Accurate and Efficient Real-World Fall Detection Using Time Series Techniques

Timilehin B. Aderinola[1✉], Luca Palmerini[2], Ilaria D'Ascanio[2], Lorenzo Chiari[2], Jochen Klenk[3], Clemens Becker[3,4], Brian Caulfield[1], and Georgiana Ifrim[1]

[1] Insight SFI Research Centre for Data Analytics, University College Dublin, Ireland
{timilehin.aderinola,b.caulfield,georgiana.ifrim}@ucd.ie
[2] Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, Italy
{luca.palmerini,ilaria.dascanio2,lorenzo.chiari}@unibo.it
[3] Department of Clinical Gerontology, Robert Bosch Hospital, Stuttgart, Germany
{Jochen.Klenk,clemens.becker}@rbk.de
[4] Digital Geriatrics Unit, Heidelberg University Hospital, Heidelberg, Germany

Abstract. Falls pose a significant health risk, particularly for older people and those with specific medical conditions. Therefore, timely fall detection is crucial for preventing fall-related complications. Existing fall detection methods often have high false alarm or false negative rates, and many rely on handcrafted features. Additionally, most approaches are evaluated using simulated falls, leading to performance degradation in real-world scenarios. This paper explores a new fall detection approach leveraging real-world fall data and state-of-the-art time series techniques. The proposed method eliminates the need for manual feature engineering and has efficient runtime. Our approach achieves high accuracy, with false alarms and false negatives each as few as one in three days on FARSEEING, a large dataset of real-world falls (mean F_1 score: 90.7%). We also outperform existing methods on simulated falls datasets, FallAllD and SisFall. Furthermore, we investigate the performance of models trained on simulated data and tested on real-world data. This research presents a real-time fall detection framework with potential for real-world implementation.

Keywords: Fall detection · Time series · Real-world falls.

1 Introduction

A fall is "an event which results in a person coming to rest inadvertently on the ground, floor, or other lower level". Globally, falls cause more than 684,000 deaths and about 172 million disabilities annually [1]. Most falls go unreported, especially those without injuries [42]. However, older adults [52] and people with specific medical conditions are considered high-risk groups (HRG) because they fall more frequently [33] and may be unable to alert anyone if they sustain fall-related injuries. Hence, falls among HRGs can significantly affect their livelihood, instilling a fear of falling, threatening their independence, and posing life-threatening risks. The aging global population increases the strain on healthcare

systems from falls, due to the need for hospitalisation, medical care, and potential surgery [5]. Consequently, extensive research has been dedicated to automated fall detection for timely intervention to prevent fall-related complications, such as permanent disability or mortality.

Fall detection typically involves data capture, preprocessing, feature extraction, and classification [26]. The data capture stage records participants' daily activities to identify both normal activities of daily living (ADLs) and falls. Fall data can be collected using vision-based techniques, ambient devices, and wearable sensors. Recent vision-based techniques leverage human pose estimation for fall detection using body geometry features in videos [3]. However, vision-based techniques face challenges like privacy concerns and high storage requirements. Ambient devices, while suitable for private areas like bathrooms, are expensive and may have limited coverage [26]. Wearable devices offer a practical solution for capturing real-world falls since they are relatively low-cost and can be carried for extended periods [21] in both indoor and outdoor settings.

Fall detection algorithms analyse motion data to distinguish between falls and ADLs using threshold-based, machine learning based, or deep learning based approaches [26]. Threshold-based methods [22, 49] use cut-off values set on the sensor signals to distinguish between falls and ADLs. However, these techniques commonly have high false alarm rates [44], which could lead to "false alarm fatigue" [34]. On the other hand, machine learning-based methods such as [23,48] use conventional classifiers with manually crafted features, yet, modelling the diverse factors that lead to falls is challenging. These factors can be internal (age, impaired mobility, disease), external (lighting, obstacles), or situational (activity leading to the fall). Consequently, fall feature extraction is complex and often requires a multidisciplinary approach [52].

Recently, deep learning approaches [7,24,27,31,38,54] have been used for fall detection, removing the need for manual feature extraction. While these methods show promising results, most are trained and tested on simulated data due to the scarcity of real fall data, raising concerns about their applicability in real-world settings [50]. Furthermore, deep learning models require large amounts of representative data, long training times, with a large number of parameters [16]. Hybrid methods [15, 27, 53] combining thresholds, machine learning, and deep learning have been proposed to improve accuracy, but often at the expense of simplicity and efficiency.

A major limitation of most fall detection approaches is their lack of transferability to real-world scenarios. Studies [40, 47] have shown significant performance drops when models trained on simulated falls are tested on real-world falls. Several factors involved in real falls are difficult to simulate accurately. Hence, simulating falls involves assumptions that cannot be determined in advance in real-world falls. For instance, while real-world falls may lead to injuries, simulated falls use controlled fall heights to prevent injuries, resulting in lower impact velocities [4]. Additionally, for ethical reasons, simulated datasets are typically created with healthy young adults whose postural control differs from that of the elderly.

In this paper, we present a simple, yet efficient approach to fall detection using state-of-the-art time series techniques, trained and tested on real-world falls. Our main contributions are:

1. We perform realistic segmentation of falls and activities of daily living in a manner that simulates real-time signals and considers the context of falls.
2. We perform fall detection using both classic tabular machine learning methods and recent state-of-the-art time series classification algorithms.
3. We evaluate the transferability of the proposed algorithms from simulated datasets to real-world falls.

Our approach requires no feature engineering and has very short inference times. We evaluate it on the FARSEEING dataset [21], a large real-world dataset with 92 fallers (mean age 76.1±12.6 years) and 208 verified falls captured using inertial sensors. We also evaluate our approach on the simulated FallAllD [45] and SisFall [51] datasets, which use waist-worn sensors. For all datasets, we simulate a real-time scenario by using a fixed overlapping sliding window segmentation technique to extract falls and ADLs, followed by fall detection using state-of-the-art time series classifiers. Our methods obtain cross-validated F_1 scores up to 90.7% on FARSEEING and 97.2% on the simulated datasets. We also identify important segments in motion signals for fall detection using post-hoc explanation techniques. Our work proposes a realistic, accurate, and efficient framework for timely fall detection with great potential for real-world implementation. We achieve low daily false alarm and false negative rates of 0.29 and 0.31 respectively on real-world falls (FARSEEING), each equating to about 1 error in every 3 days. To support this paper, we have made all our code available[5].

The rest of this paper is organised as follows. Section 2 reviews related work on fall detection using time series and real-world falls. In Section 3, we discuss the datasets and preprocessing steps, including segmentation. Details and results of our experiments are given in Section 4, and Section 5 offers concluding remarks.

2 Related Work

In this section, we discuss recent time series approaches to fall detection, as well as fall detection algorithms evaluated on real-world falls.

2.1 Fall Detection Using Time Series Approaches

Although signals obtained from inertial sensors can be easily modelled as time series to preserve the temporal information and context of falls, most fall detection approaches have traditionally relied on manual feature extraction or deep representation learning. However, a few recent approaches have modelled fall detection as a time series classification problem.

Recently, a recurrent neural network approach was proposed for fall detection, achieving recall of up to 96% on the SisFall dataset. Similarly, a long-short-term

[5] https://github.com/mlgig/ts_fall_detection

memory (LSTM) model trained and tested on the SisFall dataset achieved 96.4% recall. Another approach, the Temporal Convolutional Network and Temporal Attention Network (TCN-TAM) [37], extracted features from motion signals as time series and obtained an 88% recall on the FallAllD dataset. Likewise, the DeepFall model [19], a deep residual network, achieved an F_1 score of 92.79% on the FallAllD dataset. However, despite modelling fall signals as time series, these techniques used complex deep learning models for representation learning and classification. Furthermore, training and testing were conducted on simulated datasets, with no analysis of their transferability to real-world data.

2.2 Real-World Fall Detection

Falls are accidental and diverse, making them rare compared to activities of daily living [20]. Consequently, most fall detection research relies on simulated falls, which, despite their limitations [50], are useful for developing detection solutions. The scarcity of real-world fall datasets means that few studies have trained and evaluated techniques on actual fall data.

In a recent study [35], an ensemble of random forests was trained on a clinical dataset of 25 multiple sclerosis patients monitored over 2 months in free-living conditions, although the dataset is not publicly available for analysis. Another study [40] trained several machine learning models on the FARSEEING dataset using five manually extracted features, achieving an F_1 score of 65%. A more recent study [43] achieved an F_1 score of 91% in fall detection using a Residual Network, though the authors used only a subset (22 subjects) of FARSEEING.

