

FALLNEST: Dual-Domain Dataset Condensation of Imbalanced IMU Data for Wearable Fall Detection

Rashi Niyas P*, Dr. Ritika Jain[†], and Dr. Hitika Tiwari*

*Indian Institute of Technology Madras
Zanzibar
zda24m005@iitmz.ac.in, hitika@iitmz.ac.in

[†]Indian Institute of Technology Guwahati
Assam, India
rjain1992@iitg.ac.in

Abstract

Wearable fall detection is hampered by rare fall events, scarce labeled IMU data, and the cost of training deep models on large, imbalanced corpora. Existing methods either rely on hand-crafted features with shallow classifiers, which limits learning from raw signals, or train deep networks on full datasets that require re-sampling to counter class skew and demand substantial data and compute. To address this, we propose a novel **Fall** detection framework via **Normalized-gradient** and **Embedding-matched Synthetic Time-frequency** condensation, dubbed **FALLNEST**, a data-centric approach that distills large, imbalanced IMU datasets into tiny, class-balanced synthetic training sets. Building on dual-domain time-series condensation (CondTSC), FALLNEST introduces a fall-specific 7-channel input with accelerometer vector magnitude, class-balanced K-Means initialization, and normalized gradient matching with gradient clipping, combined with MMD-based embedding alignment and multi-view Fourier-domain augmentation, so that class balance arises from the condensed set itself rather than from re-sampling. We evaluate on SisFall, KFall, and MobiFall using strict subject-wise splits and naturally imbalanced test sets, reporting accuracy, Macro-F1, and PR-AUC over five seeds. A 102K-parameter TinyCNN trained on only 32 to 146 synthetic windows (about only 1% of the training windows) reaches 93.5-95.8% mean accuracy and 0.89-0.98 PR-AUC, with standard deviations of 1.0-1.7% in accuracy, demonstrating the efficacy of our approach.

Keywords: Fall detection, wearable sensors, dataset condensation, class imbalance, dual-domain matching.

Introduction

Falls are a leading cause of injury among older adults and a recurring hazard in workplaces [1], and wearable inertial measurement units (IMUs) enable continuous, unobtrusive monitoring [2]. Learning a reliable fall detector from IMU data is hard for two reasons. Falls are rare compared with activities of daily living (ADLs), so labeled data are scarce and skewed [3], and wearables operate under tight power, memory, and processing budgets [4]. The usual remedies for skew, oversampling and augmentation, enlarge the training set and its compute cost. Lightweight pipelines instead rely on hand-crafted features, for example spectral features from 25 Hz IMU data fed to a feed-forward network [5], which limits learning from raw signals. Deep networks learn features automatically and reach near-perfect accuracy on standard benchmarks [6, 7, 8], but need large imbalanced training sets, substantial compute, and often re-sampling. Dataset condensation distills a large dataset into a small synthetic set that retains its training value [9]. It has mainly been studied on images, and its time-series extension [10] was not evaluated on wearable fall detection under subject-wise splits with naturally imbalanced test sets.

We propose **Fall** detection framework via **Normalized-gradient** and **Embedding-matched Synthetic Time–frequency condensation (FALLNEST)**, which shrinks the training data rather than the model. It distills a large, imbalanced IMU dataset into a tiny, class-balanced synthetic set on which a lightweight CNN learns discriminative fall patterns. FALLNEST builds on CondTSC [10] and differs in four ways: (i) a fall-specific 7-channel input (tri-axial accelerometer, tri-axial gyroscope, accelerometer vector magnitude); (ii) class-balanced K-Means initialization, which balances the set by construction; (iii) normalized gradient matching with gradient clipping, which stabilizes optimization across datasets of very different sizes; and (iv) strict subject-wise splits with naturally imbalanced test sets. Raw signals are left unfiltered so that sharp impact peaks are preserved. On SisFall [11], KFall [12], and MobiFall [13], a 102 K-parameter TinyCNN trained on 32 to 146 synthetic windows (about 1% of the training windows) reaches 93.5 to 95.8% mean accuracy and 0.89 to 0.98 PR-AUC over five seeds. The roughly 100-fold reduction applies to the downstream training set, not to the data accessed during condensation, which uses all of $\mathcal{T}$; a real validation set only selects the checkpoint.

Our contributions are: a dual-domain condensation framework that turns imbalanced IMU recordings into compact, class-balanced training sets without re-sampling; a dual-objective scheme pairing normalized, clipped gradient matching with MMD alignment in both domains, plus multi-view Fourier augmentation; and a rigorous evaluation protocol (subject-wise splits, natural test priors, accuracy, Macro-F1, PR-AUC over five seeds) with ablations and a PCA analysis of real versus synthetic embeddings. All datasets are public, and code will be released upon acceptance.

