Physics-guided contrastive learning for wearable fall detection: towards low-latency on-device inference

Bowen Zuo, Junhui Xie, Siqi Wang, Sihan Liu and Xiaojuan Luo

East China University of Science and Technology, 130 Meilong Rd., Xuhui District, Shanghai, 200237, China

Abstract. Fall detection on wearable edge devices requires high sensitivity within tight latency, memory and energy budgets. We propose PhyCL-Net, a time-domain network without an explicit spectral feature branch. It combines a physics-guided dynamic kernel (PDK), which adapts temporal receptive fields to local signal dynamics, with jerk-aware fall-aware attention (FAA), which emphasises transient segments. A hierarchical dual-view contrastive regulariser is used only during training, and its auxiliary heads are removed for deployment. On SisFall with 12-fold leave-one-subject-out (LOSO) validation, PhyCL-Net achieves $98.21 \pm 0.10\%$ accuracy and $98.16 \pm 0.10\%$ Macro-F1 across five seeds, with 1.049M parameters and 0.15 GFLOPs. Compared with a matched spectral baseline, its median single-thread CPU latency is 31.6% lower (125.99 ms). On the empirical ROC curve, the false-positive rate (FPR) at a 95% true-positive rate (TPR) is 0.72%, but the TPR at 1% FPR is 2.73 percentage points lower than that of the spectral baseline.

Keywords: edge AI, wearable fall detection, physics-inspired deep learning, contrastive learning, real-time inference, resource-constrained deployment

1. Introduction

Falls are a leading cause of injury among older adults, with substantial healthcare costs and effects on quality of life [2]. Wearable sensors allow continuous fall monitoring [26], and surveys have reviewed the detection methods developed over the past decade [19, 20]. Running a detector on a resource-constrained edge device, however, forces a trade-off between detection accuracy and computational cost.

Many methods improve representations by combining time- and frequency-domain information [33], at an extra processing cost that depends on the implementation. False alarms matter as much as missed falls: in clinical monitoring, excessive alarms reduce staff responsiveness [5], so detectors should also be compared at low false-positive rates. From a biomechanical perspective, acceleration and changes in body orientation are the main cues for wearable fall detection [26], which motivates time-domain models that capture these temporal features directly.

We propose PhyCL-Net, a time-domain network for fall detection on edge devices that has no explicit spectral feature branch. On SisFall with 12-fold leave-one-subject-out (LOSO) validation, it achieves $98.21 \pm 0.10\%$ accuracy with 1.049M parameters and 0.15 GFLOPs, and its median CPU latency is 31.6% lower than that of a matched spectral baseline. The cost is a true-positive rate (TPR) at a 1% false-positive rate (FPR) that is 2.73 percentage points lower. The contributions are as follows:

- A time-domain backbone built from two lightweight components: a physics-guided dynamic kernel (PDK), which adapts temporal receptive fields to local signal dynamics, and jerk-aware fall-aware attention (FAA), which combines temporal-difference cues with axis attention to emphasise transient segments.

0009-0005-0546-3498 (B. Zuo); 0009-0008-2322-2390 (J. Xie); 0009-0002-2786-1524 (S. Wang); 0009-0000-9218-8298 (S. Liu); 0009-0003-2778-8299 (X. Luo)
blackwhitez246@gmail.com (B. Zuo); bossnailongz2@gmail.com (J. Xie); wyininge@gmail.com (S. Wang); luoxj@ecust.edu.cn (X. Luo)

Received 2026-01-15 Accepted 2026-11-20 Published 2026-11-20

© Copyright for this article by its authors, published by the Academy of Cognitive and Natural Sciences. This is an Open Access article distributed under the terms of the Creative Commons License Attribution 4.0 International (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

- A hierarchical dual-view contrastive regulariser used only during training; its auxiliary heads are removed afterwards, so it adds no inference cost.
- An evaluation that reports parameters, FLOPs and CPU latency, on a desktop computer and on an embedded board, alongside accuracy.
- A comparison of empirical ROC operating points at common sensitivity and false-positive-rate targets relevant to continuous monitoring.

2. Related work

Wearable fall-detection research includes threshold-based methods and feature-based classifiers [19], and deep learning has been studied extensively for sensor-based activity recognition [24]. Residual convolutional networks [8] and long short-term memory (LSTM) networks [9] are common backbones. Transformer-based models [22] have been adapted to time-series tasks, although their advantage over simpler architectures has been questioned in long-term time-series forecasting [31]. Deep fall-detection methods include a CNN–GRU ensemble [17] and LSTM and CNN-transfer approaches [3]. Accuracy alone does not determine whether a model suits a wearable device; latency, memory, energy and the deployed decision rule also need to be evaluated.

Another line of work improves discrimination through spectral or time–frequency preprocessing. These representations can highlight transient structure and are often paired with increasingly expressive sequence models. On an edge device, however, the extra transforms add memory traffic, implementation effort and runtime variability. This study therefore compares a matched spectral baseline with a lighter time-domain alternative under a common protocol. For the time-domain model, channel attention [11], spatial-channel attention [25] and adaptive kernel selection [16] provide established, low-cost ways of emphasising salient features, and they inform the design of PDK and FAA.

Self-supervised contrastive learning has been effective for learning representations from unlabelled images [4]. Time-series formulations, such as temporal and contextual contrasting [6] and time–frequency consistency [33], have been evaluated on time-series classification and transfer-learning benchmarks. Augmentation strategies for time series [13] and transfer-learning and domain-adaptation methods [14, 27] have improved generalisation under the sensing conditions they were evaluated on. In PhyCL-Net, the contrastive component is used only during training, and cross-dataset transfer is reported as supplementary evidence alongside the main LOSO study.

3. Methodology

We formulate fall detection as window-based binary time-series classification. Each sample (X_i, y_i) consists of an inertial window $X_i \in \mathbb{R}^{C\times T}$ and a label $y_i \in \{0, 1\}$, where 1 denotes a fall. Training minimises the class-weighted cross-entropy (CE) loss together with the training-only contrastive term described in section 3.5.

3.1. Overview of PhyCL-Net

PhyCL-Net is a compact time-domain backbone with a lightweight stem and three stages that progressively downsample the temporal dimension (figure 1). For an input $X \in \mathbb{R}^{B\times 3\times 512}$, the stem applies GhostConv1d (k=5, s=1, p=2), mapping 3 to 48 channels while preserving T=512. Stage 1 operates at (C, T)=(48, 512), Stage 2 at (96, 256) and Stage 3 at (192, 128), each with two PhyCL-Blocks. The transition layers between stages use SeparableConv1d (k=5, s=2, p=2) for downsampling. The head applies AdaptiveAvgPool1d(1), a linear layer (192 to 2) and softmax.

Each PhyCL-Block has two parallel streams: PDK, which adapts temporal receptive fields to local signal dynamics, and FAA, which emphasises high-jerk intervals. Their outputs are combined by symmetric cross-gated fusion with a residual connection and a learnable scale. The lower part of figure 1 shows the internal structure of a single block.