3 Materials and Methods

3.1 Datasets

We evaluate our fall detection techniques using the FARSEEING dataset [21], a large collection of real-world falls. For completeness, we also evaluate our techniques on two simulated fall datasets, FallAllD [45] and SisFall [51]. SisFall uses only waist-worn sensors because signals obtained from sensors placed around the centre of mass are preferred for fall detection [39]. Therefore, we use accelerometer signals from sensors worn around the centre of mass across all datasets for consistency. Fig. 1 shows sample acceleration signals from each dataset. Table 1 provides summary statistics for all the datasets after preprocessing.

FARSEEING. The FARSEEING dataset [21] comprises 208 verified falls from 92 individuals (mean age 76.1 $\pm$ 12.6 years) captured using tri-axial inertial sensors. Each fall includes 20 minutes of data centered around the impact event, with 10 minutes before and after the fall. The fall signals were captured across different studies, and hence, have various sensor configurations: 72 falls with accelerometer signals only, 15 with accelerometer and gyroscope signals, and 121 with accelerometer, gyroscope, and magnetometer signals. Sensors were placed

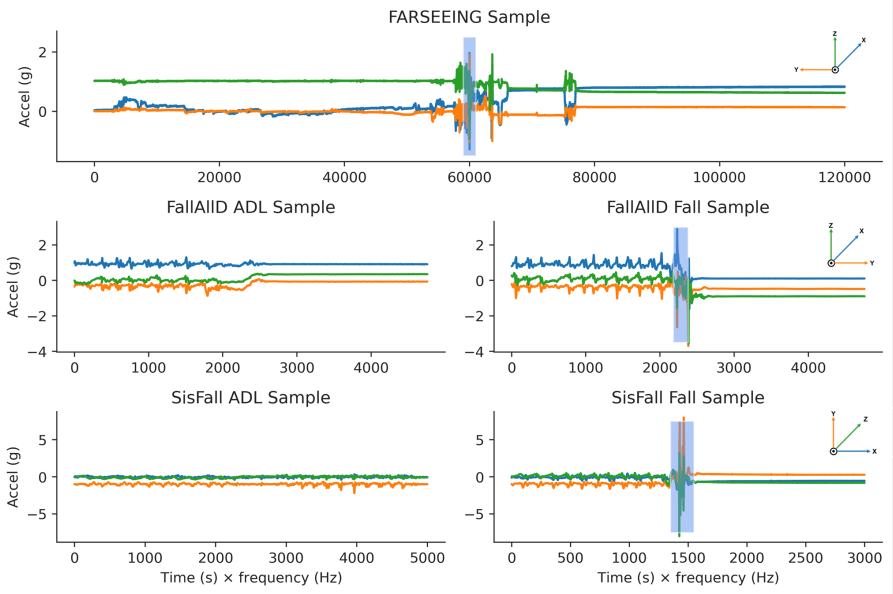

Fig. 1. Accelerometer signals from the FARSEEING, FallAllD, and SisFall datasets with the impact sample highlighted (from 1 second before to one second after the fall). The datasets have different sensor orientations; we aggregate the channels using the magnitude of acceleration, so the data becomes invariant to sensor orientation.

on the fifth lumbar (L5) for 150 falls and on the thigh for 58 falls. Sampling rates were 100Hz for 152 falls and 20Hz for 56 falls. In this study, we use accelerometer signals from 41 participants with L5 sensors, upsampling 20Hz signals to 100Hz using Fast Fourier Transform resampling [14] to minimise distortion. Some of the accelerometers had a range of ±2g, while others had a range of ±6g. Therefore, we clipped all the signals to ±2g for uniformity.

FallAllD. The FallAllD dataset [45] is an open-source simulated falls dataset. It contains 35 types of falls and 44 types of activities of daily living (ADLs) performed by 15 participants. The dataset comprises 26,420 files collected from tri-axial accelerometers, gyroscopes, magnetometers, and barometers worn on the waist, wrist, and neck. The accelerometers have a range of ±8g and sampling frequency of 238Hz. In this study, we use only the accelerometer signals collected from 14 participants using waist-worn sensors.

SisFall. The SisFall dataset [51] is an open-source simulated falls dataset with simulated falls and ADLs performed by 38 healthy participants, including 23 young adults and 15 older adults. Accelerometer and gyroscope signals were collected using a waist-worn sensor sampling at 200Hz.

3.2 Data Preprocessing

Transformation. To account for varying sensor orientations across datasets, we first aggregated the multivariate acceleration signals into orientation-invariant univariate acceleration magnitude signals: $M = \sqrt{A_x^2 + A_y^2 + A_z^2}$, where A_x, A_y, and A_z are the x, y, and z accelerations. We then performed segmentation to obtain samples, followed by standardisation across all samples to achieve zero mean and unit variance for modelling.

Segmentation. Research on falls often focuses on the immediate event, whereas the lived experience of older adults and healthcare providers considers the surrounding factors before and after the fall [50]. To capture the full context of falls and not just the "impact" event, we use a three-phase modification of the multi-phase fall model [2] with a falling phase, an impact phase, and a post-fall phase. As shown in Fig. 2, we define $[t_0, t_1)$ as the falling phase, $[t_1, t_2)$ as the impact phase, and $[t_2, t_3)$, the post-fall phase (which combines the resting and recovery phases of the original model). Each sample is $t_3 - t_0$ seconds long.

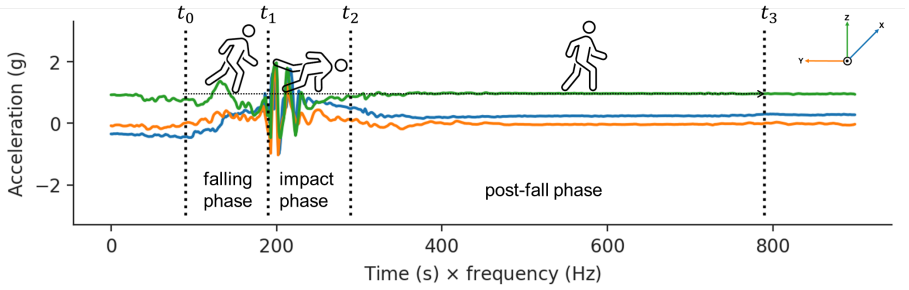

Fig. 2. Multiphase fall model on a FARSEEING fall sample.

As proposed in [40], we set the falling phase and impact phase to 1 second each. Since fall samples in SisFall are only 15 seconds long, we set a post-fall phase of 5 seconds, yielding a total sample window size of 7 seconds. However, using FARSEEING, we empirically demonstrate that longer post-fall phases do not always improve performance (see Appendix B). To simulate real-time signals, we use a fixed overlapping sliding window technique to segment samples from the accelerometer signals with a step size of 1 second ($\approx 86\%$ overlap).

Fall Signals Segmentation. Each fall sample in FARSEEING has 20 minutes of accelerometer data with 10 minutes before t_1 and 10 minutes after t_1. Hence, for FARSEEING, we first separated the portion $[t_0, t_3)$ of the signal that includes the reported fall by setting t_0 at 1 second before the impact event. We then discarded the remainder of the post-fall signal, since the subject is likely to walk

carefully after a fall [40] and the signal may not be characteristic of ADLs. ADL samples were then extracted from the 9-minute signal before the impact event.

In FallAllD and SisFall, falls and ADL were recorded as separate signals. Each FallAllD signal is 20 seconds long with the impact event centered (at the 10th second) in the fall trial. However, SisFall fall signals are 15 seconds long with the impact event at arbitrary locations in the signal. For all datasets, we extracted one fall sample from each fall signal, while we used the windowing technique to extract several ADL samples from ADL signals.

Segmentation of Activities of Daily Living. Following the pattern of the multi-phase model, each ADL signal has a duration of $t_3 - t_0$ seconds. Using the phase $[t_1, t_2)$ corresponding to the impact phase as the main search window, we scan from the beginning of the sample with a step size of 1 second. For each step, we select the current $[t_0, t_3)$ as a sample if the maximum magnitude of the acceleration within the impact sample is above a threshold τ. Similar works have selected τ as high as 1.9g [6], but we set τ at 1.4g to account for less energetic ADLs among older adults in the datasets.

4 Experiments

This section details baseline experiments using tabular classification models (Section 4.2), fall detection with time series classifiers (Section 4.3), cross-dataset evaluation (Section 4.4), and comparisons with related works (Section 4.5). Model explanation results are only preliminary and are presented in Appendix C. Detailed experimental results are in Appendix E.

All experiments were conducted using Python 3.11.9 on a Linux server with Ubuntu 22.04.3 LTS (1.5TB RAM, 24GB NVIDIA GeForce RTX 4090 GPU). We used time series classifiers from aeon [32] *v0.8.1* and tabular classifiers from scikit-learn [41] *v1.4.2*.

4.1 Evaluation Approach

Due to the small number of falls compared to ADLs (Table 1), we use a five-fold-subject cross-validation approach for evaluation. First, we shuffle and divide all subjects into five groups. Then, we use samples from the subjects in each group for testing while training on the remaining groups. We report metrics as mean $\pm$ standard deviation across all folds. We focus on AUC, Precision, Recall, and F_1 score for falls. Although adjusting classifier thresholds can increase recall by minimising undetected falls, it also reduces precision, leading to more false alarms. Therefore, we prioritise the F_1 score to balance recall and precision.