Method

FALLNEST condenses a large, imbalanced IMU training set into a tiny, class-balanced synthetic set, and a lightweight classifier is trained on that set alone (Fig. 1). Recordings become subject-disjoint windows; a balanced synthetic set is initialized with class-wise K-Means and refined in time and frequency domains through four augmented views by a joint normalized gradient-matching and MMD objective (Alg. 1). A TinyCNN is then trained on the condensed set ($\mathcal{S}$), selected on a real validation set ($\mathcal{V}$), and evaluated on a subject-independent, naturally imbalanced test set ($\mathcal{T}_{\text{test}}$).

The real training set is $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^n$ with $x_i \in \mathbb{R}^{C \times L}$ ($C = 7$ channels, L time steps) and $y_i \in \{0, 1\}$ (0: ADL, 1: Fall); $\mathcal{T}_c$ denotes class c , and the validation set $\mathcal{V}$ and test set $\mathcal{T}_{\text{test}}$ are subject-disjoint from $\mathcal{T}$ and each other. The synthetic set $\mathcal{S}$ holds *spc* learnable tensors per class with fixed labels:

$$spc = \lceil 0.01 n/2 \rceil, \quad |\mathcal{S}| = 2 spc \approx 0.01 |\mathcal{T}|, \quad (1)$$

with $n = |\mathcal{T}|$ counting *all* training windows. With $\mathcal{L}_{\mathcal{D}}(\theta)$ the mean cross-entropy of a network with parameters θ on set $\mathcal{D}$, condensation seeks

$$\min_{\mathcal{S}} \mathcal{L}_{\mathcal{T}}(\theta^*(\mathcal{S})) \quad \text{s.t.} \quad \theta^*(\mathcal{S}) = \arg \min_{\theta} \mathcal{L}_{\mathcal{S}}(\theta), \quad (2)$$

i.e. a network trained only on $\mathcal{S}$ should perform on real data like one trained on $\mathcal{T}$. Solving Eq. (2) exactly needs a training run per update, so we replace it with the matching objectives of Eqs. (5) and (6).

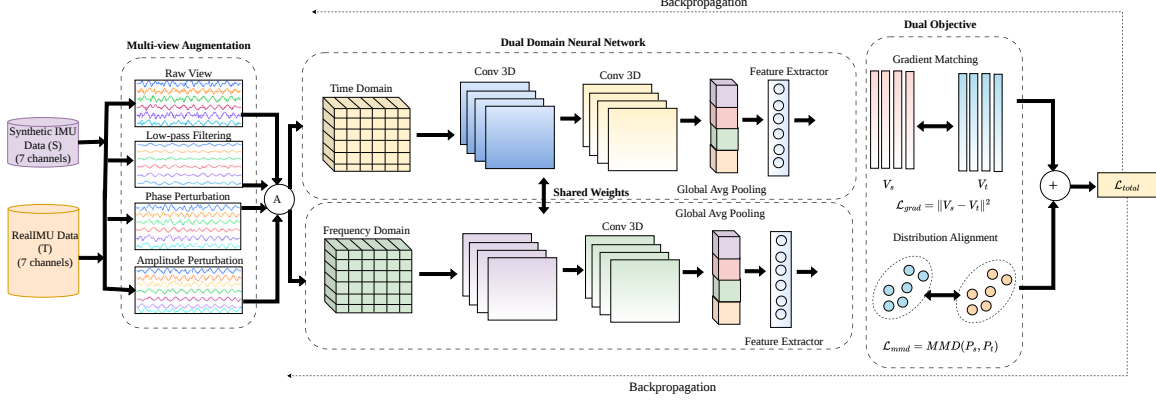


Figure 1: FALLNEST pipeline. Imbalanced IMU windows $\mathcal{T}$ are condensed into a class-balanced set $\mathcal{S}$ via K-Means initialization, four-view Fourier augmentation, and joint gradient/MMD matching in the time and frequency domains. A TinyCNN is then trained on the obtained condensed set $\mathcal{S}$ and tested on $\mathcal{T}_{\text{test}}$.

Data Preparation

SisFall, KFall, and MobiFall contain 15, 15, and 4 fall types and 19, 21, and 9 ADL types; all falls map to $y=1$ and all ADLs to $y=0$. Recordings are resampled to 200 Hz and segmented with a 2 s stride into windows of $L = 2000, 1200$, and 1000 samples (10, 6, and 5 s, chosen from each dataset’s mean fall duration). Each window inherits the binary label of its source trial, since only trial-level labels are available, and windows containing transitions are kept so that training matches deployment. Splits are *by subject* (26/6/6, 16/4/4, and 15/4/13 subjects for SisFall, MobiFall, and KFall; all five seeds share the same split; see Table 1). Besides the tri-axial accelerometer and gyroscope, each window carries the accelerometer vector magnitude $AVM_t = \sqrt{a_{x,t}^2 + a_{y,t}^2 + a_{z,t}^2}$ of the *raw* axes $a_{\cdot,t}$ at step t , an orientation-independent impact measure. Each of the 7 channels is standardized with the mean and std of $\mathcal{T}$. No global low-pass filter is applied, since it would smooth the impact peak; smoothing enters only as an augmentation view.