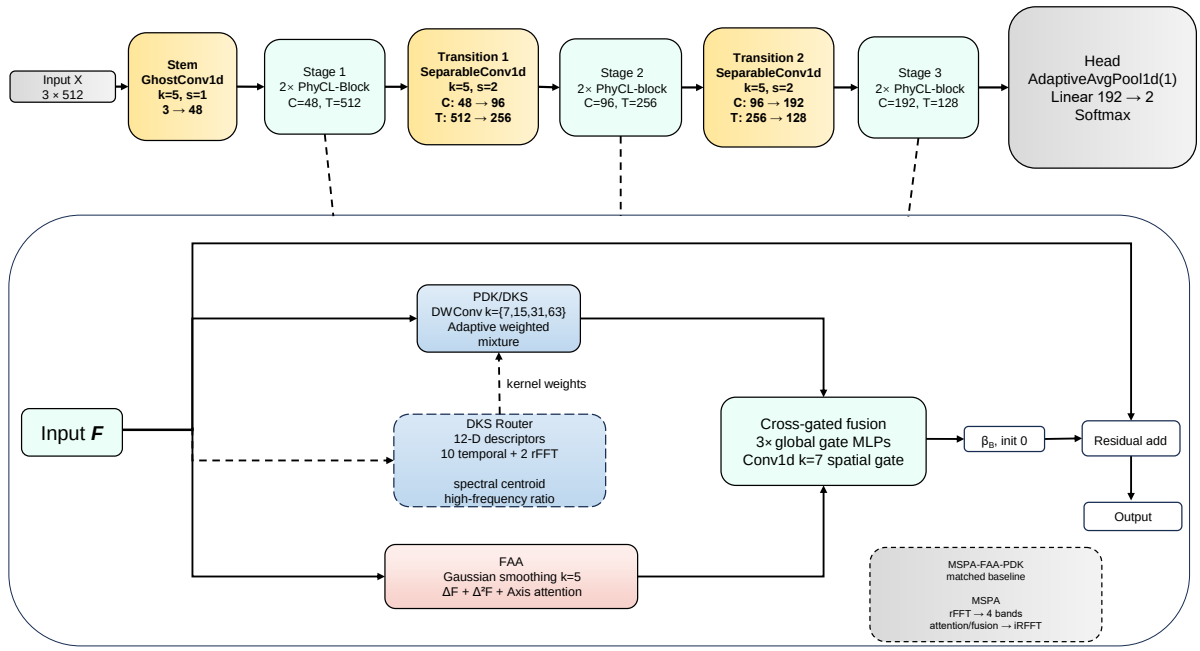


Figure 1: Architecture of PhyCL-Net. Top: the stem, the three stages of PhyCL-Blocks with downsampling transitions, and the classification head. Bottom: a single PhyCL-Block, in which the block input F passes through two parallel streams, PDK for multi-scale temporal adaptation and FAA for transient emphasis, which are combined by the symmetric cross-gated fusion unit. The dashed routing path carries the 10 temporal and two rFFT descriptors that set the PDK kernel weights. The inset summarises the spectral (MSPA) branch of the MSPA-FAA-PDK baseline, which PhyCL-Net does not contain.

3.2. Physics-guided dynamic kernel

Changes in acceleration and orientation call for analysis across multiple temporal scales [26], and a single fixed kernel limits the temporal contexts that can be represented within a small compute budget. Following selective kernel networks [16], PDK passes the features through four depthwise one-dimensional temporal convolutions with kernel sizes $\{7, 15, 31, 63\}$ and combines their outputs with input-dependent mixture weights; we refer to this routing as dynamic kernel selection (DKS). The roughly doubling kernel sizes span short impact transients as well as the longer post-impact evolution, while keeping the number of branches small enough for edge inference.

The mixture weights are predicted from two lightweight signals: (i) a 12-dimensional descriptor $p = \phi(F) \in \mathbb{R}^{12}$ of the block input F, comprising 10 temporal descriptors and two descriptors derived from the real-input fast Fourier transform (rFFT), namely the spectral centroid and the high-frequency energy ratio; and (ii) a four-way complexity logit ℓ^{cmp} predicted from globally average-pooled features. A three-layer physics MLP ($12 \to 32 \to 32 \to 4$) with a final sigmoid gate maps p to $w^{\mathrm{phy}} \in \mathbb{R}^4$, and a two-layer complexity MLP ($C \to 32 \to 4$) produces ℓ^{cmp}. Their element-wise product ($\odot$) is divided by a learnable softmax temperature $\tau_{\mathrm{dks}} = \max(0.1, \tilde{\tau}_{\mathrm{dks}})$, initialised to 1.0, to obtain the routing weights $a = \mathrm{Softmax}((\ell^{\mathrm{cmp}} \odot w^{\mathrm{phy}})/\tau_{\mathrm{dks}})$. The PDK output is

$$\mathrm{PDK}(F) = \sum_{k \in \mathscr{K}} a_k \, \mathrm{DWConv}_k(F), \quad \mathscr{K} = \{7, 15, 31, 63\}, \tag{1}$$

where DWConv_k is the depthwise temporal convolution with kernel size k. Equation (1) omits branch normalisation and the subsequent feature refinement. All feature convolutions operate in the time domain; the two rFFT-derived descriptors only set the kernel weights and do not form a separate frequency-feature branch. Figure 2 shows the module in detail.

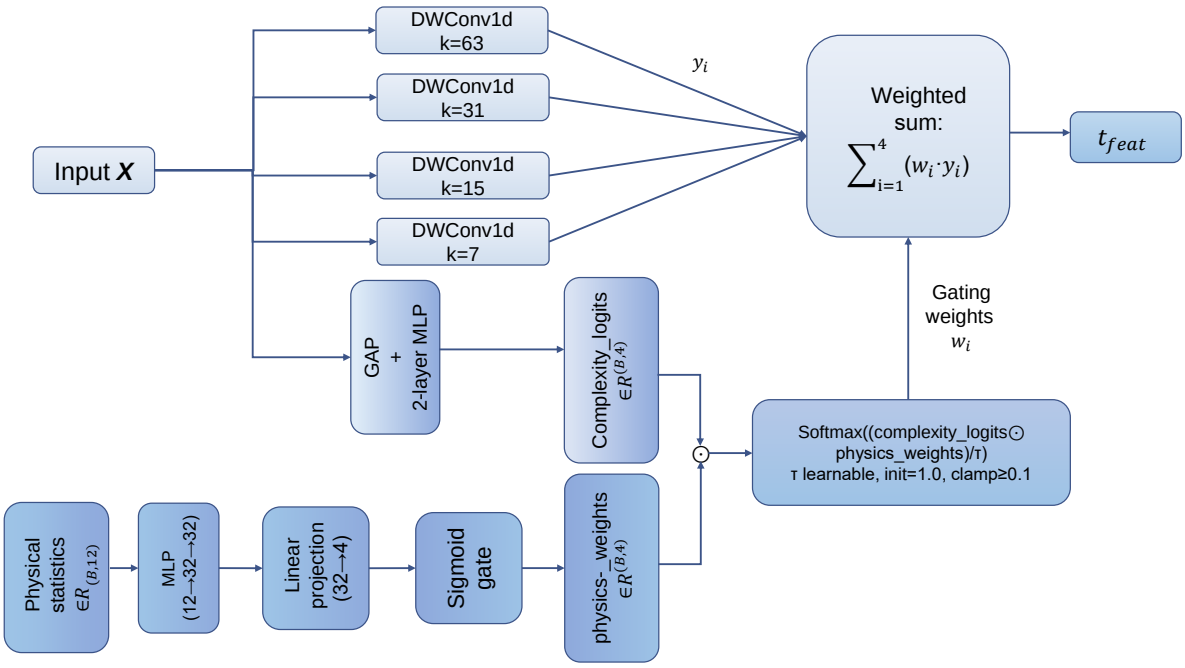


Figure 2: Physics-guided dynamic kernel (PDK) module with dynamic kernel selection (DKS). Features pass through four depthwise temporal kernels, whose outputs are mixed with input-dependent weights derived from the physics descriptors and the complexity logits. The input X in the diagram is the block input F in equation (1).

3.3. Jerk-aware fall-aware attention

Falls can be confused with non-fall activities that involve abrupt motion [30]. FAA therefore combines axis attention, inspired by coordinate attention [10], with Gaussian smoothing and first- and second-order temporal differences. Let $F_a = \mathcal{A}_{\text{axis}}(F)$ and $\bar{F}_a = G_5 * F_a$, where G_5 is a five-tap Gaussian smoothing kernel. The local difference cues are $J_1 = \Delta\bar{F}_a$ and $J_2 = \Delta^2\bar{F}_a$. A learned mask combines these cues with globally pooled context, and the output is, schematically,

$$\text{FAA}(F) = F + \rho_A \,\text{GN}(F_a \odot M(J_1, J_2, F_a)), \tag{2}$$

where M is the learned sigmoid attention mask, ρ_A is a learnable scale and GN denotes group normalisation [28]. The mask is computed from local depthwise convolutions of the absolute first and second differences and from an MLP-based global context path. Because the differences are taken on learned features, they are not calibrated physical jerk measurements.