4.2 Baseline Experiments With Tabular Models

As recommended in [12], we first perform baseline experiments using tabular models. We use the scikit-learn implementations of five tabular classifiers,

Table 1. Dataset sizes after preprocessing and segmentation

Dataset	Subjects	ADL Samples	Falls Samples	Total	Falls:ADL Ratio
FARSEEING	41	1064	145	1209	0.14
FallAllD	14	1279	466	1745	0.36
SisFall	38	10843	1200	12043	0.11

namely, *ExtraTrees* with 150 estimators, *K-Nearest Neighbours* (KNN, $k = 5$), *LogisticRegressionCV* ($CV = 5$), *RandomForest* with 150 estimators, and *RidgeCV* ($CV = 5$). We perform no feature extraction but model each sample as a 1D vector $\mathbf{v} \in \mathbb{R}^L$, where $L = w \times f$, w is the window size in seconds, and f is the frequency of data capture for the target dataset. Therefore, for each dataset, we obtain $\{X \in \mathbb{R}^{N \times L}, y \in \{0,1\}^N\}$, where X are the sample signals, y are the targets, N is the number of samples, and L is the total length of each sample. We evaluate each dataset on all the tabular models using a five-fold-subject cross-validation. Fig. 3 shows boxplots of F_1 scores across the five folds for each tabular model (detailed results in Tables A4, A5, and A6 in Appendix E).

Table 2. Tabular Models Average Cross-validation Results

Dataset	Model	AUC	Precision (%)	Recall (%)	F_1 Score (%)
FARSEEING	**ExtraTrees**	99.2 ± 0.8	91.3 ± 11.3	82.6 ± 11.9	**85.9 ± 8.7**
	KNN	79.2 ± 8.8	77.1 ± 26.4	37.9 ± 13.3	47.5 ± 11.3
	LogisticCV	94.0 ± 3.7	87.2 ± 10.7	76.9 ± 14.4	81.3 ± 11.0
	RandomForest	99.0 ± 1.0	88.6 ± 12.0	83.4 ± 10.5	85.0 ± 6.7
	RidgeCV	96.0 ± 2.9	98.8 ± 2.8	67.5 ± 10.3	79.7 ± 7.5
FallAllD	**ExtraTrees**	98.8 ± 0.8	90.1 ± 7.7	92.9 ± 4.9	**91.2 ± 4.0**
	KNN	81.4 ± 7.5	84.5 ± 13.8	31.7 ± 12.0	44.6 ± 13.6
	LogisticCV	98.2 ± 0.8	89.2 ± 11.2	87.8 ± 11.4	87.6 ± 6.9
	RandomForest	98.6 ± 0.6	89.2 ± 8.9	89.7 ± 7.8	89.0 ± 5.2
	RidgeCV	98.4 ± 0.9	91.6 ± 8.6	82.9 ± 13.5	86.1 ± 7.7
SisFall	ExtraTrees	98.8 ± 0.8	91.1 ± 5.5	72.6 ± 8.5	80.4 ± 4.0
	KNN	95.2 ± 2.6	85.0 ± 4.9	77.4 ± 7.5	**80.9 ± 5.4**
	LogisticCV	75.4 ± 4.5	53.5 ± 27.1	12.3 ± 7.5	17.6 ± 10.3
	RandomForest	97.6 ± 1.1	91.2 ± 4.5	67.9 ± 8.0	77.5 ± 4.7
	RidgeCV	73.8 ± 3.5	94.3 ± 12.8	1.2 ± 1.1	2.4 ± 2.2

The best performing model and best F_1 score for each dataset is shown in **bold**.

Table 2 shows the mean cross-validation results achieved by tabular models on all datasets. *ExtraTrees* achieves the best result overall on FARSEEING and FallAllD, while *KNN* achieves the best performance on SisFall. Overall, the SisFall dataset is the most challenging for the tabular classifiers. This may be a result of the arbitrary positioning of the impact event in the SisFall fall samples.

4.3 Experiments With Time Series Models

After reshaping each dataset as $\{X \in \mathbb{R}^{N \times 1 \times L}, y \in \{0,1\}^N\}$, where X, y, N, and L are as previously defined, we performed fall detection using the *aeon toolkit* im-

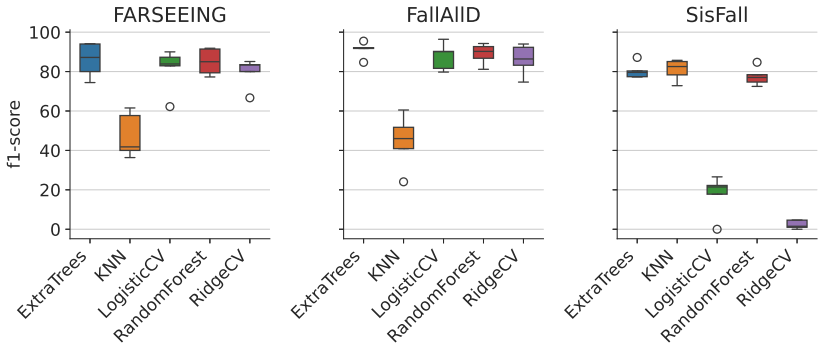

Fig. 3. Tabular Models Cross-validated F_1 scores.

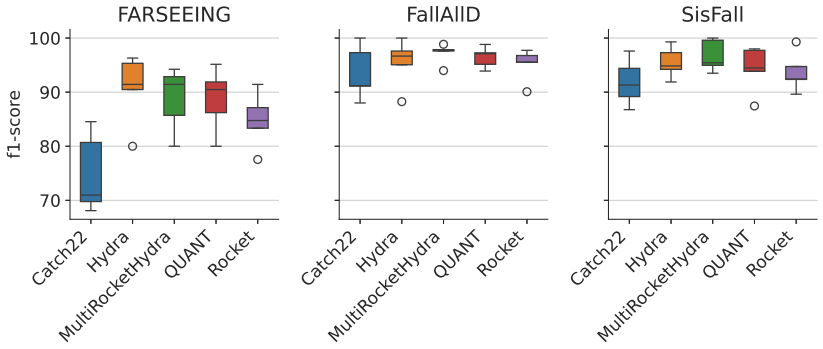

Fig. 4. Time Series Models Cross-validated F_1 scores.

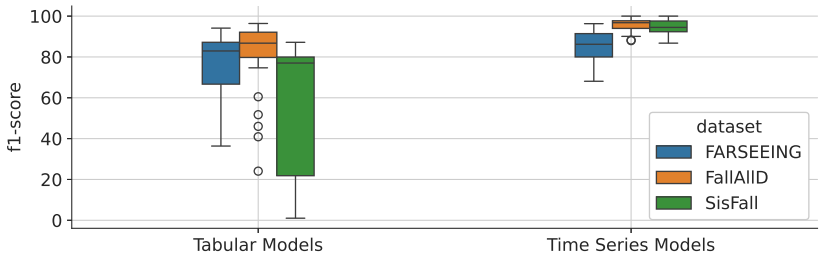

Fig. 5. Summary of all Cross-validated F_1 scores across all datasets.

Table 3. Time Series Models Average Cross-validation Results

Dataset	Model	AUC	Precision (%)	Recall (%)	F_1 Score (%)
FARSEEING	Catch22	96.6 ± 1.8	79.2 ± 10.4	72.0 ± 10.5	74.8 ± 7.3
	Hydra	94.0 ± 5.2	93.2 ± 3.9	89.1 ± 11.5	**90.7 ± 6.5**
	M.R.Hydra	93.0 ± 5.0	91.3 ± 5.8	87.4 ± 10.5	88.9 ± 5.9
	QUANT	99.4 ± 0.9	89.8 ± 6.8	88.7 ± 11.0	88.8 ± 5.9
	Rocket	90.4 ± 4.04	88.1 ± 7.26	82.5 ± 8.5	84.8 ± 5.1
FallAllD	Catch22	99.0 ± 1.0	92.2 ± 6.5	95.1 ± 5.2	93.5 ± 5.0
	Hydra	97.0 ± 3.0	95.4 ± 5.8	95.7 ± 4.2	95.5 ± 4.4
	M.R.Hydra	98.4 ± 0.6	95.9 ± 4.2	98.7 ± 1.8	**97.2 ± 1.9**
	QUANT	100.0 ± 0.0	95.9 ± 4.1	97.2 ± 2.5	96.5 ± 1.9
	Rocket	96.4 ± 2.2	96.2 ± 3.5	94.2 ± 4.6	95.1 ± 3.0
SisFall	Catch22	99.4 ± 0.6	94.5 ± 6.4	89.7 ± 6.0	91.9 ± 4.3
	Hydra	97.0 ± 2.1	96.7 ± 1.8	94.4 ± 4.9	95.5 ± 2.7
	M.R.Hydra	97.4 ± 2.4	99.2 ± 1.4	94.4 ± 4.8	**96.7 ± 2.9**
	QUANT	99.8 ± 0.5	95.6 ± 5.0	93.1 ± 4.1	94.3 ± 4.3
	Rocket	96.0 ± 2.8	95.5 ± 5.3	92.3 ± 6.2	93.7 ± 3.6