Class-Balanced K-Means Initialization

For each class c we cluster the flattened windows $\text{vec}(x)$ of $\mathcal{T}_c$ into spc clusters (Eq. (1)) with centroids $\mu_{c,k}$,

$$\{\mu_{c,k}\}_{k=1}^{spc} = \arg \min_{\{\mu_k\}} \sum_{x \in \mathcal{T}_c} \min_k \|\text{vec}(x) - \mu_k\|_2^2, \quad (3)$$

and reshape each $\mu_{c,k}$ to $C \times L$ as an initial synthetic sample. Both classes get exactly spc samples, so $\mathcal{S}$ is balanced whatever the real fall/ADL ratio, and centroids spread over the dominant modes of each class, which matters most for the heterogeneous ADL class. Centroids (Eq. (3)) are averages and blur sharp peaks, so restoring impact structure is the job of the refinement.

Dual-Domain Representation

The time-domain input is the window itself; the frequency-domain input is the channel-averaged magnitude spectrogram

$$\mathbf{P}(x) = \frac{1}{C} \sum_{j=1}^C |\text{STFT}(x_j)| \in \mathbb{R}^{129 \times F}, \quad (4)$$

where x_j is channel j , $n_{\text{fft}} = 256$ (giving $n_{\text{fft}}/2 + 1 = 129$ frequency bins), hop 64, and $F = 28, 15, 12$ frames for SisFall, KFall, MobiFall. Each domain $d \in \{\text{t}, \text{f}\}$ has its own backbone ϕ_d , shared between synthetic and real data.

Multi-View Fourier Augmentation

Each sample S is observed through four views v : the raw signal and three Fourier-domain views produced in sequence. The low-pass view $S_{\text{LPF}} = \text{FFT}^{-1}(H_{f_c} \odot \text{FFT}(S))$ applies a hard mask H_{f_c} ($\odot$: elementwise product) at cutoff $f_c = 5$ Hz and emphasizes macro-motion. The phase-perturbed view S_{PP} rebuilds S_{LPF} from its spectrum magnitude and its phase plus noise $\Phi \sim \mathcal{N}(0, 0.1^2)$ per frequency bin (tolerance to small timing shifts). The magnitude-perturbed view S_{MP} adds $\Gamma \sim \mathcal{N}(0, 0.1)$ to each magnitude bin of S_{PP} (intensity variation). All views are returned to the time domain.

Dual-Objective Matching

For domain d , view v , and class c , we draw random initial parameters θ_d^0 of ϕ_d , take N differentiable SGD steps on $v(\mathcal{S}_c)$ giving $\theta_d^{\mathcal{S}}$ and M SGD steps on real batches of $\mathcal{T}_c$ giving $\theta_d^{\mathcal{T}}$, and form the displacements $G^{\mathcal{S}} = \theta_d^0 - \theta_d^{\mathcal{S}}$ and $G^{\mathcal{T}} = \theta_d^0 - \theta_d^{\mathcal{T}}$. They are aligned with a normalized squared distance, which, unlike cosine matching, penalizes both direction and magnitude mismatch:

$$\mathcal{L}_{\text{grad}}^{(v,d)} = \sum_c \frac{\|G^{\mathcal{S}} - G^{\mathcal{T}}\|_2^2}{\|G^{\mathcal{S}}\|_2^2 + \|G^{\mathcal{T}}\|_2^2 + \epsilon}. \quad (5)$$

Here $\epsilon > 0$ is a small constant, and each class-wise term lies in $[0, 2]$ regardless of dataset size, comparable to the bounded MMD term. Penultimate-layer embeddings $z^{\mathcal{S}}$ and $z^{\mathcal{T}}$ of synthetic and real samples of the same class are aligned with the squared MMD under a Gaussian kernel of bandwidth $\sigma = 1$:

$$\mathcal{L}_{\text{emb}}^{(v,d)} = \sum_c \text{MMD}^2(z^{\mathcal{S}}, z^{\mathcal{T}}). \quad (6)$$

Eq. (5) aligns *how the network learns*, Eq. (6) *what it represents*, and the total objective (weight λ_{emb} on the embedding term) is

$$\mathcal{L} = \sum_v \sum_d \left(\mathcal{L}_{\text{grad}}^{(v,d)} + \lambda_{\text{emb}} \mathcal{L}_{\text{emb}}^{(v,d)} \right), \quad \lambda_{\text{emb}} = 1. \quad (7)$$

We backpropagate through the unrolled inner loop, clip $\nabla_{\mathcal{S}} \mathcal{L}$ (Eq. (7)) elementwise to $[-1, 1]$, and update $\mathcal{S}$ with Adam ($\text{lr} = 0.01$).