3.4. Symmetric cross-gated fusion

PDK and FAA are intended to provide multi-scale temporal context and transient cues, respectively. Let $t = \text{PDK}(F)$ and $f = \text{FAA}(F)$. Each stream modulates the other through three global gate MLPs applied to temporally pooled features. With $\mathcal{G}_{t,j}$ and $\mathcal{G}_{f,j}$ denoting the gate logits and GAP$(\cdot)$ denoting global average pooling over time (AdaptiveAvgPool1d(1)), $g_t = \sigma(\frac{1}{3}\sum_{j=1}^{3}\mathcal{G}_{t,j}(\text{GAP}(f)))$ and $g_f = \sigma(\frac{1}{3}\sum_{j=1}^{3}\mathcal{G}_{f,j}(\text{GAP}(t)))$. The gated features $\tilde{F} = t \odot g_t + f \odot g_f$ are weighted by the spatial attention $a_s = \sigma(\text{Conv}_7^{\text{sp}}([\text{AvgPool}_C(\tilde{F}); \text{MaxPool}_C(\tilde{F})]))$, where the subscript C denotes pooling over channels and $\text{Conv}_7^{\text{sp}}$ is a convolution with kernel size 7. Denoting the normalised, projected fusion output, including its internal residual refinement, by $\mathcal{R}(\tilde{F} \odot a_s, t)$, the block output is

$$F_{\text{out}} = F + \beta_B \,\mathcal{R}(\tilde{F} \odot a_s, t), \tag{3}$$

where the block-level residual bypasses both streams and its learnable scale β_B is initialised to 0 (figure 1). The fusion is symmetric in that each stream gates the other; this does not imply that the two gating directions have identical parameters.

3.5. Training-only hierarchical dual-view contrastive regularisation

To improve feature separability without adding deployment cost, we add a training-only contrastive term based on the InfoNCE objective [4, 33], denoted TFCL in the tables. Six auxiliary projection heads (three stages × two streams) are attached during training, each consisting of global average pooling followed by a linear layer that maps to 128 dimensions. The heads are discarded after training, so the inference cost is unchanged.

For a batch of N input windows, two independent stochastic augmentations $t_1(\cdot)$ and $t_2(\cdot)$ [13] generate paired views. For each sample X_i, the backbone f_θ and a projection head h produce ℓ_2-normalised embeddings

$$z_i^{(q)} = \frac{h(f_\theta(t_q(X_i)))}{\|h(f_\theta(t_q(X_i)))\|_2}, \quad q \in \{1,2\}. \tag{4}$$

The loss aligns the embeddings of the same sample and separates those of different samples. We use a symmetric InfoNCE formulation with cosine similarity $\mathrm{sim}(u,v) = u^\top v$ and temperature $\tau = 0.1$:

$$\mathscr{L}_{\mathrm{NCE}} = -\frac{1}{2N}\sum_{i=1}^{N}\left[\ell_i^{(1\to 2)} + \ell_i^{(2\to 1)}\right], \tag{5}$$

where

$$\ell_i^{(q\to q')} = \log \frac{\exp\left(\mathrm{sim}(z_i^{(q)}, z_i^{(q')})/\tau\right)}{\sum_{j=1}^{N}\exp\left(\mathrm{sim}(z_i^{(q)}, z_j^{(q')})/\tau\right)}$$

is the log-softmax probability of the positive match among the candidate samples in the other view. The loss is computed for each of the six stage/stream heads, and the six values are averaged to give the hierarchical contrastive term. The final objective is

$$\mathscr{L} = \mathscr{L}_{\mathrm{CE}} + \lambda_{\mathrm{tfcl}}\,\mathscr{L}_{\mathrm{NCE}}, \quad \lambda_{\mathrm{tfcl}} = 0.1. \tag{6}$$

3.6. Reproducibility and deployment protocol

PhyCL-Net and the spectral baseline share one implementation and differ only in a single configuration flag. Each run writes machine-readable outputs: aggregated metrics, fold-level LOSO results, efficiency measurements and training logs. FLOPs and parameter counts are computed on the CPU for an input of shape $1 \times 3 \times 512$. Latency is measured on the CPU in single-thread mode and reported as the median (p50) and 95th percentile (p95) at batch sizes 1 and 32, together with the runtime environment. For the board benchmark, export scripts generate a TorchScript package, prepare benchmark windows, and record the board metadata, runtime backend, input shape, warm-up and repeat counts, and p50/p95 latency in JSON format. The code and documentation are available in the public repository listed in the data availability statement.

4. Experiments and results

This section describes the evaluation protocol and reports the comparative, efficiency, transfer, hardware and ablation results.

4.1. Dataset and preprocessing

Evaluation uses the SisFall dataset [21], restricted to 12 of its young adult participants (SA01, SA02, SA04, SA05, SA06, SA09, SA10, SA11, SA17, SA18, SA19 and SA21). The raw inertial signals, sampled at 200 Hz, are downsampled to 50 Hz, low-pass filtered (fourth-order Butterworth, 5 Hz cut-off), segmented into windows of 512 samples with a stride of 256 (50% overlap), z-score normalised per channel, and zero-padded when shorter than 512 samples. Unless stated otherwise, only the three accelerometer channels ('accel3') are used. The resulting evaluation set contains 21,678 windows: 12,678 of activities of daily living (ADL) and 9,000 of falls.

4.2. Evaluation protocol and metrics

We use 12-fold LOSO cross-validation. To avoid reporting a single favourable training run, PhyCL-Net is trained with five random seeds (42, 123, 456, 789 and 1024), and we report the mean, standard deviation and 95% confidence interval.

Performance is measured by accuracy, sensitivity (TPR), specificity (true-negative rate, TNR) and Macro-F1, the unweighted mean of the class-wise F1 scores:

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}, \qquad \text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \qquad \text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}, \tag{7}$$

where TP, TN, FP and FN are the numbers of true positives, true negatives, false positives and false negatives, with falls as the positive class.

Because continuous monitoring is sensitive to false alarms [5], we also compare discrimination at common reference constraints: FPR limits of 1% and 5% and a sensitivity target of 95%. ROC analysis describes the trade-off between sensitivity and false positives across score thresholds [7]; the constraints are comparison targets, not clinical alarm limits. For each model and each of two seeds, the continuous fall-class scores and labels from the 12 held-out LOSO folds are pooled into one empirical ROC curve. TPR@FPR$\leq a$ is the largest observed TPR with FPR$\leq a$, and FPR@TPR$\geq b$ is the smallest observed FPR with TPR$\geq b$. Tied scores share a threshold, no interpolation is applied, the two seed-level values are averaged, and the same rules apply to every model. This two-seed ROC comparison is separate from the five-seed classification results.

These values describe the discrimination of the held-out scores; they are not results at a deployed decision threshold. For deployment, an application-specific threshold should be calibrated on validation subjects, frozen, and then evaluated for the achieved TPR and FPR on independent test subjects. The supplementary paired FAA comparison in section 4.10 uses a fixed threshold, predicting a fall ($\hat{y} = 1$) when the fall-class probability is at least 0.5.

4.3. Training setup

Table 1 lists the training settings, which are used for all SisFall LOSO experiments unless stated otherwise.

4.4. Baselines

We compare PhyCL-Net with our implementations of representative time-series architectures: InceptionTime [12], a temporal convolutional network (TCN), a lightweight Transformer [22], ResNet [8] and LSTM [9]. We also include MSPA-FAA-PDK, an in-house baseline that adds a spectral branch (MSPA) to the same backbone; the branch splits the rFFT of the features into four bands, applies attention and fusion, and returns to the time domain by an inverse rFFT (inset in figure 1). Because PhyCL-Net is this model without the MSPA branch, the comparison shows the accuracy and computational cost of the spectral branch.

Table 1
Training hyperparameters for the SisFall LOSO experiments.

