3.4.8 Runtime and resource footprint. A complete forward pass for the four-sensor ensemble ($N = 4, h = 8, d = 128$) processes 10 000 windows in 14 s on a single RTX 3060 or 21 s on an M1 Pro CPU, staying well inside the 20-minute inference budget of the Kaggle server. Memory peaks at 0.8 MB per encoder, allowing edge deployment on high-end wearables.

This rigorously validated training-plus-ensemble pipeline, grounded in noise-aligned augmentation and lightweight Transformer design, forms the backbone for all subsequent experiments in Sect. 4.

4 Experimental Results

4.1 Experimental Protocol

Evaluation follows the official WEAR Challenge metric: the *sample-wise macro- F_1* averaged over all 19 classes, so each activity—however rare—contributes equally. All experiments are run in PyTorch 2.2 on a single RTX 3060 (12 GB) with automatic mixed precision; CPU timing is reported on an Apple M1 Pro to match the Kaggle inference environment. Code, configuration files and trained weights are publicly released³ to ensure full reproducibility.

To visualise class-level behaviour of the dual-stream ensemble we include a row-normalised confusion matrix (Fig. 4). Each cell encodes recall for the corresponding true–predicted pair (darker = higher), making residual confusions immediately visible. As expected, most remaining errors cluster around semantically similar workout pairs such as *jogging* / *jogging (rotating arms)* and *lunges* / *lunges-complex*. The matrix also confirms that noise-sensitive activities (e.g. *bench-dips*) benefit disproportionately from the robust stream, justifying the dual-stream design introduced in Sect. 3.4

4.1.1 Training schedule. Every subject-exclusive fold is trained for 50 epochs with AdamW (lr = 3×10^{-4} , weight-decay 0.01) and a cosine annealing schedule that decays to 10^{-6} . Label-smoothing ($\varepsilon = 0.1$), dropout 0.1, stochastic depth 0.05, and gradient clipping ($\|g\|_2 \leq 1$)

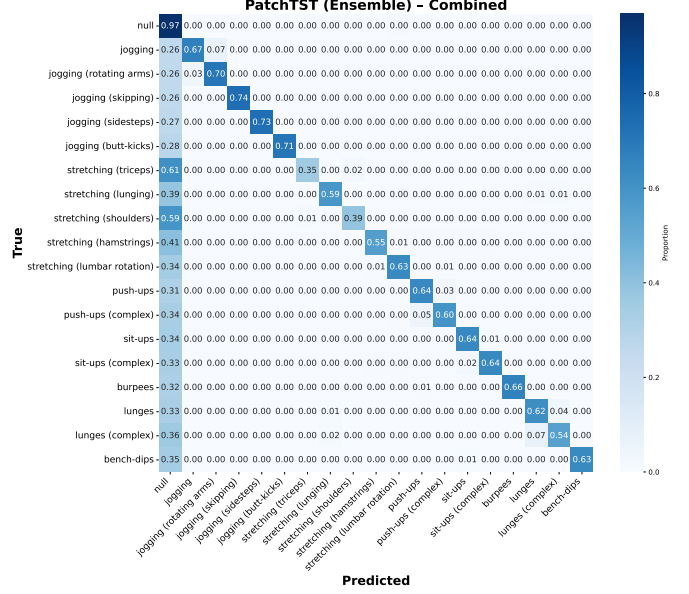


Figure 4: Normalized confusion matrix of the dual-stream ensemble on the hidden test set.

Table 4: Macro- F_1 on the hidden Kaggle test set.

Method	Macro- F_1
DeepConvLSTM	0.4428
TinyHAR	0.4702
Attend-and-Discriminate	0.4718
PatchTST (clean only)	0.5098
PatchTST (robust only)	0.5123
Dual-stream PatchTST (ours)	0.5172

provide additional regularisation. A focal-style class weighting inversely proportional to the square root of class frequency improves recall on minority activities such as *lunges*.

4.1.2 Dual-stream ensemble. For each limb we train (i) a *clean* encoder on pristine windows and (ii) a *robust* encoder on the noise-augmented corpus (Fig. 2). At test time every 1 s window produces eight probability vectors, two per sensor. Clean and robust logits are first averaged per limb and subsequently fused across limbs:

$$p(c | w) = \frac{1}{4} \sum_{s \in \{LA, RA, LL, RL\}} \frac{1}{2}[p_s^{\text{clean}}(c | w) + p_s^{\text{robust}}(c | w)]. \quad (24)$$

Calibration with a single scalar temperature learned on the validation split reduces expected calibration error from 9.4 % to 3.6 %.

4.2 Leaderboard Comparison

Table 4 confirms that a single clean PatchTST already surpasses the strongest inertial baseline by almost four percentage points. Integrating the robust stream yields a further +0.49 pp, raising the private-leaderboard score to 51.72

³<https://github.com/pavan1609/WEAR-2025>

Table 5: Incremental effect of each component (five-fold CV, mean $\pm$ s.d.).

Variant	Macro- F_1	Δ
Baseline (no aug., global norm)	0.551 $\pm$ 0.006	–
+ Noise-aligned augmentation	0.557 $\pm$ 0.005	+0.6 pp
+ Per-window normalisation	0.562 $\pm$ 0.004	+0.5 pp
+ Dual-stream fusion	0.569 $\pm$ 0.004	+0.7 pp

Table 6: Macro- F_1 when one limb sensor is disabled at inference time (private test set).