Algorithm 1 FALLNEST condensation and training

Input: Train set $\mathcal{T}$, spc , views (raw, LPF, PP, MP), domains $\{t, f\}$, N , M , λ_{emb}

Output: Trained classifier f_{θ^*}

- 1: $\mathcal{S}_c \leftarrow$ K-Means centroids of $\mathcal{T}_c$, $c \in \{0, 1\}$ $\triangleright$ Eq. (3)
 - 2: **for** epoch = 1 **to** 100 **do**
 - 3: **for** each view v , domain d , class c **do**
 - 4: draw θ_d^0 ; run N SGD steps on $v(\mathcal{S}_c)$ and M on batches of $\mathcal{T}_c$
 - 5: $\mathcal{L} \leftarrow \mathcal{L}_{\text{grad}} + \lambda_{\text{emb}} \mathcal{L}_{\text{emb}}$ $\triangleright$ Eqs. (5), (6)
 - 6: **end for**
 - 7: $\mathcal{S} \leftarrow \text{Adam}(\mathcal{S}, \text{clip}_{[-1,1]}(\nabla_{\mathcal{S}} \mathcal{L}))$; stop early if converged
 - 8: **end for**
 - 9: Train a fresh TinyCNN on $\mathcal{S}$; keep the epoch with best Macro-F1 on $\mathcal{V}$
-

Downstream Training, Backbone, and Evaluation

A fresh TinyCNN is trained on $\mathcal{S}$ only (cross-entropy, 100 epochs, Adam lr 0.001, batch size 16). Each backbone has three Conv1D–BatchNorm–ReLU blocks, adaptive average pooling, and a fully connected layer; the deployed classifier is the time-domain TinyCNN alone, and the frequency branch is used only during condensation. The model has 102 K parameters (≈ 400 KB in float32) and needs about 103 M MACs ($L=1000$) to 206 M MACs ($L=2000$) per inference, with 2.15 ms median CPU latency (Kaggle CPU, single-threaded PyTorch, batch size 1). These are architecture properties, not condensation effects; microcontroller validation is future work. We keep the checkpoint with the highest Macro-F1 on the real validation set, $e^* = \arg \max_e \text{MacroF1}(f_{\theta_e}; \mathcal{V})$, where f_{θ_e} is the classifier after epoch e ; since condensation uses $\mathcal{T}$ alone, $\mathcal{V}$ is the only other real data that enters. The checkpoint is evaluated once on $\mathcal{T}_{\text{test}}$. Each of five seeds draws a new K-Means initialization, runs an independent condensation, and trains a new classifier on the same subject-wise split. We report mean $\pm$ std of Accuracy, Macro-F1, and PR-AUC (Fall positive), plus fall precision, recall, specificity, and false alarms per hour (FA/h, false-positive windows per ADL hour at the 2 s stride; a window-level estimate on laboratory data). Overlapping windows from one trial are correlated, so window counts overstate independent events.

Theoretical Analysis of the Objective

Fix a domain, view, and class, and let $\Delta = \theta^{\mathcal{S}} - \theta^{\mathcal{T}} = G^{\mathcal{T}} - G^{\mathcal{S}}$ be the parameter gap, r the class-wise summand of Eq. (5), and P the real-batch distribution. Assume the real loss L_P (expected cross-entropy under P) is Λ -Lipschitz and β -smooth on a convex set containing the iterates, SGD steps (rate η , gradient norm $\leq g_{\text{max}}$) are bounded by ηg_{max} (so $\Omega^2 = \eta^2 g_{\text{max}}^2 (N^2 + M^2) + \epsilon$), the head loss is approximable in the RKHS of the Gaussian kernel (norm $\leq B$, error $\leq \kappa$), and the n_T real embeddings are i.i.d. from P given $\mathcal{S}$.