Setting	Value
Optimiser	AdamW
Initial learning rate	0.004
Epochs	50
Warm-up epochs	10
Batch size	256
Weight decay	1×10^{-4}
Scheduler	CosineAnnealingLR ($\eta_{\min}=1\times10^{-6}$)
Automatic mixed precision (AMP)	Enabled
Gradient clipping (maximum norm)	1.0
Loss weights	CE: 1.0, TFCL: 0.1
Class weighting	Automatic (inverse frequency)

4.5. Main LOSO results on SisFall

Table 2 summarises the LOSO results. Across five seeds, PhyCL-Net achieves $98.21 \pm 0.10\%$ accuracy and $98.16 \pm 0.10\%$ Macro-F1, the highest means among the evaluated models. The margins over the closest models are small: 0.17 percentage points over the spectral baseline MSPA-FAA-PDK (98.04%) and 0.30 points over InceptionTime (97.91%), whereas TCN, the Transformer, ResNet and LSTM are 1.1–3.2 points lower. The baselines were run with two seeds, and a fold-level Wilcoxon test did not detect an accuracy difference between PhyCL-Net and MSPA-FAA-PDK ($p=0.47$); this is not evidence of equivalence. Figure 3 relates accuracy to model size and latency. PhyCL-Net is not the smallest model: InceptionTime comes within 0.30 points with about 4% of its parameters (0.041M vs 1.049M).

Table 2
LOSO performance on SisFall (12 folds). PhyCL-Net values are means over five seeds, with 95% confidence intervals in parentheses; the other models were run with two seeds and are shown as means.

Model	Parameters	Accuracy (%)	Macro-F1 (%)	Sensitivity (%)	Specificity (%)
PhyCL-Net (ours)	**1.049M**	**98.21** (98.09–98.33)	**98.16** (98.04–98.28)	**97.93** (97.80–98.06)	**98.41** (98.27–98.55)
MSPA-FAA-PDK	1.657M	98.04	97.98	97.67	98.30
InceptionTime	0.041M	97.91	97.85	97.82	97.97
TCN	0.101M	97.13	97.04	96.43	97.63
Transformer	0.200M	95.48	95.34	94.71	96.02
ResNet	0.014M	95.13	94.98	94.41	95.64
LSTM	0.532M	95.02	94.86	94.35	95.50

Figure 4 shows the accuracy of each LOSO fold for PhyCL-Net and the baseline.

Figures 5–7 and table 3 give further views of the results. The radar chart (figure 5) compares PhyCL-Net, MSPA-FAA-PDK and TCN on five metrics. The t-SNE projection (figure 6) illustrates class separation in the learned feature space, and the FAA visualisation (figure 7) shows the temporal regions emphasised around an impact transient. Table 3 reports the mean conditional rates for the two classes, ADL (non-fall) and fall.

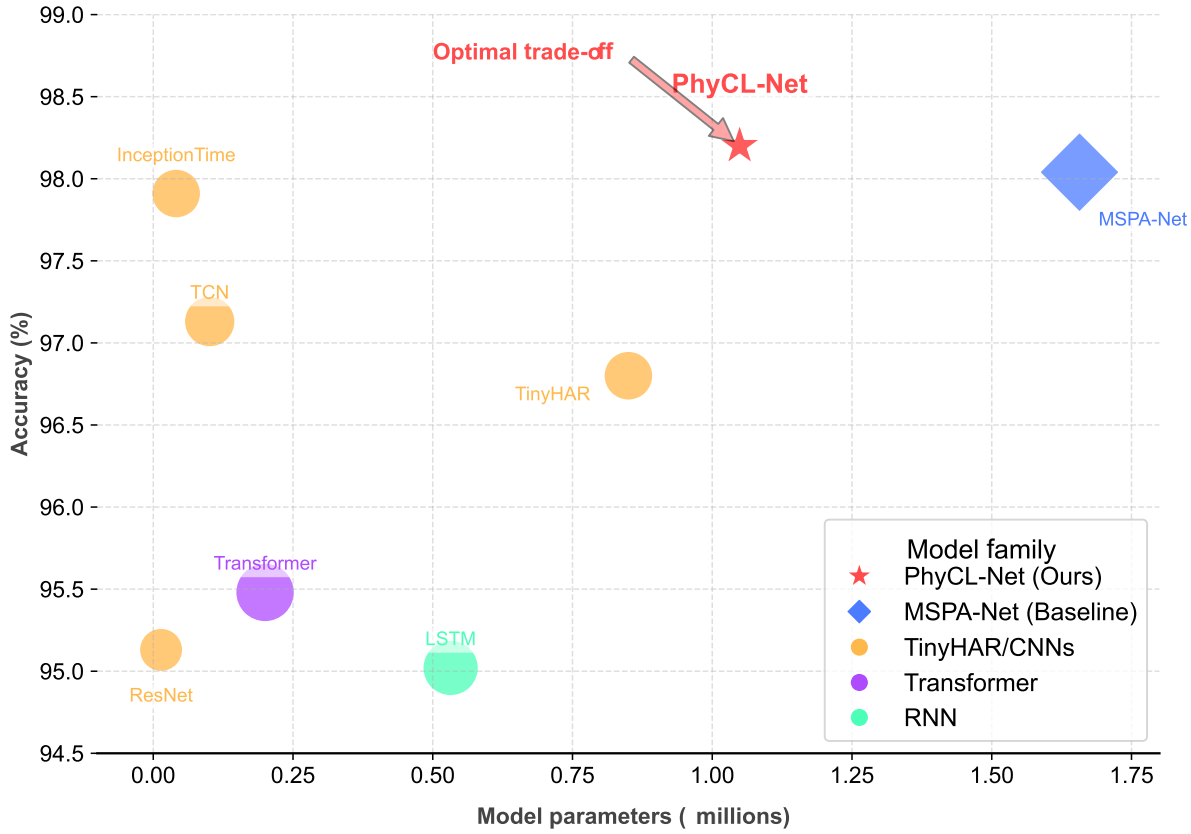


Figure 3: Accuracy versus number of parameters. Bubble size is proportional to single-thread CPU latency (p50). PhyCL-Net (red star) has the highest accuracy (98.21%) among the models in table 2, with 1.049M parameters and a median latency of 125.99 ms. MSPA-Net in the plot denotes MSPA-FAA-PDK. The TinyHAR point is not part of the comparisons in this paper.

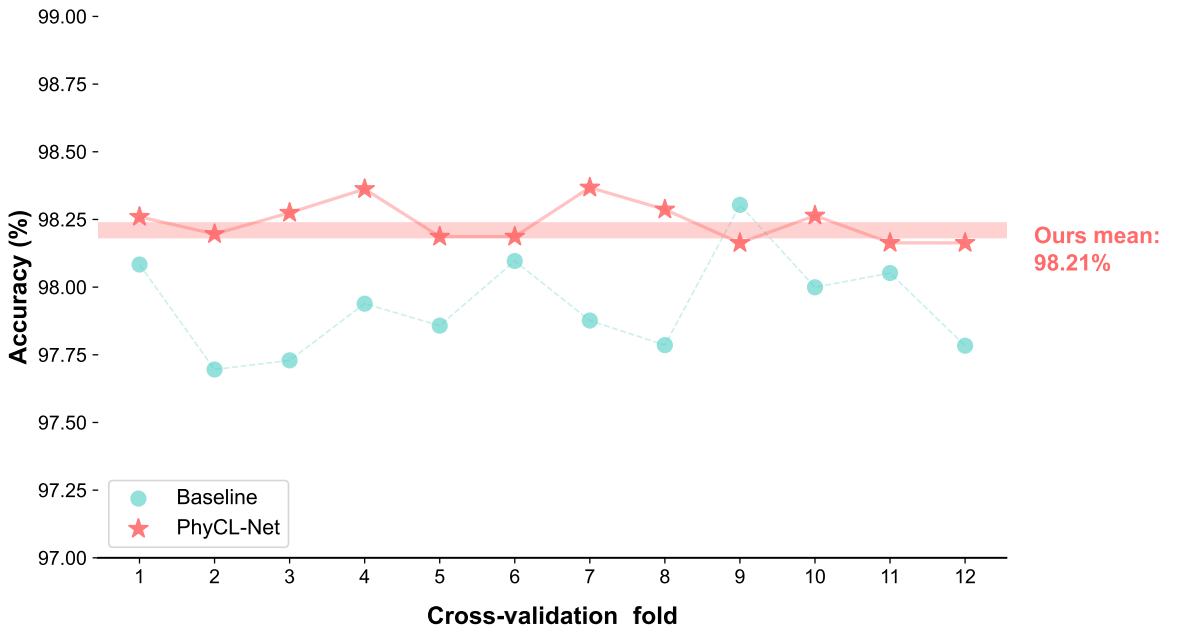


Figure 4: Fold-wise LOSO accuracy across the 12 held-out subjects. The comparison is descriptive; it does not test for lower variance or statistical superiority.