M.R.Hydra: MultiRocketHydra. The best-performing model and best F_1 score for each dataset is shown in **bold**.

plementations of five state-of-the-art time series classifiers: *Hydra* [10], *Rocket* [9], *MultiRocketHydra*, *Catch22* [28], and *Quant* [11]. Initial single train/test split experiments with deep learning models (*InceptionTime* [18] and *FCN* [56]) showed longer training times and poorer performance compared to other models. For example, FCN achieved an F_1 score of just 14%, taking 252 seconds, while Hydra achieved an F_1 score of 90% in just 49 seconds. InceptionTime's training was stopped after over 3 hours, which we deemed too long for real-time performance. Prioritising accuracy, efficiency, and real-time performance, we excluded FCN and InceptionTime from cross-validation experiments.

Table 3 shows that MultiRocketHydra achieves the best results on FallAllD and SisFall, while Hydra performs best on FARSEEING. Quant also performs well and is fastest among the compared methods. Fig. 5 demonstrates that time series models exhibit better and more consistent performance across individual datasets, with F_1 scores ranging from 68% - 100% (Fig. 4). This superior performance is expected, as these time series models are robust to variations in the positioning of the impact phase. Unlike tabular models, which treat each time step as an individual feature, time series models effectively capture the temporal dynamics and context of the signals. Detailed results for the time series models are shown in Tables A7, A8, and A9 in Appendix E.

4.4 Cross-Dataset Evaluation

We now investigate the transferability of time series models from simulated datasets to a real-world dataset by cross-evaluating each simulated dataset with FARSEEING. To align with FARSEEING's sensor specifications — 100Hz sampling frequency and accelerometer range of ±2g — we resampled FallAllD and SisFall to 100Hz and clipped their acceleration values accordingly. Standardisation was applied to all datasets.

In this experiment, we employ a single subject-wise train/test split for each dataset, reserving 33% (13 subjects) of the FARSEEING dataset for testing. We use this single test set $\mathcal{T}$ of 13 subjects exclusively from FARSEEING. We explored six training scenarios: training solely on the FARSEEING training set after excluding $\mathcal{T}$, using each of the FallAllD and SisFall training sets individually, individual combinations of the FallAllD and SisFall training sets with FARSEEING (FallAllD+ and SisFall+), and a combination of FARSEEING, FallAllD, and SisFall training sets (All). Each training set —FARSEEING, FallAllD, SisFall, FallAllD+, SisFall+, and All — was evaluated on the same test set $\mathcal{T}$ for a consistent and fair comparison (detailed results in Table A10, Appendix E).

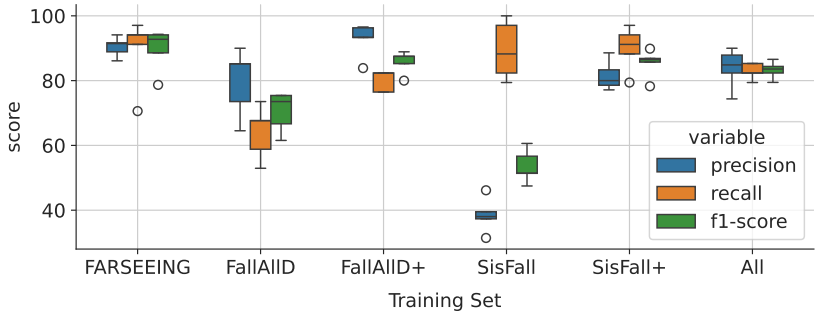

Fig. 6. Cross-dataset precision and recall obtained using time series techniques.

Fig. 6 illustrates that training on FallAllD and testing on FARSEEING generally led to low recall and F_1 scores but slightly higher precision. Conversely, training on SisFall and testing on FARSEEING resulted in low precision and F_1 scores, but higher recall. This disparity indicates that training on simulated datasets and testing on real-world data can cause imbalances between precision and recall, leading to either increased missed falls or increased false alarms. Combining simulated and real-world datasets (FallAllD+ and SisFall+) improved scores, with the combination of all three training sets giving a better balance between precision and recall and lower variance overall. This suggests a promising approach of augmenting real-world fall datasets with multiple simulated datasets.

4.5 Comparison With Related Work

We compare our best results with results obtained with other techniques evaluated on FARSEEING, FallAllD, and SisFall. On FARSEEING, we compare with [40], which uses manual feature extraction and ResNet [43] evaluated on 22 subjects. On FallAllD, comparisons include TCN-TAM [37], DeepFall [19], and CNN-GRU [25] employing convolutional neural networks and gated recurrent

units. On SisFall, comparisons are made with LSTM-CVAE [55], an RNN-based approach [36], and an LSTM-based method [30].

We report each study's cross-validation approach, recall, and F_1 score as documented by the respective authors (Table 4). Despite employing simple and efficient time series techniques without explicit feature extraction, our results are competitive with those achieved using deep learning models. However, this comparison is only indicative due to differences in preprocessing steps.

Table 4. Performance Comparison with Related Work

Method	Dataset	CV	Recall (%)	F_1 Score (%)
Feature-based [40]	FARSEEING	5-fold subject CV	81.1	64.6
ResNet* [43]		NR	94.0	91.0
Ours (proposed)		5-fold subject CV	89.1 ± 11.5	90.7 ± 6.5
TCN-TAM [37]	FallAllD	NR	88.0	89.0
DeepFall [19]		5-fold	NR	92.8
CNN and GRU [25]		LOSO	92.5	94.3
Ours (proposed)		5-fold subject CV	98.7 ± 1.8	97.2 ± 1.9
LSTM-CVAE [55]	SisFall	NR	99.0	95.0
RNN [36]		NR	96.7	NR
LSTM [30]		5-fold	96.4	NR
Ours (proposed)		5-fold subject CV	94.4 ± 4.8	96.7 ± 2.9

*This approach was trained and tested on only a subset of the dataset (22 subjects).
NR: Not reported. **LOSO**: Leave-One-Subject-Out cross-validation.

As shown in Table A1 (Appendix A), the *Hydra* model, which performs best on average on the FARSEEING dataset, achieves a false alarm rate of 0.012 per hour and a miss rate of 0.013 per hour. This corresponds to ≈ 28% chance of one false alarm and a 31% chance of one missed fall per day. Therefore, a deployed Hydra model could have as few as 1 false alarm and 1 missed fall in 3 days.

5 Conclusion

In this work, we presented a framework for real-world fall detection using state-of-the-art time series techniques applied to real-world fall data. Our approach is simple, efficient, and requires no manual feature engineering. We performed segmentation in a manner that imitates real-time signals and includes the pre-fall and post-fall contexts, with about 1 false alarm and 1 false negative in 3 days. While our primary dataset was real-world, we also evaluated our method on two simulated datasets and achieved comparable results. Experimentation with both tabular and time series models across all datasets consistently demonstrated superior performance of time series models. We also performed cross-dataset evaluation and found significant reductions in precision and recall when models trained on simulated data were applied to real-world falls. However, we found that extending real-world falls with multiple simulated datasets is a promising

approach. Finally, we used a post-hoc explanation technique to highlight segments of the motion time series that are relevant for each classifier (Appendix C).

We hereby propose four main practical considerations for fall detection, namely, *reality*, *accuracy*, *timeliness*, and *explainability* (**RATE**). A *realistic* fall detection algorithm should be validated, at least partially, using real-world falls, possibly through a blend of simulated and real-world data. For *accuracy*, algorithms must strike a balance between precision and recall, considering that missed falls and false alarms are both undesirable. Real-time detection (*timeliness*) is crucial to prevent prolonged immobility after a fall, known as "long-lie," which can lead to severe consequences. Finally, we suggest incorporating *explanations* and online learning mechanisms to refine models based on segments that contribute to false alarms and false negatives.

Although our proposed fall detection framework has good **RATE** characteristics, there are a few limitations that will be addressed in future work. Currently, we use a simple univariate time series approach for compatibility with tabular baseline models. Future work will explore a multivariate approach using all accelerometer axes and additional signals from the magnetometer and gyroscope sensors. Deep learning models were excluded for efficiency reasons, but we plan to investigate recent efficient models like LITE [17] and explore the use of reasonable contract times for training models (see [13]). Lastly, the scarcity of publicly available real-world fall datasets is a challenge. Future efforts will focus on curating and releasing open-source real-world fall datasets to advance fall detection research.