Corollary 1. *Let $\delta \in [r_0, 1)$ with r_0 the value of r at $\epsilon = 0$, $s = 1/(1 - \delta)$, $\rho_{\pm} = s \pm \sqrt{s^2 - 1}$, and $K(\delta) = 1 + \rho_+^2$. Then $\cos \angle(G^{\mathcal{S}}, G^{\mathcal{T}}) \geq 1 - \delta$, $\|G^{\mathcal{S}}\|/\|G^{\mathcal{T}}\| \in [\rho_-, \rho_+]$, $|L_P(\theta^{\mathcal{S}}) - L_P(\theta^{\mathcal{T}})| \leq \Lambda \|\Delta\|$, and, with probability at least $1 - \eta'$ over the real batch,*

$$\|\Delta\| \leq \min \left\{ \Omega \sqrt{r}, \sqrt{\delta K(\delta)} \|G^{\mathcal{T}}\| \right\}, \quad (8)$$

$$L_P(\theta^{\mathcal{S}}) \leq L_{\mathcal{S}}(\theta^{\mathcal{S}}) + B \left[\sqrt{\mathcal{L}_{\text{emb}}} + \frac{1 + \sqrt{\ln(1/\eta')}}{\sqrt{n_T}} \right] + 2\kappa. \quad (9)$$

Eq. (8) shows that Eq. (5) controls both direction and scale of the update, which bounds the real-loss gap to the real-trained network, while Eq. (9) lets the MMD term of Eq. (6) bound the real loss by the synthetic loss. The constants are unmeasured, so the bounds are qualitative and concern the training distribution, not unseen subjects (assessed empirically below).

Experimental Setup

Datasets and Implementation

We evaluate on three binary fall-detection datasets: SisFall (38 subjects), KFall (32), and MobiFall (24). Dataset statistics and subject-wise splits are summarized in Table 1. All condensation hyperparameters are fixed across datasets, except the real-data batch size, with $spc = \{73, 16, 18\}$ for SisFall, MobiFall, and KFall, respectively. Condensation uses at most 100 epochs, learning rate 0.01, $\lambda_{emb} = 1$, $N = 15$, $M = 100$, inner learning rate 0.001, patience 10, and $f_c = 5$ Hz. The real batch size is 128 for SisFall and 64 for KFall and MobiFall. No dataset-specific hyperparameter tuning is performed. All models are implemented in PyTorch and trained on a 16 GB P100 GPU. Experiments use five fixed random seeds; seed 42 is used for representative single-run results. Since the subject splits remain fixed across seeds, the reported standard deviations capture initialization variability rather than variability across subject partitions.

Table 1: Dataset statistics after subject-wise splitting (2 s stride). $\mathcal{T}$ is the only set used for condensation, $\mathcal{V}$ only selects the checkpoint, and $\mathcal{T}_{test}$ is used once and keeps its natural class distribution.

Dataset	Split	Subj.	Win.	Total	ADL	Fall
SisFall	$\mathcal{T}$	26	10 s	14,528	10,259	4,269
	$\mathcal{V}$	6		2,962	2,287	675
	$\mathcal{T}_{test}$	6		2,778	2,328	450
MobiFall	$\mathcal{T}$	16	5 s	3,146	2,568	578
	$\mathcal{V}$	4		1,000	856	144
	$\mathcal{T}_{test}$	4		571	427	144
KFall	$\mathcal{T}$	15	6 s	3,574	2,272	1,302
	$\mathcal{V}$	4		1,046	583	463
	$\mathcal{T}_{test}$	13		2,933	1,922	1,011

Results and Discussion

Classification Performance

Table 2 reports test Accuracy, Macro-F1, and PR-AUC. Using only 1% of training windows, FALLNEST achieves 93.5–95.8% accuracy and 0.891–0.975 PR-AUC, exceeding the always-ADL accuracy baselines by 9.7, 20.7, and 30.3 percentage points on SisFall, MobiFall, and KFall, respectively. Compared with full-data training, accuracy drops by 6.4, 4.5, and 3.6 points, respectively; SisFall also shows the largest PR-AUC reduction (0.999 to 0.891). Although five-seed variability is modest (1.0–1.7 accuracy points; 0.013–0.023 PR-AUC), fixed test subjects mean these standard deviations reflect initialization variability only. SisFall’s larger performance gap may relate to its lower fall prevalence, longer windows that dilute impact signatures (Table 1), and greater ADL heterogeneity relative to its 146-sample condensed set; these hypotheses require targeted evaluation. Table 4 presents the performance metrics in terms of precision, recall, and specificity for each dataset.

Comparison with Equal-Budget Real Subsets

Does condensation beat simply training on a 1% real subset? Table 3 compares FALLNEST with random real subsets of the same size $|\mathcal{S}|$, drawn either uniformly or with equal numbers

Table 2: Main results on subject-independent, naturally imbalanced $\mathcal{T}_{\text{test}}$. Rows $\mathcal{S}$ train on the condensed set (1% of $\mathcal{T}$), mean $\pm$ std over 5 seeds; rows $\mathcal{T}$ train the same TinyCNN on the full set (single run). Size is the number of training windows. Always-ADL accuracy floors: 83.8/74.8/65.5%; random-scorer PR-AUC floors: 0.162/0.252/0.345 (SisFall/MobiFall/KFall).