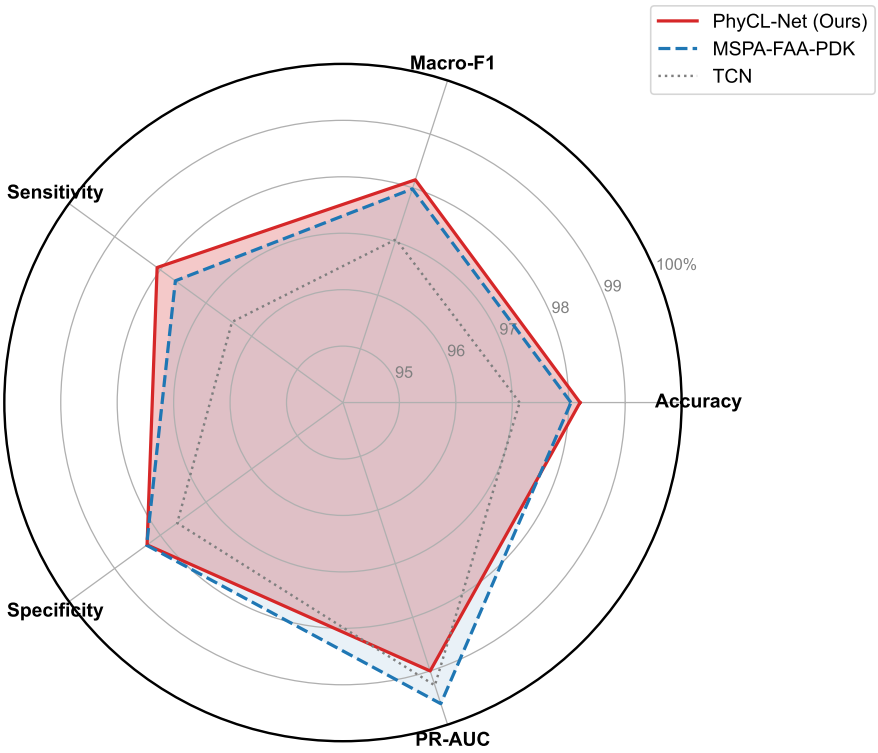


Figure 5: Radar chart comparing PhyCL-Net, MSPA-FAA-PDK and TCN on five metrics, including the area under the precision–recall curve (PR-AUC). The radial axis runs from 94% to 100% to make the differences visible.

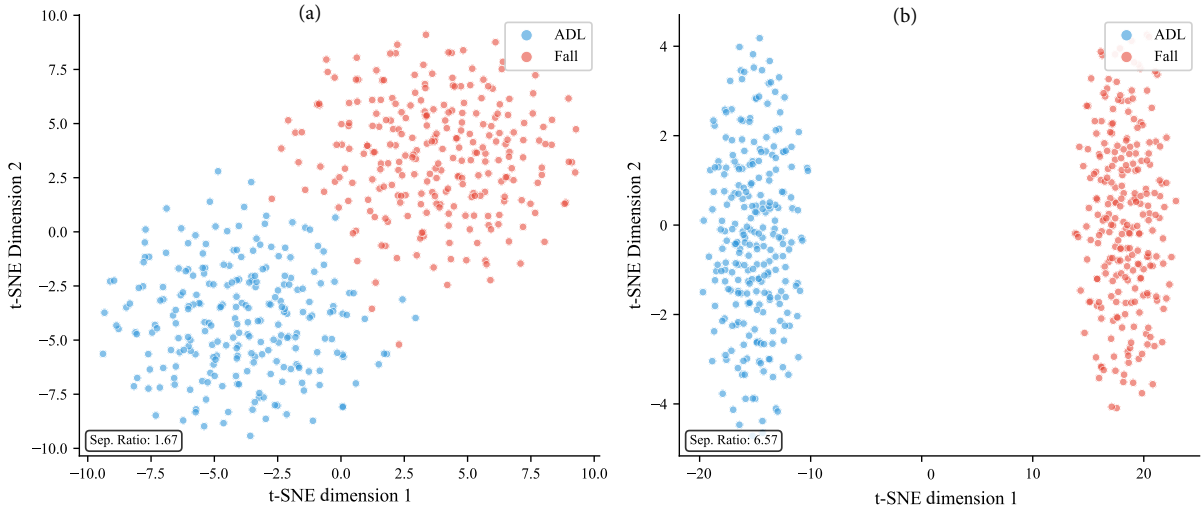


Figure 6: t-SNE visualisation of the learned embeddings of (a) the baseline model and (b) PhyCL-Net. Fall and ADL samples form separate clusters in both panels; the separation ratio shown in each panel is 1.67 for the baseline and 6.57 for PhyCL-Net.

Table 3
Row-normalised binary summary for the main SisFall experiment. Rows are true classes and columns are predicted classes. The diagonal entries are the mean specificity and sensitivity in table 2, and the off-diagonal entries are their complements. These are mean conditional percentages, not pooled prediction counts.

True class	Predicted ADL (non-fall)	Predicted fall
ADL (non-fall)	98.41%	1.59%
Fall	2.07%	97.93%

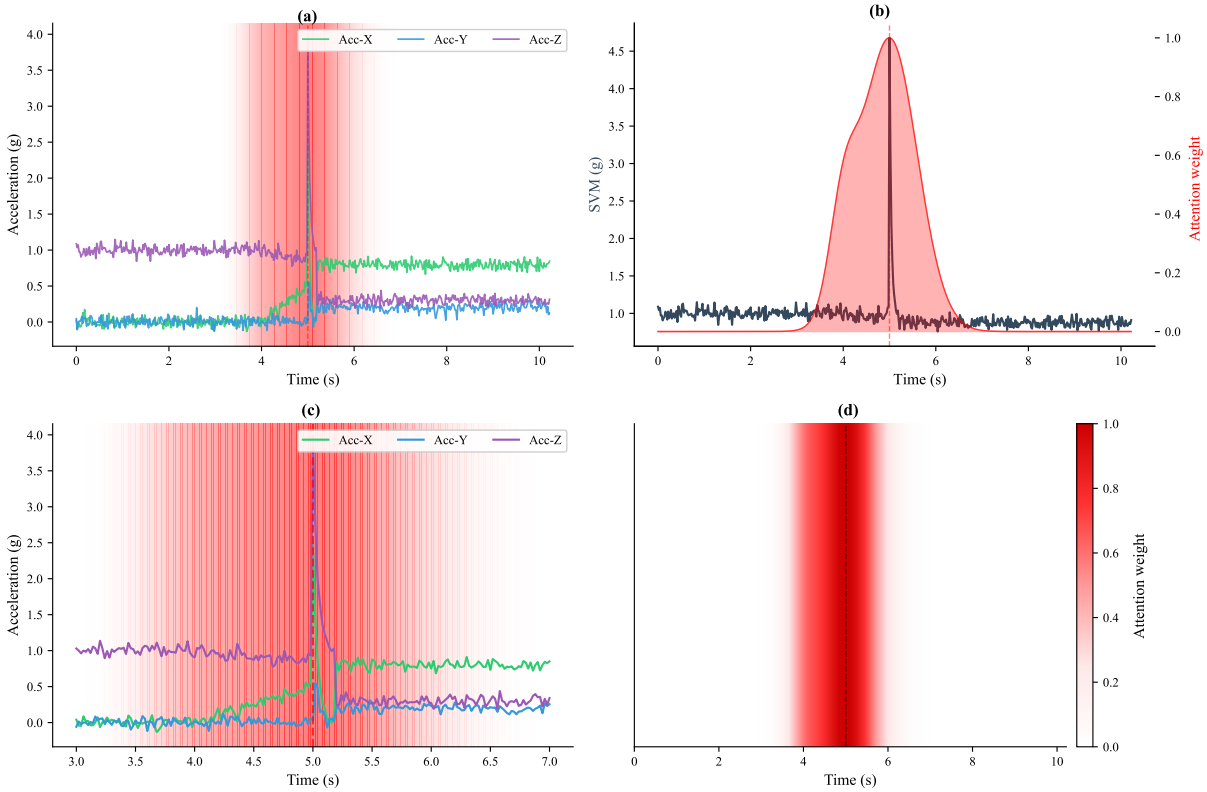


Figure 7: Visualisation of the fall-aware attention module for a fall window: (a) full signal with attention overlay; (b) signal vector magnitude (SVM) and attention weight; (c) zoomed view (±2 s around the peak); (d) attention weight over time. The display shows temporal emphasis near the impact transient; it does not by itself establish physical jerk alignment or a causal contribution to performance.