Acknowledgments. This study has received funding from Science Foundation Ireland [12/RC/2289_P2] at the Insight SFI Research Centre for Data Analytics and the European Union's H2020 Marie Skłodowska-Curie Cofund programme, NeuroInsight [Grant ID: 101034252].

References

1. World Health Organization (Apr 2021), https://www.who.int/publications/i/item/978924002191-4
2. Becker, C., Schwickert, L., Mellone, S., Bagalà, F., Chiari, L., Helbostad, J.L., Zijlstra, W., Aminian, K., Bourke, A., Todd, C., et al.: Proposal for a multiphase fall model based on real-world fall recordings with body-fixed sensors. Zeitschrift fur Gerontologie und Geriatrie **45**(8), 707 (2012)
3. Beddiar, D.R., Oussalah, M., Nini, B.: Fall detection using body geometry and human pose estimation in video sequences. Journal of Visual Communication and Image Representation **82**, 103407 (2022)
4. Borrelli, J., Creath, R.A., Rogers, M.W.: A method for simulating forward falls and controlling impact velocity. MethodsX **11**, 102399–102399 (Dec 2023)
5. Camp, K., Murphy, S., Pate, B.: Integrating fall prevention strategies into ems services to reduce falls and associated healthcare costs for older adults. Clinical interventions in aging pp. 561–569 (2024)
6. Chen, J., Kwong, K., Chang, D., Luk, J., Bajcsy, R.: Wearable sensors for reliable fall detection. In: 2005 IEEE engineering in medicine and biology 27th annual conference. pp. 3551–3554. IEEE (2006)

7. Choi, A., Kim, T.H., Yuhai, O., Jeong, S., Kim, K., Kim, H., Mun, J.H.: Deep learning-based near-fall detection algorithm for fall risk monitoring system using a single inertial measurement unit. IEEE transactions on neural systems and rehabilitation engineering **30**, 2385–2394 (2022)
8. Davide Italo Serramazza, Thach Le Nguyen, G.I.: tscaptum: adapting captum explainers for time series and scikit-learn-like predictors. GitHub (2024), https://github.com/mlgig/tscaptum, temporary bibtex entry
9. Dempster, A., Petitjean, F., Webb, G.I.: Rocket: exceptionally fast and accurate time series classification using random convolutional kernels. Data Mining and Knowledge Discovery **34**(5), 1454–1495 (2020)
10. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: competing convolutional kernels for fast and accurate time series classification. Data mining and knowledge discovery **37**(5), 1779–1805 (May 2023)
11. Dempster, A., Schmidt, D.F., Webb, G.I.: Quant: A minimalist interval method for time series classification. Data Mining and Knowledge Discovery pp. 1–26 (2024)
12. Dhariyal, B., Le Nguyen, T., Ifrim, G.: Back to basics: A sanity check on modern time series classification algorithms. In: Ifrim, G., Tavenard, R., Bagnall, A., Schaefer, P., Malinowski, S., Guyet, T., Lemaire, V. (eds.) Advanced Analytics and Learning on Temporal Data. pp. 205–229. Springer Nature Switzerland, Cham (2023)
13. Flynn, M., Large, J., Bagnall, T.: The contract random interval spectral ensemble (c-rise): The effect of contracting a classifier on accuracy. In: Hybrid Artificial Intelligent Systems: 14th International Conference, HAIS 2019, León, Spain, September 4–6, 2019, Proceedings 14. pp. 381–392. Springer (2019)
14. Hawkins, W.: Fourier transform resampling: theory and application [medical imaging]. In: 1996 IEEE Nuclear Science Symposium. Conference Record. vol. 3, pp. 1491–1495 vol.3 (1996)
15. He, J., Zhang, Z., Wang, X., Yang, S.: A low power fall sensing technology based on fd-cnn. IEEE Sensors Journal **19**(13), 5110–5118 (2019)
16. Hu, X., Chu, L., Pei, J., Liu, W., Bian, J.: Model complexity of deep learning: A survey. Knowledge and Information Systems **63**, 2585–2619 (2021)
17. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Lite: Light inception with boosting techniques for time series classification. In: 2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA). pp. 1–10 (2023)
18. Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D.F., Weber, J., Webb, G.I., Idoumghar, L., Muller, P.A., Petitjean, F.: Inceptiontime: Finding alexnet for time series classification. Data Mining and Knowledge Discovery **34**(6), 1936–1962 (2020)
19. Jitpattanakul, A.: Wearable fall detection based on motion signals using hybrid deep residual neural network. In: Multi-disciplinary Trends in Artificial Intelligence: 15th International Conference, MIWAI 2022, Virtual Event, November 17–19, 2022, Proceedings. vol. 13651, p. 216. Springer Nature (2022)
20. Khan, S.S., Hoey, J.: Review of fall detection techniques: A data availability perspective. Medical engineering & physics **39**, 12–22 (2017)
21. Klenk, J., Schwickert, L., Palmerini, L., Mellone, S., Bourke, A., Ihlen, E.A., Kerse, N., Hauer, K., Pijnappels, M., Synofzik, M., et al.: The farseeing real-world fall repository: a large-scale collaborative database to collect and share sensor signals from real-world falls. European review of aging and physical activity **13**, 1–7 (2016)

22. La Blunda, L., Gutierrez-Madronal, L., Wagner, M.F., Medina-Bulo, I.: A wearable fall detection system based on body area networks. IEEE Access **8**, 193060–193074 (2020)
23. Le, H.L., Nguyen, D.N., Nguyen, T.H., Nguyen, H.N.: A novel feature set extraction based on accelerometer sensor data for improving the fall detection system. Electronics **11**(7), 1030 (2022)
24. Liu, C.P., Li, J.H., Chu, E.P., Hsieh, C.Y., Liu, K.C., Chan, C.T., Tsao, Y.: Deep learning-based fall detection algorithm using ensemble model of coarse-fine cnn and gru networks. In: 2023 IEEE International Symposium on Medical Measurements and Applications (MeMeA). pp. 1–5. IEEE (2023)
25. Liu, C.P., Li, J.H., Chu, E.P., Hsieh, C.Y., Liu, K.C., Chan, C.T., Tsao, Y.: Deep learning-based fall detection algorithm using ensemble model of coarse-fine cnn and gru networks. In: 2023 IEEE International Symposium on Medical Measurements and Applications (MeMeA). pp. 1–5. IEEE (2023)
26. Liu, J., Li, X., Huang, S., Chao, R., Cao, Z., Wang, S., Wang, A., Liu, L.: A review of wearable sensors based fall-related recognition systems. Engineering Applications of Artificial Intelligence **121**, 105993 (2023)
27. Liu, K.C., Hung, K.H., Hsieh, C.Y., Huang, H.Y., Chan, C.T., Tsao, Y.: Deep-learning-based signal enhancement of low-resolution accelerometer for fall detection systems. IEEE Transactions on Cognitive and Developmental Systems **14**(3), 1270–1281 (2021)
28. Lubba, C.H., Sethi, S.S., Knaute, P., Schultz, S.R., Fulcher, B.D., Jones, N.S.: catch22: Canonical time-series characteristics: Selected through highly comparative time-series analysis. Data Mining and Knowledge Discovery **33**(6), 1821–1852 (2019)
29. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 4768–4777. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
30. Magalhães, C., Ribeiro, J., Leite, A., Solteiro Pires, E., Pavão, J.: Automatic fall detection using long short-term memory network. In: International Work-Conference on Artificial Neural Networks. pp. 359–371. Springer (2021)
31. Mekruksavanich, S., Jantawong, P., Hnoohom, N., Jitpattanakul, A.: Wearable fall detection based on motion signals using hybrid deep residual neural network. In: International Conference on Multi-disciplinary Trends in Artificial Intelligence. pp. 216–224. Springer (2022)
32. Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Rubio, D.G., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., et al.: aeon: a python toolkit for learning from time series. arXiv preprint arXiv:2406.14231 (2024)
33. Montero-Odasso, M., Van Der Velde, N., Martin, F.C., Petrovic, M., Tan, M.P., Ryg, J., Aguilar-Navarro, S., Alexander, N.B., Becker, C., Blain, H., et al.: World guidelines for falls prevention and management for older adults: a global initiative. Age and ageing **51**(9), afac205 (2022)
34. Mosquera-Lopez, C., Wan, E., Shastry, M., Folsom, J., Leitschuh, J., Condon, J., Rajhbeharrysingh, U., Hildebrand, A., Cameron, M., Jacobs, P.G.: Automated detection of real-world falls: Modeled from people with multiple sclerosis. IEEE journal of biomedical and health informatics **25**(6), 1975–1984 (2020)
35. Mosquera-Lopez, C., Wan, E., Shastry, M., Folsom, J., Leitschuh, J., Condon, J., Rajhbeharrysingh, U., Hildebrand, A., Cameron, M., Jacobs, P.G.: Automated detection of real-world falls: Modeled from people with multiple sclerosis. IEEE Journal of Biomedical and Health Informatics **25**(6), 1975–1984 (2021)