Dataset	Train set	Size	Acc (%)	Macro-F1	PR-AUC
SisFall	$\mathcal{S}$ (condensed)	146	93.50 $\pm$ 1.02	0.887 $\pm$ 0.016	0.891 $\pm$ 0.023
	$\mathcal{T}$ (full)	14,528	99.85	0.990	0.999
MobiFall	$\mathcal{S}$ (condensed)	32	95.48 $\pm$ 1.73	0.942 $\pm$ 0.023	0.975 $\pm$ 0.017
	$\mathcal{T}$ (full)	3,146	100.00	1.000	1.000
KFall	$\mathcal{S}$ (condensed)	36	95.80 $\pm$ 1.68	0.954 $\pm$ 0.018	0.973 $\pm$ 0.013
	$\mathcal{T}$ (full)	3,574	99.39	0.999	1.000

Table 3: Equal-budget comparison, Acc / Macro-F1 / PR-AUC (mean over 5 seeds). Random subsets contain $|\mathcal{S}|$ windows drawn from $\mathcal{T}$.

Training set	SisFall	MobiFall	KFall
Real (uniform)	94.0/0.90/0.88	91.73/0.892/0.915	89.74/0.89/0.90
Real (balanced)	92.55/0.879/0.765	93.06/0.916/0.932	95.2/0.95/0.86
FALLNEST (ours)	93.50/0.887/0.891	95.48/0.942/0.975	95.80/0.954/0.973

Table 4: Operating-point metrics (seed 42)

Dataset	Precision	Recall	Specificity
SisFall	0.84	0.86	0.97
MobiFall	0.90	0.97	0.96
KFall	0.96	0.94	0.98

per class, under the same TinyCNN, checkpoint selection, and five-seed protocol.

On MobiFall and KFall, where only 32 and 36 samples are available, FALLNEST leads in accuracy and especially PR-AUC (0.975 and 0.973 versus at most 0.932 and 0.90 for the random subsets), so condensation helps most where random draws leave ADL regions uncovered, whereas synthetic samples span both classes (Fig. 2). On SisFall the uniform subset is competitive in accuracy (94.0 versus 93.5) and Macro-F1, so there condensation mainly improves ranking quality over the class-balanced subset (PR-AUC 0.891 versus 0.765), and its benefit over uniform sampling is not established. Balancing alone does not explain the gains on the two smaller datasets, since the class-balanced subset trails FALLNEST in all three metrics there.

Comparison with Full-Data Methods

Table 5 lists recent full-data methods, which report 98–99.8% accuracy on these datasets, with the FALLNEST rows for the same datasets below. Each published row gives one metric only, accuracy for most methods and PR-AUC for the BiLSTM–BiGRU model, so a full comparison on both metrics is not possible. Against these methods, FALLNEST gives up about 3–6 points of accuracy (for example 99.3% versus 93.5% on SisFall) while using about two orders of magnitude fewer training windows. The comparison is not like for like, because published protocols differ in splitting, class balance, and metrics. LiteFallNet [8] and BiLSTM–BiGRU [7] use random window-level splits, which can inflate results through temporal leakage, whereas [6] and [14] use variants of subject-wise cross-validation. One could suspect that published numbers are inflated by window-level splits, but our own full-data TinyCNN under strict subject-wise splitting reaches 99.85%, 100.00%, and 99.39%, at the level of the published methods, so leakage is not the main explanation. These benchmarks appear saturated under either protocol, and the 3–6 point gap is mostly the price

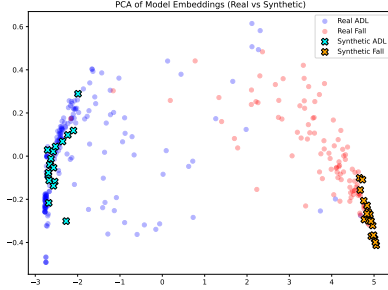


Figure 2: PCA projection of feature embeddings. Synthetic samples (crosses) lie within the real ADL and Fall distributions (circles).

Table 5: Published full-data methods (100% of the training data) versus FALLNEST (about 1%). Splits and metrics differ, so the table is indicative only.

Method (year)	Dataset	Acc / PR-AUC
Deep learning (2024) [6]	SisFall / MobiFall	Acc $\approx$ 99.3 / 99.6%
LiteFallNet (2025) [8]	KFall / SisFall	Acc $\approx$ 99.0 / 98–99%
BiLSTM–BiGRU (2025) [7]	SisFall	PR-AUC $\approx$ 0.99
Hybrid KNN (2020) [14]	KFall	Acc $\approx$ 99.8%
FALLNEST (1% data)	SisFall	93.5% / 0.891
FALLNEST (1% data)	MobiFall	95.5% / 0.975
FALLNEST (1% data)	KFall	95.8% / 0.973

of the 100-fold data reduction. We do not claim parity with full-data training. The case for condensation is use cases in which storing or processing the full corpus is impractical, such as rapid prototyping or on-device personalization.