4.6. Computational efficiency

PhyCL-Net has 1.049M parameters and requires 0.150979 GFLOPs (measured with fvcore). With single-thread CPU execution (PyTorch 2.5.1, Python 3.10.11, Windows 10) and a batch size of 1, its latency is 125.99 ms at p50 and 141.43 ms at p95, compared with 184.31 ms and 203.27 ms for MSPA-FAA-PDK, a 31.6% reduction in p50 latency (table 4).

Table 4
Model complexity and single-thread CPU latency (batch size 1).

Model	Parameters (M)	FLOPs (G)	Latency p50 (ms)	Latency p95 (ms)
MSPA-FAA-PDK (baseline)	1.657	0.202923	184.31	203.27
PhyCL-Net (ours)	**1.049**	**0.150979**	**125.99**	**141.43**

4.7. Cross-dataset transfer

As supplementary evidence, we evaluated transfer to three external datasets, MobiFall v2.0 [23], UniMiB SHAR [18] and KFall [29], under a separate protocol with six input channels and windows of length 200. Table 5 reports zero-shot transfer and transfer with 5-shot adaptation. In the zero-shot setting, Macro-F1 ranges from 78.77% to 82.85% and TPR@FPR≤1% from 48.24% to 55.72%. With 5-shot adaptation, Macro-F1 rises to 93.52–96.32% and TPR@FPR≤1% to 86.15–90.43%. The learned representation therefore retains useful structure across datasets. These results are not directly comparable with the main study, whose protocol (three channels, windows of 512 samples, LOSO) differs, and the stronger results depend on target-side adaptation.

Table 5
Supplementary cross-dataset transfer results (six channels, window length 200).

Target dataset	Method	Accuracy (%)	Macro-F1 (%)	Sensitivity (%)	Specificity (%)	TPR@FPR≤1% (%)
MobiFall	Zero-shot	82.47	82.85	83.56	81.38	55.72
MobiFall	Transfer (5-shot)	96.21	96.32	97.22	95.21	90.43
UniMiB	Zero-shot	78.53	78.77	80.28	76.85	48.24
UniMiB	Transfer (5-shot)	93.57	93.52	94.90	92.30	86.15
KFall	Zero-shot	79.64	79.80	81.46	77.82	50.19
KFall	Transfer (5-shot)	94.38	94.61	95.81	92.95	87.27

4.8. Board-level and prototype benchmarks

To complement the desktop measurements, we exported a trained PhyCL-Net checkpoint to TorchScript and ran it on the CPU of an Orange Pi AI Pro 20T 24G board (Ubuntu 22.04.5 LTS on aarch64, Python 3.9.2, PyTorch 2.1.0, four CPU cores). The benchmark used the input shape 1×3×512, 50 warm-up runs and 200 timed repeats, once with a fixed-shape synthetic input and once with 32 prepared windows from the exported benchmark set. Table 6 reports the p50/p95 latency, the peak resident set size (RSS) and the model size. This is a board-level CPU benchmark, not an evaluation on a commercial wearable or clinical device.

Table 6
CPU benchmark of the exported TorchScript model on an Orange Pi AI Pro 20T 24G.

Benchmark	Backend	Mode	p50 (ms)	p95 (ms)	Peak RSS (MB)	Model size (MB)
Fixed shape 1×3×512	TorchScript	CPU	622.78	669.91	255.25	5.80
Prepared benchmark windows	TorchScript	CPU	619.19	670.92	255.95	5.80

The median latency of about 620 ms and the peak RSS of about 255 MB show that the exported model runs on an embedded CPU, but also that this board needs platform-specific runtime optimisation.

Table 7 summarises measurements with and without FAA on two further prototypes: a Raspberry Pi Zero 2 W with an ADXL345 accelerometer and an Apollo4 Blue Plus with a BMI270 inertial sensor. With FAA, the p95 latency is 78.6 ms for the software path on the Pi and 94.7 ms for the processing path on Apollo4. Because the timed boundaries differ from those of the Orange Pi benchmark, these values do not establish a cross-platform speed-up or the latency from fall onset to alert. The power values are mean board power, not energy per decision, and battery lifetime was not evaluated.

Table 7
Prototype measurements with and without FAA on two platforms. Timing boundaries and memory measures differ between the platforms; the power entries are mean board power, not per-window energy.

Platform	Metric	FAA	No FAA
Raspberry Pi Zero 2 W	Software-path p95 (ms)	78.6	71.2
	Software-path p99 (ms)	96.4	89.3
	Maximum over 24 h (ms)	118.2	110.5
	Peak RSS (MB)	163.8	151.3
	Mean board power (mW)	412	386
Apollo4 Blue Plus	Processing p95 (ms)	94.7	86.2
	Processing p99 (ms)	112.3	104.6
	Maximum over 72 h (ms)	126.4	119.8
	Peak SRAM, including tensor arena (KB)	1,865	1,804
	Non-volatile storage (KB)	1,042	812
	Mean board power (mW)	28.4	26.7

In long-duration runs, the Pi processed 67,500 windows over 24 h in 12 two-hour intervals, and the second Apollo4 board, with FAA, processed 202,500 windows over 72 h in six twelve-hour intervals; no interval recorded a deadline violation. These counts reflect the device workload, not the LOSO windowing protocol, and they are not battery-endurance tests. Across three Apollo4 boards with three sessions each, the session p95 ranged from 94.1 to 95.1 ms, the mean board power from 28.2 to 28.6 mW and the peak SRAM from 1,857 to 1,868 KB. The FAA setting was not recorded for the Pi intervals or for these nine sessions. The values are session summaries rather than per-window logs, so their ranges do not estimate a pooled percentile or a paired FAA effect.

4.9. Constraint-based ROC operating-point analysis

Table 8 compares the models under the common empirical ROC constraints. At 95% sensitivity, PhyCL-Net has the lowest FPR, 0.72%, compared with 0.82–4.19% for the baselines. At FPR≤1%, however, its TPR (93.29%) is lower than that of MSPA-FAA-PDK (96.02%), InceptionTime (95.52%) and TCN (93.38%); removing the spectral branch costs 2.73 percentage points at this operating point. At FPR≤5%, PhyCL-Net (99.26%) and the spectral baseline (99.28%) are almost equal.

Table 8
Constraint-based empirical ROC summaries, averaged over two seeds per model and computed from observed ROC points without interpolation. They describe discrimination, not performance at a calibrated deployment threshold.

Model	TPR@FPR≤1% (%)	TPR@FPR≤5% (%)	FPR@TPR≥95% (%)
MSPA-FAA-PDK (baseline)	**96.02**	**99.28**	0.82
InceptionTime	95.52	99.11	0.90
PhyCL-Net (ours)	93.29	99.26	**0.72**
TCN	93.38	98.12	1.38
Transformer	82.98	95.86	4.19

4.10. Ablation study

Table 9 summarises the main design variants, and table 10 gives the component-wise ablations; with two seeds per variant, these comparisons are descriptive. Disabling dynamic kernel selection in PDK left accuracy essentially unchanged (98.22 ± 0.08%, n=2) but increased the CPU p50 latency to 213.3 ms; this is a configuration-level observation and does not show that adaptive routing alone reduces latency. Removing TFCL from the spectral baseline gave 97.96 ± 0.03% accuracy (n=2), compared with 98.04% with TFCL; this small difference does not establish an effect of the contrastive term. The variant without FAA reached a similar or slightly higher accuracy (98.24 ± 0.12%, n=2) and a lower latency (120.15 ms, compared with 125.99 ms for the complete model with n=5), so this unequal-seed comparison shows no benefit from FAA.