36. Musci, M., De Martini, D., Blago, N., Facchinetti, T., Piastra, M.: Online fall detection using recurrent neural networks on smart wearable devices. IEEE Transactions on Emerging Topics in Computing **9**(3), 1276–1289 (2020)
37. Nguyen, D.A., Pham, C., Argent, R., Caulfield, B., Le-Khac, N.A.: Model and empirical study on multi-tasking learning for human fall detection (2024)
38. Nho, Y.H., Ryu, S., Kwon, D.S.: Ui-gan: Generative adversarial network-based anomaly detection using user initial information for wearable devices. IEEE Sensors Journal **21**(8), 9949–9958 (2021)
39. Özdemir, A.T.: An analysis on sensor locations of the human body for wearable fall detection devices: Principles and practice. Sensors **16**(8), 1161 (2016)
40. Palmerini, L., Klenk, J., Becker, C., Chiari, L.: Accelerometer-based fall detection using machine learning: Training and testing on real-world falls. Sensors **20**(22), 6479 (2020)
41. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
42. Pernes, M., Agostinho, I., Bernardes, R.A., Fernandes, J.B., Baixinho, C.L.: Documenting fall episodes: a scoping review. Frontiers in public health **11** (May 2023)
43. Ramanathan, A., McDermott, J.: Fall detection with accelerometer data using residual networks adapted to multi-variate time series classification. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2021)
44. Rastogi, S., Singh, J.: A systematic review on machine learning for fall detection system. Computational intelligence **37**(2), 951–974 (2021)
45. Saleh, M., Abbas, M., Le Jeannes, R.B.: Fallalld: An open dataset of human falls and activities of daily living for classical and deep learning applications. IEEE Sensors Journal **21**(2), 1849–1858 (2020)
46. Serramazza, D., Le Nguyen, T., Ifrim, G.: Improving the evaluation and actionability of explanation methods for multivariate time series classification. In: 2024 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (06 2024)
47. Silva, C.A., Casilari, E., García-Bermúdez, R.: Cross-dataset evaluation of wearable fall detection systems using data from real falls and long-term monitoring of daily life. Measurement **235**, 114992 (2024)
48. Son, H., Lim, J.W., Park, S., Park, B., Han, J., Kim, H.B., Lee, M.C., Jang, K.J., Kim, G., Chung, J.H.: A machine learning approach for the classification of falls and activities of daily living in agricultural workers. IEEE Access **10**, 77418–77431 (2022)
49. de Sousa, F.A.S.F., Escriba, C., Bravo, E.G.A., Brossa, V., Fourniols, J.Y., Rossi, C.: Wearable pre-impact fall detection system based on 3d accelerometer and subject's height. IEEE Sensors Journal **22**(2), 1738–1745 (2021)
50. Stack, E.: Falls are unintentional: Studying simulations is a waste of faking time. Journal of Rehabilitation and Assistive Technologies Engineering **4**, 2055668317732945 (2017)
51. Sucerquia, A., López, J.D., Vargas-Bonilla, J.F.: Sisfall: A fall and movement dataset. Sensors **17**(1), 198 (2017)
52. Vaishya, R., Vaish, A.: Falls in older adults are serious. Indian journal of orthopaedics **54**, 69–74 (2020)
53. Wang, G., Li, Q., Wang, L., Zhang, Y., Liu, Z.: Cmfall: A cascade and parallel multi-state fall detection algorithm using waist-mounted tri-axial accelerometer signals. IEEE Transactions on Consumer Electronics **66**(3), 261–270 (2020)

54. Wu, X., Zheng, Y., Chu, C.H., Cheng, L., Kim, J.: Applying deep learning technology for automatic fall detection using mobile sensors. Biomedical Signal Processing and Control **72**, 103355 (2022)
55. Yi, M.K., Han, K., Hwang, S.O.: Fall detection of the elderly using denoising lstm-based convolutional variant autoencoder. IEEE Sensors Journal (2024)
56. Zhao, B., Lu, H., Chen, S., Liu, J., Wu, D.: Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics **28**(1), 162–169 (2017)

Appendix A Practical Considerations

In this section, we discuss the practical usability of the proposed models based on our segmentation technique, model recall, and specificity. Since we use an overlapping sliding window with a step size of 1 second, each classifier would process a total of $1 \times 60(secs) \times 60(mins) = 3600$ samples per hour in practice. Taking the *Hydra* classifier as an example, we obtain a rough estimate of false alarm and miss rates per hour (Table A1) in the following manner.

Let P be the total number of falls, and N be the total number of ADLs per hour, such that $P + N = 3600$. From the Fall : ADL ratio in Table 1, we know that

$$P = 0.14N \tag{1}$$

So that

$$0.14N + N = 3600 \tag{2}$$

Hence, we can estimate the number of ADLs per hour as $N = \frac{3600}{1.14} \approx 3158$, and the number of falls $P = 3600 - 3158 = 442$ per hour. Therefore, we estimate the number of misses per hour, FN as

$$FN = P \times (1 - Recall) \tag{3}$$

$$FN = 442 \times (1 - 0.8913) \tag{4}$$

$$FN \approx 48 \tag{5}$$

Similarly, false alarms per hour, FP:

$$FP = N \times (1 - Specificity) \tag{6}$$

$$FP = 3158 \times (1 - 0.986) \tag{7}$$

$$FP \approx 44 \tag{8}$$

Hence, miss rate $= \frac{48}{3600} \approx 0.013$, and false alarm rate $= \frac{44}{3600} \approx 0.012$ per hour.

Table A1. False Alarm Rate and Miss Rate Per Hour

Model	Ave. Recall	Ave. Specificity	False Alarm Rate	Miss Rate
Catch22	72.04	96.55	0.030	0.034
Hydra	89.13	98.60	0.012	0.013
MultiRocketHydra	87.37	98.31	0.015	0.016
QUANT	88.74	98.34	0.015	0.014
Rocket	82.52	98.50	0.013	0.021

Appendix B Effect of Post-fall Phase

In this section, we train the time series models on the FARSEEING dataset with total sample window sizes ranging from 3 seconds (corresponding to a 1-second post-fall phase) to 27 seconds (corresponding to a 25-second post-fall phase. As shown in Fig. A1, longer post-fall phases lead to longer inference time per sample (as expected) and higher AUC. However, longer post-fall phases do not necessarily offer improvement in terms of recall, specificity, and F_1 scores. On the contrary, longer post-fall phases may negatively impact performance while also increasing runtime. According to expert advice, 5 seconds is sufficient to capture the post-fall context.

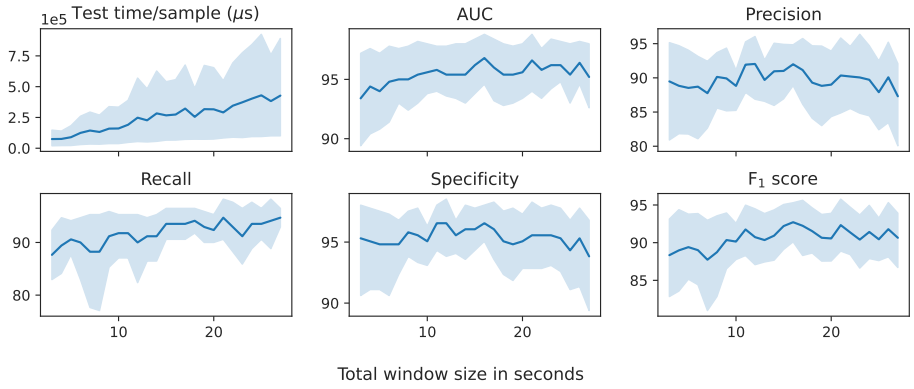

Fig. A1. Effect of post-fall window size on model performance on the FARSEEING dataset.

Appendix C Model Explanation

In this section, we use *tscaptum* [8, 46] to identify specific time intervals within motion data that are critical for classifier decisions using the FARSEEING dataset. As a post-hoc explanation technique, *tscaptum* employs a chunking procedure to aggregate adjacent time points into segments of size c to enhance robustness and reduce runtime. Important time points are identified by iteratively masking signal segments. Then, SHAP (SHapley Additive exPlanations) scores [29] are computed for each segment based on its impact on prediction outcomes.

First, we conduct a single subject-wise train-test split, allocating 30% (13 subjects) for testing and training models on the remaining 70% (28 subjects) of the FARSEEING dataset (see Fig. A2). Subsequently, temporal SHAP attribution scores are obtained for predictions on the test set using *tscaptum*. Here, we

set $c = 100$, corresponding to a 1-second interval in the FARSEEING dataset (sampled at 100Hz). Thus, attribution scores for each time point within each chunk are equally distributed, resulting in an attribution profile with the same shape as the input sample. We show the attribution profiles for representative samples of true positives, false positives, true negatives, and false negatives across each classifier.