Ablation Studies

Table 6 reports single runs (seed 42), for different ablations. MobiFall is generally the most sensitive. Accuracy and PR-AUC sometimes diverge: on MobiFall, several variants lose 6–20 accuracy points while keeping PR-AUC at 0.97–0.98, suggesting an operating-point shift between validation (14.4% fall prevalence) and test data (25.2%). Frequency-only is the exception, with PR-AUC down to 0.71.

Domain. Frequency-only loses 4.3, 27.3, and 4.7 accuracy points on SisFall, MobiFall, and KFall, and time-only loses 3.6, 0.9, and 4.0. The dual-domain gain is clear on SisFall and KFall, while the MobiFall time-only result is within seed noise. The frequency-only collapse on MobiFall may reflect its 12 spectrogram frames.

Initialization. Random initialization costs 1.7 and 5.6 points on SisFall and MobiFall and gains 0.3 on KFall; on SisFall, PR-AUC drops from 0.90 to 0.79.

Augmentation. Removing it costs 3.4, 20.0, and 1.6 points, most on MobiFall, where only 32 samples are condensed. Its preserved PR-AUC suggests an operating-point effect.

Loss. Either loss alone is below the combination, except gradient-only on SisFall (within noise). Gradient-only beats MMD-only on SisFall and KFall, and the reverse holds on MobiFall. The ablations do not isolate AVM, normalized versus cosine matching, or clipping, and we include no CondTSC or other coreset/condensation baselines (e.g. herding, k-center, DM), so these differences from CondTSC are design choices, not validated gains.

Table 6: Ablations, Acc (%) / PR-AUC, single run (seed 42). Full = dual domain, K-Means init., multi-view aug., combined loss.

Config.	SisFall	MobiFall	KFall
Full	95.03/0.90	98.07/0.996	96.59/0.99
Freq. only	90.78/0.896	70.75/0.71	91.9/0.93
Time only	91.47/0.88	97.20/0.995	92.57/0.91
Random init	93.30/0.79	92.47/0.98	96.9/0.95
No aug.	91.68/0.83	78.11/0.978	95.02/0.95
Grad. only	93.74/0.85	84.41/0.97	94.6/0.93
MMD only	88.66/0.84	91.07/0.97	93.7/0.92

Training and Inference Cost

Condensation reduces the *training data* and offline compute, not inference latency, memory, or energy, so we report both and claim no deployment speedup from it. Table 7 trains architectures from 0.8 K to about 1 M parameters on the same condensed sets.

TinyCNN is best on SisFall and MobiFall and within 0.2 points on KFall (96.6 versus 96.8 for LargeMLP). Smaller CNNs underfit unevenly: MicroCNN is 2.7 points below TinyCNN on SisFall but 30.0 on MobiFall, below the 74.8% always-ADL floor, while SmallCNN (76.7%) and LargeMLP (75.1%) barely exceed it. On SisFall the high floor (83.8%) hides this, and

Table 7: Model-scale sweep on the same condensed $\mathcal{S}$ (Acc / Macro-F1; Acc only for KFall; seed 42) and cost of the TinyCNN used throughout.

Arch.	Params	SisFall	MobiFall	KFall
MicroCNN	0.8K	92.3/0.87	68.1/0.67	73.6
SmallCNN	10K	93.3/0.88	76.7/0.75	—
TinyCNN	102K	95.0/0.91	98.1/0.975	96.6
LargeMLP	~500K–1M	93.3/0.88	75.1/0.73	96.8
MACs / size / CPU latency		103–206 M / $\approx$ 400 KB / 2.15 ms		
Condensation time		~82 min	~21 min	~26 min
Train: $\mathcal{S}$ vs. full $\mathcal{T}$		~8 vs 37 min	~3 vs 18 min	~4.5 vs 25 min

Macro-F1 (0.87 versus 0.91) is the better indicator. LargeMLP differs in architecture as well as size, so the sweep does not isolate parameter count. TinyCNN needs 103–206 M MACs, about 400 KB, and 2.15 ms median CPU latency (single-threaded PyTorch, batch size 1); we did not test it on a microcontroller. Training on $\mathcal{S}$ is 4.6 to 6 times faster than on the full $\mathcal{T}$, but condensation plus training (about 90, 24, and 30.5 min) exceeds one full-data run (37, 18, and 25 min), so condensation pays off only when many architectures or hyperparameters are tried on one condensed set.