To examine FAA in a paired design, we ran a separate SisFall evaluation on 12 subjects (SA01–SA12) with the same five seeds (42, 123, 456, 789 and 1024) in both arms and a fixed probability threshold of 0.5. This subject set differs from the main cohort (section 4.1), so these results supplement the ablation above. For each subject, Macro-F1 is averaged over the five seeds; table 11 reports the mean and sample standard deviation (SD) of these 12 subject averages.

The mean paired gain from FAA is 0.63 percentage points, and all 60 paired subject–seed differences are positive. FAA does not reduce latency, however: on Apollo4 it increases the p95 latency by 8.5 ms (9.9%), the mean board power by 1.7 mW (6.4%) and the peak SRAM by 61 KB (3.4%), so its classification benefit has to be weighed against these costs.

Table 9
Ablation summary under SisFall LOSO. Seed counts refer to the classification results; the ROC column uses two seeds.

Variant	Parameters (M)	Accuracy (%)	Macro-F1 (%)	TPR@FPR≤1% (%)	CPU p50 (ms)
MSPA-FAA-PDK (n=2)	1.657	98.04	97.98	**96.02**	184.31
Spectral baseline w/o TFCL (n=2)	1.657	97.96±0.03	97.90±0.03	95.45	181.33
PhyCL-Net (n=5)	**1.049**	**98.21±0.10**	**98.16±0.10**	93.29	**125.99**

Table 10
Component-wise ablation study on SisFall LOSO. Ablations with n=2 seeds are indicative; the main model uses n=5.

Configuration	Accuracy (%)	Macro-F1 (%)	Latency (ms)	Seeds
PhyCL-Net (complete)	**98.21±0.10**	**98.16±0.10**	125.99	n=5
− DKS (fixed kernel)	98.22±0.08	98.18±0.09	213.3	n=2
− FAA (no jerk-aware attention)	98.24±0.12	98.19±0.12	120.15	n=2

Table 11
Supplementary paired FAA comparison on SA01–SA12 at threshold 0.5. The SD is taken over the 12 subject means (each averaged over five seeds), not over seeds.

Configuration	Macro-F1, mean ± subject SD (%)
With FAA	97.73 ± 0.49
Without FAA	97.10 ± 0.59
FAA minus no FAA (percentage points)	+0.63 ± 0.12

5. Discussion

The results show that, on SisFall, a time-domain backbone without an explicit spectral branch can reach the classification accuracy of its spectral counterpart while reducing desktop CPU latency. This section discusses the cost of that trade-off, the interpretation of the design and the limits of the evaluation.

5.1. Latency and operating-point trade-off

Removing the spectral branch reduces single-thread CPU latency by about a third (section 4.6) without a detectable change in LOSO accuracy (section 4.5). The branch still helps at the strict end of the ROC curve: without it, TPR at FPR≤1% is lower, although PhyCL-Net has the lowest FPR at 95% sensitivity (section 4.9). Which operating point matters depends on the alert policy, and translating either value into an alarm burden for always-on monitoring requires a calibrated threshold (section 4.2). The ranking in table 2 also does not show which architectural component, such as recurrence or attention, causes the differences in accuracy.

5.2. Interpreting the time-domain design

Falls are typically characterised by a brief impact transient and a short transition phase, whereas many ADLs have a more regular temporal structure. PDK is designed to capture this with multi-scale receptive fields that adjust to local dynamics within a lightweight computation graph, and FAA to emphasise jerk-dominant segments. The ablations support this reading only in part: a fixed kernel gave the same accuracy as dynamic kernel selection, and FAA showed a modest benefit only in the supplementary paired comparison. The physics-inspired components are therefore best seen as an inductive bias that is compatible with edge constraints, not as a demonstrated mechanism.

5.3. Robustness and practical considerations

To assess robustness to sensor noise, additive white Gaussian noise was injected into the accelerometer signals at signal-to-noise ratios (SNRs) from 40 to 5 dB. Signal power was defined as the variance of the demeaned signal, so that gravity does not bias the SNR. On a held-out subject (SA01), with five repetitions per SNR, accuracy remained close to the clean level at 40 and 35 dB ($98.04 \pm 0.03\%$ and $97.40 \pm 0.07\%$), dropped to $91.49 \pm 0.15\%$ at 30 dB and declined sharply below 25 dB. The SDs describe five noise realisations at each SNR, not variation across subjects or training seeds. This synthetic test shows how performance deteriorates as the SNR decreases; it does not establish a typical SNR range for wearables or separate sensing effects from model limitations.

The experiments use the fixed accelerometer placement of SisFall, so the results apply most directly to similar placements. Rotated axes or different body locations alter jerk profiles and inter-axis correlations; signal vector magnitude cues and multi-scale temporal modelling may reduce this sensitivity, but placement effects were not tested. Apart from the six-channel transfer protocol, the experiments also use only the three accelerometer axes, although the architecture accepts any multichannel inertial window and could in principle take gyroscope inputs.

5.4. Limitations and future directions

The main evidence comes from a 12-subject LOSO study of young adults in SisFall; older adults, the population most at risk, are not represented, and the baselines were run with only two seeds. The transfer results on MobiFall, UniMiB SHAR and KFall use a different protocol and rely on target-side adaptation, so cross-dataset and cross-population behaviour needs further validation [14, 27]. The hardware evidence, the Orange Pi CPU benchmark and the Pi Zero 2 W and Apollo4 prototype measurements, shows execution on these boards only; it is not a comparison with commercial wearables or clinical alarm systems, which differ in sensors, firmware, scheduling and alert logic. Continuous-wear operation, latency from sensing to alert, battery lifetime, motion artefacts and long-term sensor drift remain to be evaluated in an integrated system, including their effect on TPR at low FPR. Possible next steps include model quantisation [32], further embedded optimisation informed by microcontroller-based pre-impact fall detection [1], and graph-based models of inter-channel dependencies [15].

6. Conclusion

This work presented PhyCL-Net, a physics-guided time-domain network for wearable fall detection that combines PDK, FAA, symmetric cross-gated fusion and a training-only hierarchical dual-view contrastive regulariser, without an explicit spectral feature branch. Under 12-fold LOSO validation on SisFall, it achieved $98.21 \pm 0.10\%$ accuracy and $98.16 \pm 0.10\%$ Macro-F1 across five seeds, with 1.049M parameters and 0.15 GFLOPs, and its single-thread CPU p50 latency was 31.6% lower than that of the spectral baseline. The price is a TPR at 1% FPR that is 2.73 percentage points lower, so the choice between the two models depends on the required operating point. The supplementary transfer experiments and board-level benchmarks indicate that the representation transfers to other datasets

with limited adaptation and that the exported model runs on embedded hardware; validation on older adults, on wearable devices and at calibrated thresholds remains to be done.

Author contributions

Conceptualisation, methodology, software, validation, formal analysis, investigation, resources, data curation, writing – original draft, visualisation: Bowen Zuo, Junhui Xie, Siqi Wang, Sihan Liu, Xiaojuan Luo. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Data availability statement

The SisFall [21], MobiFall [23], UniMiB SHAR [18] and KFall [29] datasets are publicly available. The code, reproducibility documentation and the artefacts referenced in this paper, including the Orange Pi CPU benchmark records, are available at https://github.com/AlexZander-666/PhyCL_Net. The datasets, trained checkpoints, the inputs used for full reruns and for the board benchmark, and other generated outputs are not stored in the repository; their preparation and use are described in the reproducibility documentation.

Conflicts of interest

The authors declare no conflict of interest.

Declaration on Generative AI

Semantic Scholar and Google Scholar were used for literature discovery. Claude Opus 4.5 assisted with drafting, structure, and consistency of terminology; Grammarly was used for grammar and spelling. The authors manually verified all technical claims, references, and experimental results. The authors take full responsibility for the content and accuracy of this publication.

References

[1] Benoit, A., Escriba, C., Gauchard, D., Esteve, A. and Rossi, C., 2024. Analyzing and Comparing Deep Learning Models on an ARM 32 Bits Microcontroller for Pre-Impact Fall Detection. *IEEE Sensors Journal*, 24(7), pp.11829–11842. Available from: https://doi.org/10.1109/JSEN.2024.3364249.