Similarities in attribution profiles are observed between true falls and false alarms, as well as between true ADLs and missed falls. Notably, while the impact phase occurs within $t = [1, 2)$, attribution scores for *Hydra* (Fig. A3) and *MultiRocketHydra* (Fig. A5) indicate nearly uniform importance across all signal phases. However, *Rocket* (Fig. A4), *Catch22* (Fig. A6), and *QUANT* (Fig. A7) show high attribution scores between the end of the falling phase and the start of the impact phase.

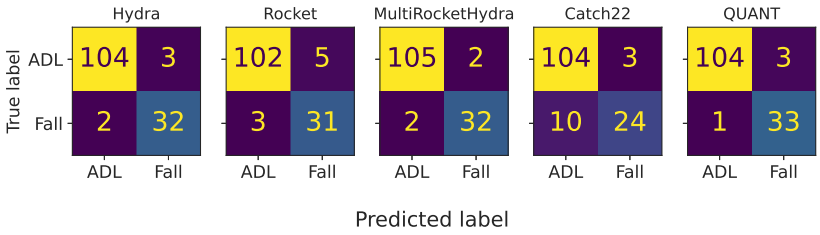

Fig. A2. Confusion matrices for time series classifiers on the FARSEEING test set.

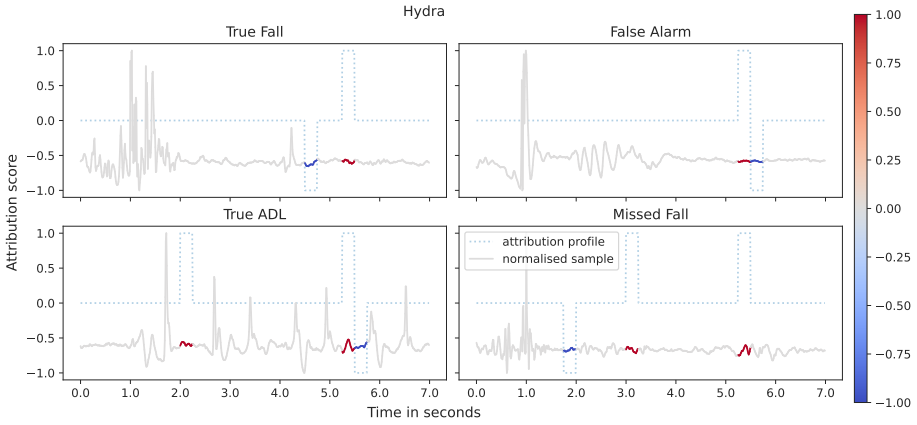

Fig. A3. Hydra + SHAP temporal attribution profiles for some samples.

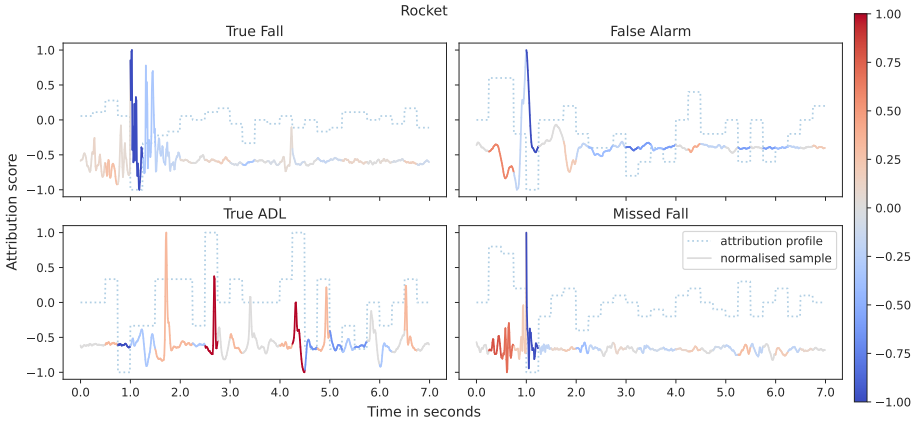

Fig. A4. Rocket + SHAP temporal attribution profiles for some samples.

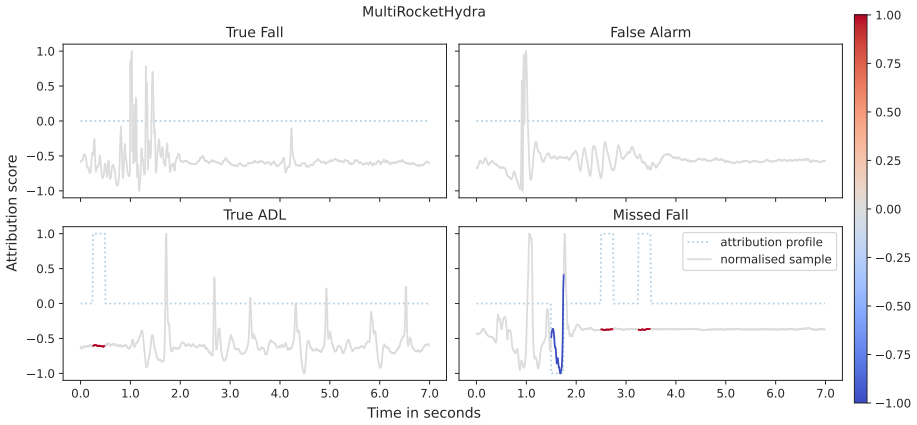

Fig. A5. MultiRocketHydra + SHAP temporal attribution profiles for some samples.

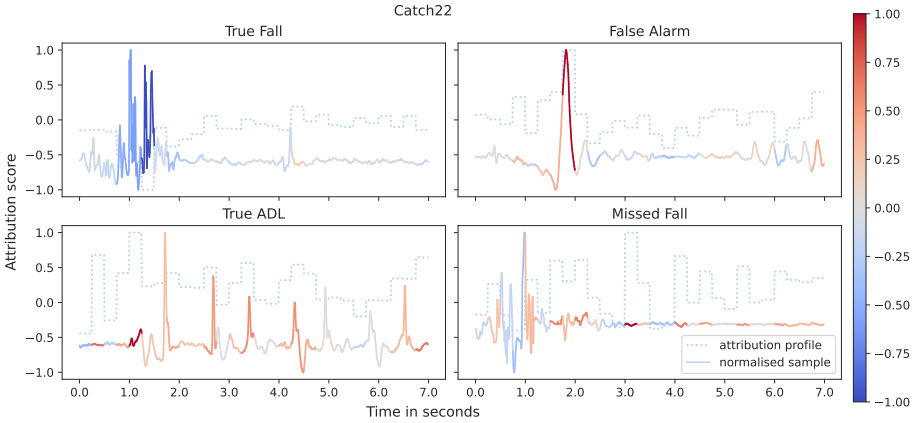

Fig. A6. Catch22 + SHAP temporal attribution profiles for some samples.

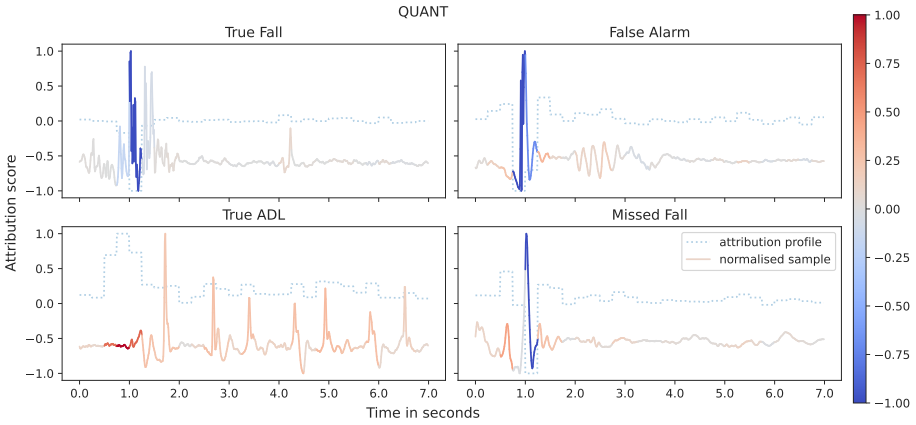

Fig. A7. QUANT + SHAP temporal attribution profiles for some samples.

Appendix D Average Model Runtime

In our experiments, we compute runtime in milliseconds (ms) as the time it takes each model to perform inference on one sample. As shown in Table A2, the tabular models run extremely fast. However, the time series models also run fast enough to be deployed in real-time, with *MultiRocketHydra* having the slowest inference speed of 144 milliseconds per sample in one instance (see Fig. A8).