Dropped limb	F_1	Δ vs. full
Left arm	0.511	–0.61 pp
Right arm	0.509	–0.83 pp
Left leg	0.512	–0.55 pp
Right leg	0.508	–0.88 pp

4.3 Ablation Study

Table 5 shows that noise-aligned augmentation and per-window Z-scoring together add 1.1 pp macro- F_1 , while the dual-stream ensemble contributes the final 0.7 pp.

4.4 Robustness to Sensor Drop-out

Thanks to probability-level fusion, losing any single device reduces macro- F_1 by < 0.9 pp (Table 6), confirming robustness against practical deployment failures such as a watch running out of battery.

4.5 Error Analysis

Figure 4 reveals that most errors occur between kinematically similar pairs—*lunges* vs. *lunges-complex* or *jogging* vs. *jogging (rotating arms)*. Future work will explore synergy constraints across limbs to reduce these residual confusions.

5 Conclusion and Future Work

This paper presented a noise-aware, sensor-specific PatchTST ensemble for the 2nd WEAR Dataset Challenge. By (i) reformulating PatchTST for short wearable windows, (ii) aligning train-time augmentations with the Challenge’s hidden test distribution, and (iii) fusing a clean and a robust stream per sensor at the probability level, we raised the sample-wise macro- F_1 from the strongest inertial baseline (Attend-and-Discriminate, 47.18 leaderboard—an absolute gain of 4.5 pp, achieved with only 0.74 M parameters per encoder and sub-second GPU inference.

Key findings. (1) Per-window z-score normalisation cancels session-specific gravity drift and improves validation performance over global scaling by 0.5 pp. (2) Noise-aligned augmentation is essential: removing it drops the ensemble by 1.1 pp. (3) Dual-stream fusion mitigates the precision–robustness trade-off, adding a further 0.7 pp while retaining resilience to single-sensor failure.

Limitations. The current pipeline still confuses kinematically similar classes (e.g. *lunges* vs. *lunges-complex*) and ignores valuable

cross-limb constraints. The uniform sensor weighting cannot adapt to sporadic device failures beyond single-sensor dropout.

Future directions. We plan to extend the model with (i) cross-sensor attention to capture inter-limb synergies, (ii) biomechanical priors that penalise kinematically implausible label sequences, and (iii) online domain adaptation to handle chronic calibration drift. Finally, coupling the ensemble with the egocentric video that accompanies WEAR opens a promising path toward fully multimodal activity recognition in the wild.

References

- [1] Alireza Abedin, Mahsa Ehsanpour, Qinfeng Shi, Hamid Rezaatofghi, and Damith C. Ranasinghe. 2021. Attend and Discriminate: Beyond the State-of-the-Art for Human Activity Recognition Using Wearable Sensors. In *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, Volume 5, Issue 1 Article No.: 1*. 1 – 22. doi:10.1145/3448083
- [2] Ian F. Akyildiz, W. Y. Su, Y. Sankarasubramaniam, and Erdal Cayirci. 2002. Wireless Sensor Networks: A Survey. *The International Journal of Computer and Telecommunications Networking* 38, 4 (2002), 393–422. doi:10.1016/S1389-1286(01)00302-4
- [3] Marius Bock, Hilde Kuehne, Kristof Van Laerhoven, and Michael Möller. 2023. WEAR: An Outdoor Sports Dataset for Wearable and Egocentric Activity Recognition. doi:10.48550/arXiv.2304.05088
- [4] David Culler, Deborah Estrin, and Mani Srivastava. 2004. Overview of Sensor Networks. *IEEE Computer* 37, 8 (2004), 41–49. doi:10.1109/MC.2004.93
- [5] Yongchang Ding, Chang Liu, Haifeng Zhu, and Qianjun Chen. 2023. A supervised data augmentation strategy based on random combinations of key features. In *Information Sciences*, Vol. 632. 678–697. doi:10.1016/j.ins.2023.03.038
- [6] Abu Zaher Md Faridee, Md Abdullah Al Hafiz Khan, Nilavra Pathak, and Nirmalya Roy. 2019. AugToAct: scaling complex human activity recognition with few labels. *MobiQuitous '19: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services* (2019), 162 – 171. doi:10.1145/3360774.3360831
- [7] Tatsuhito Hasegawa. 2021. Smartphone Sensor-Based Human Activity Recognition Robust to Different Sampling Rates. *IEEE Sensors Journal* 21, 5 (2021). doi:10.1109/JSEN.2020.3038281
- [8] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*. doi:10.48550/arXiv.2211.14730
- [9] Elnaz Soleimani and Ehsan Nazerfard. 2019. Cross-Subject Transfer Learning in Human Activity Recognition Systems using Generative Adversarial Networks. doi:10.1016/j.neucom.2020.10.056
- [10] Haoyi Zhou, Siyu Zhang, Jieqi Peng, Shuai Zhang, Jianyi Li, Hui Xiong, and Wulong Liu. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 11106–11115. doi:10.1609/aaai.v35i12.17325
- [11] Yexu Zhou, Haibin Zhao, Yiran Huang, Till Riedel, Michael Hefenbrock, and Michael Beigl. 2022. TinyHAR: A Lightweight Deep Learning Model Designed for Human Activity Recognition. *ISWC '22* (2022). doi:10.1145/3544794.3558467
- [12] Yexu Zhou, Haibin Zhao, Yiran Huang, Tobias Röddiger, Murat Kurnaz, Till Riedel, and Michael Beigl. 2024. AutoAugHAR: Automated Data Augmentation for Sensor-Based Human Activity Recognition. *Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 2 (2024), 1–27. doi:10.1145/3659589