[2] Bergen, G., Stevens, M.R. and Burns, E.R., 2016. Falls and Fall Injuries Among Adults Aged ≥65 Years—United States, 2014. *MMWR. Morbidity and Mortality Weekly Report*, 65(37), pp.993–998. Available from: https://doi.org/10.15585/mmwr.mm6537a2.

[3] Butt, A., Narejo, S., Anjum, M.R., Yonus, M.U., Memon, M. and Samejo, A.A., 2022. Fall Detection Using LSTM and Transfer Learning. *Wireless Personal Communications*, 126(2), pp.1733–1750. Available from: https://doi.org/10.1007/s11277-022-09819-3.

[4] Chen, T., Kornblith, S., Norouzi, M. and Hinton, G., 2020. A Simple Framework for Contrastive Learning of Visual Representations. In: H.D. III and A. Singh, eds. *Proceedings of the 37th International Conference on Machine Learning*, *Proceedings of machine learning research*, vol. 119. PMLR, pp.1597–1607. Available from: https://proceedings.mlr.press/v119/chen20j.html.

[5] Cvach, M., 2012. Monitor Alarm Fatigue: An Integrative Review. *Biomedical Instrumentation & Technology*, 46(4), pp.268–277. Available from: https://doi.org/10.2345/0899-8205-46.4.268.

[6] Eldele, E., Ragab, M., Chen, Z., Wu, M., Kwoh, C.K., Li, X. and Guan, C., 2021. Time-Series Representation Learning via Temporal and Contextual Contrasting. In: Z.H. Zhou, ed. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, pp.2352–2359. Available from: https://doi.org/10.24963/ijcai.2021/324.

[7] Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8), pp.861–874. Available from: https://doi.org/10.1016/j.patrec.2005.10.010.

[8] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp.770–778. Available from: https://doi.org/10.1109/CVPR.2016.90.

[9] Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735–1780. Available from: https://doi.org/10.1162/neco.1997.9.8.1735.

[10] Hou, Q., Zhou, D. and Feng, J., 2021. Coordinate Attention for Efficient Mobile Network Design. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp.13708–13717. Available from: https://doi.org/10.1109/CVPR46437.2021.01350.

[11] Hu, J., Shen, L. and Sun, G., 2018. Squeeze-and-Excitation Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, pp.7132–7141. Available from: https://doi.org/10.1109/CVPR.2018.00745.

[12] Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A. and Petitjean, F., 2020. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, 34(6), pp.1936–1962. Available from: https://doi.org/10.1007/s10618-020-00710-y.

[13] Iwana, B.K. and Uchida, S., 2021. An empirical survey of data augmentation for time series classification with neural networks. *PLOS ONE*, 16(7), e0254841. Available from: https://doi.org/10.1371/journal.pone.0254841.

[14] Jia, Q., Guo, J., Yang, P. and Yang, Y., 2024. A holistic multi-source transfer learning approach using wearable sensors for personalized daily activity recognition. *Complex & Intelligent Systems*, 10(1), pp.1459–1471. Available from: https://doi.org/10.1007/s40747-023-01218-w.

[15] Jin, M., Koh, H.Y., Wen, Q., Zambon, D., Alippi, C., Webb, G.I., King, I. and Pan, S., 2024. A Survey on Graph Neural Networks for Time Series: Forecasting, Classification, Imputation, and Anomaly Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12), pp.10466–10485. Available from: https://doi.org/10.1109/TPAMI.2024.3443141.

[16] Li, X., Wang, W., Hu, X. and Yang, J., 2019. Selective Kernel Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp.510–519. Available from: https://doi.org/10.1109/CVPR.2019.00060.

[17] Liu, C.P., Li, J.H., Chu, E.P., Hsieh, C.Y., Liu, K.C., Chan, C.T. and Tsao, Y., 2023. Deep Learning-based Fall Detection Algorithm Using Ensemble Model of Coarse-fine CNN and GRU Networks. *2023 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, pp.1–5. Available from: https://doi.org/10.1109/MeMeA57477.2023.10171944.

[18] Micucci, D., Mobilio, M. and Napoletano, P., 2017. UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data from Smartphones. *Applied Sciences*, 7(10), 1101. Available from: https://doi.org/10.3390/app7101101.

[19] Mubashir, M., Shao, L. and Seed, N.L., 2013. A survey on fall detection: Principles and approaches. *Neurocomputing*, 100, pp.144–152. Available from: https://doi.org/10.1016/j.neucom.2011.09.037.

[20] Ramachandran, A. and Karuppiah, A., 2020. A Survey on Recent Advances in Wearable Fall Detection Systems. *BioMed Research International*, 2020(1), 2167160. Available from: https://doi.org/10.1155/2020/2167160.

[21] Sucerquia, A., López, J.D. and Vargas-Bonilla, J.F., 2017. SisFall: A Fall and Movement Dataset. *Sensors*, 17(1), 198. Available from: https://doi.org/10.3390/s17010198.

[22] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention Is All You Need. In: I. Guyon, U. Von Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc. Available from: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[23] Vavoulas, G., Pediaditis, M., Chatzaki, C., Spanakis, E.G. and Tsiknakis, M., 2014. The MobiFall Dataset: Fall Detection and Classification with a Smartphone. *International Journal of Monitoring and Surveillance Technologies Research*, 2(1), pp.44–56. Available from: https://doi.org/10.4018/ijmstr.2014010103.

[24] Wang, J., Chen, Y., Hao, S., Peng, X. and Hu, L., 2019. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119, pp.3–11. Available from: https://doi.org/10.1016/j.patrec.2018.02.010.

[25] Woo, S., Park, J., Lee, J.Y. and Kweon, I.S., 2018. CBAM: Convolutional Block Attention Module. In: V. Ferrari, M. Hebert, C. Sminchisescu and Y. Weiss, eds. *Computer Vision – ECCV 2018*, *Lecture Notes in Computer Science*, vol. 11211. Springer International Publishing, pp.3–19. Available from: https://doi.org/10.1007/978-3-030-01234-2_1.

[26] Wu, F., Zhao, H., Zhao, Y. and Zhong, H., 2015. Development of a Wearable-Sensor-Based Fall Detection System. *International Journal of Telemedicine and Applications*, 2015(1), 576364. Available from: https://doi.org/10.1155/2015/576364.

[27] Wu, T., Chen, Q., Zhao, D., Wang, J. and Jiang, L., 2024. Domain adaptation of time series via contrastive learning with task-specific consistency. *Applied Intelligence*, 54(23), pp.12576–12588. Available from: https://doi.org/10.1007/s10489-024-05799-y.

[28] Wu, Y. and He, K., 2018. Group Normalization. *Computer Vision – ECCV 2018*, *Lecture Notes in Computer Science*, vol. 11217. Springer International Publishing, pp.3–19. Available from: https://doi.org/10.1007/978-3-030-01261-8_1.

[29] Yu, X., Jang, J. and Xiong, S., 2021. A Large-Scale Open Motion Dataset (KFall) and Benchmark Algorithms for Detecting Pre-impact Fall of the Elderly Using Wearable Inertial Sensors. *Frontiers in Aging Neuroscience*, 13, 692865. Available from: https://doi.org/10.3389/fnagi.2021.692865.

[30] Zafar, R.O. and Zafar, F., 2025. Real-time activity and fall detection using transformer-based deep learning models for elderly care applications. *BMJ Health & Care Informatics*, 32(1), e101439. Available from: https://doi.org/10.1136/bmjhci-2025-101439.

[31] Zeng, A., Chen, M., Zhang, L. and Xu, Q., 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9), pp.11121–11128. Available from: https://doi.org/10.1609/aaai.v37i9.26317.

[32] Zhang, M., Xu, J., Li, W. and Ning, X., 2023. DiffQuant: Reducing Compression Difference for Neural Network Quantization. *Electronics*, 12(24), 4972. Available from: https://doi.org/10.3390/electronics12244972.

[33] Zhang, X., Zhao, Z., Tsiligkaridis, T. and Zitnik, M., 2022. Self-Supervised Contrastive Pre-Training For Time Series via Time-Frequency Consistency. In: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds. *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc., pp.3988–4003. Available from: https://proceedings.neurips.cc/paper_files/paper/2022/hash/194b8dac525581c346e30a2cebe9a369-Abstract-Conference.html.