Conclusion

We proposed **Fall** detection framework via **N**ormalized-gradient and **E**MBEDding-matched **S**ynthetic **T**ime–frequency condensation (**FALLNEST**), a dual-domain dataset condensation framework for wearable fall detection that distills a large, imbalanced IMU training set into a tiny, class-balanced synthetic set using class-balanced K-Means initialization, a 7-channel input with AVM, normalized and clipped gradient matching with MMD alignment, and multi-view Fourier augmentation. Under strict subject-wise splits and naturally imbalanced test sets, a 102 K-parameter TinyCNN trained on 32 to 146 synthetic windows (about 1% of the training windows) reaches 93.5–95.8% mean accuracy and 0.89–0.98 PR-AUC over five seeds, 3.6 to 6.4 points below full-data training. The ablations indicate that K-Means initialization, multi-view augmentation, and the combined loss each matter, most clearly where the synthetic set is smallest. Inference cost was measured on a CPU only; future work includes false-alarm analysis on free-living data and microcontroller deployment.

References

- [1] World Health Organization, “Falls,” 2024, Accessed: 2024-09-15. [1](#)
- [2] Avneesh Singh, Saqib Ur Rehman, Somsak Yongchareon, and Peter H. J. Chong, “Sensor technologies for fall detection systems: A review,” *IEEE Sensors Journal*, vol. 20, no. 13, pp. 6889–6919, 2020. [1](#)
- [3] Nannapas Pannurat, Treethap Thiemjarus, and Ekarat Nantajeewarawat, “Automatic fall monitoring: A review,” *Sensors*, vol. 14, no. 7, pp. 12900–12936, 2014. [1](#)
- [4] Christian Banbury, Jessica Hinkle, Bo Hu, Shuo Lu, Jonathan Masi, Orest Melnyk, Alex Miller, Yang Ni, Vikas Saletore, Jack Smith, Andrew Trask, Shiva Venkatraman, Haoran Xu, Bin Yang, David Young, and Haotian Zhao, “The challenges and opportunities of machine learning for embedded systems,” *Proc. 11th USENIX Conf. on Networked Systems Design and Implementation (NSDI)*, pp. 317–329, 2014. [1](#)
- [5] Sara Campanella, Alaa Alnasef, Laura Falaschetti, Alberto Belli, Paola Pierleoni, and Lorenzo Palma, “A novel embedded deep learning wearable sensor for fall detection,” *IEEE Sensors Journal*, vol. 24, no. 9, pp. 15219–15229, 2024. [1](#)
- [6] Jinxi Zhang, Zhen Li, Yu Liu, Jian Li, Hualong Qiu, Mohan Li, Guohui Hou, and Zhixiong Zhou, “An effective deep learning framework for fall detection: model development and study design,” *Journal of medical internet research*, vol. 26, pp. e56750, 2024. [1](#), [7](#), [8](#)
- [7] Mithun Singh Ahirwar and Vaibhav Soni, “Optimized fall detection using hybrid bilstm bigru additive attention model and baqa driven feature selection system,” *Scientific Reports*, vol. 15, no. 1, pp. 39254, 2025. [1](#), [7](#), [8](#)
- [8] Emmanuel Owusu, Isaac Acquah, Michael Asiedu Asare, and Benjamin Appiah Yeboah, “Litefallnet: A lightweight deep learning model for efficient real-time fall detection,” *Digital Health*, vol. 11, pp. 20552076251386698, 2025. [1](#), [7](#), [8](#)
- [9] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros, “Dataset distillation,” *arXiv preprint arXiv:1811.10959*, 2020. [1](#)
- [10] Zhanyu Liu, Ke Hao, Guanjie Zheng, and Yanwei Yu, “Dataset condensation for time series classification via dual domain matching,” in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 1980–1989, ACM. [1](#), [2](#)
- [11] Angela Sucerquia, Jos’e David L’opez, and Jes’us Francisco Vargas-Bonilla, “Sisfall: A fall and movement dataset,” *Sensors*, vol. 17, no. 1, pp. 198, 2017. [2](#)
- [12] Xiaoqun Yu, Jaehyuk Jang, and Shuping Xiong, “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*, vol. 13, pp. 692865, 2021. [2](#)
- [13] George Vavoulas, Matthew Padiaditis, Emmanouil G Spanakis, and Manolis Tsiknakis, “The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones,” in *13th IEEE International Conference on BioInformatics and BioEngineering*. IEEE, 2013, pp. 1–4. [2](#)
- [14] Xiaoqun Yu, Hai Qiu, and Shuping Xiong, “A novel hybrid deep neural network to predict pre-impact fall for older people based on wearable inertial sensors,” *Frontiers in bioengineering and biotechnology*, vol. 8, pp. 63, 2020. [7](#), [8](#)