Table A2. Average Model Inference Runtime

Model	Average Runtime (ms)
RidgeCV	0.03
LogisticCV	0.08
RandomForest	0.16
KNN	0.27
ExtraTrees	0.43
Rocket	10.24
QUANT	10.40
Catch22	14.04
Hydra	48.02
MultiRocketHydra	54.91

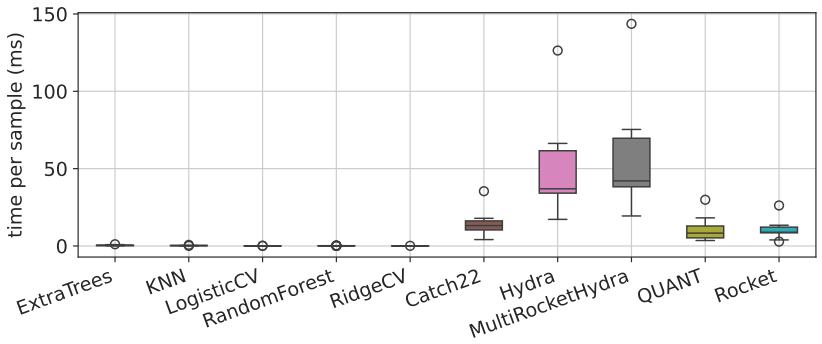

Fig. A8. Inference time per sample in milliseconds for all models

Appendix E Extended Cross-validation Results

Here, we give details of the cross-validation results, including split sizes (Table A3) and the performance of each classifier on individual splits of each dataset. More metrics are detailed here, including the inference time per sample (T) in milliseconds, AUC, precision, recall, specificity, and F_1 score.

Table A3. Cross-validation splits

Dataset	Fold	Train Set				Test Set			
		Subjects	ADLs	Falls	Fall : ADL	Subjects	ADLs	Falls	Fall : ADL
FARSEEING	0	33	865	118	0.14	8	304	27	0.09
	1	33	684	119	0.17	8	485	26	0.05
	2	33	1094	125	0.11	8	75	20	0.27
	3	33	1039	127	0.12	8	130	18	0.14
	4	33	994	92	0.09	8	175	53	0.30
FallAllD	0	12	1082	373	0.34	2	197	93	0.47
	1	12	1077	388	0.36	2	202	78	0.39
	2	12	1058	403	0.38	2	221	63	0.29
	3	12	1090	399	0.37	2	189	67	0.35
	4	12	1199	423	0.35	2	80	43	0.54
SisFall	0	31	8957	1004	0.11	7	1886	196	0.10
	1	31	8883	847	0.10	7	1960	353	0.18
	2	31	8706	993	0.11	7	2137	207	0.10
	3	31	8720	1073	0.12	7	2123	127	0.06
	4	31	8945	937	0.10	7	1898	263	0.14

Table A4. Performance of Tabular Models on the FARSEEING Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
ExtraTrees	0.40	99.0	72.7	88.9	97.0	80.0	0
	0.30	99.0	94.1	61.5	99.8	74.4	1
	1.10	98.0	89.5	85.0	97.3	87.2	2
	0.70	100.0	100.0	88.9	100.0	94.1	3
	0.50	100.0	100.0	88.7	100.0	94.0	4
KNN	0.10	80.0	60.0	55.6	96.7	57.7	0
	0.10	78.0	85.7	23.1	99.8	36.4	1
	0.30	65.0	40.0	40.0	84.0	40.0	2
	0.20	86.0	100.0	44.4	100.0	61.5	3
	0.10	87.0	100.0	26.4	100.0	41.8	4
LogisticCV	0.10	97.0	85.7	88.9	98.7	87.3	0
	0.00	88.0	73.7	53.9	99.0	62.2	1
	0.10	93.0	81.0	85.0	94.7	82.9	2
	0.10	95.0	100.0	72.2	100.0	83.9	3
	0.10	97.0	95.7	84.9	98.9	90.0	4
RandomForest	0.10	99.0	69.4	92.6	96.4	79.4	0
	0.00	98.0	94.4	65.4	99.8	77.3	1
	0.20	98.0	85.0	85.0	96.0	85.0	2
	0.10	100.0	94.1	88.9	99.2	91.4	3
	0.10	100.0	100.0	84.9	100.0	91.8	4
RidgeCV	0.00	97.0	100.0	74.1	100.0	85.1	0
	0.00	91.0	100.0	50.0	100.0	66.7	1
	0.00	96.0	93.8	75.0	98.7	83.3	2
	0.10	98.0	100.0	66.7	100.0	80.0	3
	0.00	98.0	100.0	71.7	100.0	83.5	4

Table A5. Performance of Tabular Models on the FallAllD Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
	0.51	100.0	98.8	86.0	99.5	92.0	0
	0.51	99.0	87.4	97.4	94.6	92.1	1
ExtraTrees	0.47	98.0	78.4	92.1	92.8	84.7	2
	0.50	98.0	92.4	91.0	97.4	91.7	3
	0.87	99.0	93.3	97.7	96.3	95.5	4
	0.25	76.0	86.7	14.0	99.0	24.1	0
	0.2	88.0	87.8	46.2	97.5	60.5	1
KNN	0.25	75.0	62.2	36.5	93.7	46.0	2
	0.27	77.0	85.7	26.9	98.4	40.9	3
	0.69	91.0	100.0	34.9	100.0	51.7	4
	0.08	99.0	96.9	67.7	99.0	79.8	0
	0.08	98.0	86.9	93.6	94.6	90.1	1
LogisticCV	0.08	97.0	71.4	95.2	89.1	81.6	2
	0.08	98.0	90.9	89.6	96.8	90.2	3
	0.17	99.0	100.0	93.0	100.0	96.4	4
	0.15	99.0	98.6	77.4	99.5	86.8	0
	0.16	99.0	88.4	97.4	95.1	92.7	1
RandomForest	0.16	98.0	74.7	88.9	91.4	81.2	2
	0.17	98.0	90.9	89.6	96.8	90.2	3
	0.35	99.0	93.2	95.4	96.3	94.3	4
	0.03	99.0	98.3	60.2	99.5	74.7	0
	0.03	99.0	92.3	92.3	97.0	92.3	1
RidgeCV	0.03	97.0	77.0	90.5	92.3	83.2	2
	0.03	98.0	93.1	80.6	97.9	86.4	3
	0.04	99.0	97.5	90.7	98.8	94.0	4

Table A6. Performance of Tabular Models on the SisFall Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
	0.14	99.0	99.2	63.3	100.0	77.3	0
	0.12	99.0	89.8	84.7	98.3	87.2	1
ExtraTrees	0.13	98.0	86.9	73.4	98.9	79.6	2
	0.13	100.0	85.7	75.6	99.3	80.3	3
	0.13	98.0	94.0	65.8	99.4	77.4	4
	0.31	96.0	90.3	76.0	99.2	82.6	0
	0.28	98.0	87.9	82.4	98.0	85.1	1
KNN	0.29	92.0	85.2	72.5	98.8	78.3	2
	0.30	97.0	84.1	87.4	99.0	85.7	3
	0.31	93.0	77.4	68.8	97.2	72.8	4
	0.07	70.0	33.3	15.8	96.7	21.5	0
	0.07	82.0	100.0	0.0	100.0	0.0	1
LogisticCV	0.07	77.0	42.6	19.3	97.5	26.6	2
	0.07	75.0	37.7	15.8	98.5	22.2	3
	0.07	73.0	53.9	10.7	98.7	17.8	4
	0.17	98.0	98.3	60.2	99.9	74.7	0
	0.15	98.0	90.9	79.3	98.6	84.7	1
RandomForest	0.15	97.0	86.2	69.6	98.9	77.0	2
	0.16	99.0	89.0	70.1	99.5	78.4	3
	0.15	96.0	91.3	60.1	99.2	72.5	4
	0.02	72.0	100.0	0.5	100.0	1.0	0
	0.02	78.0	100.0	0.0	100.0	0.0	1
RidgeCV	0.02	77.0	71.4	2.4	99.9	4.7	2
	0.02	70.0	100.0	2.4	100.0	4.6	3
	0.02	72.0	100.0	0.8	100.0	1.5	4

Table A7. Performance of Time Series Models on the FARSEEING Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
	5.8	96.0	76.7	85.2	97.7	80.7	0
	4.1	95.0	76.2	61.5	99.0	68.1	1
Catch22	17.9	95.0	65.2	75.0	89.3	69.8	2
	9.3	99.0	84.6	61.1	98.5	71.0	3
	7.6	98.0	93.2	77.4	98.3	84.5	4
	18.2	98.0	96.3	96.3	99.7	96.3	0
	17.2	85.0	94.7	69.2	99.8	80.0	1
Hydra	61.4	96.0	86.4	95.0	96.0	90.5	2
	37.2	94.0	94.1	88.9	99.2	91.4	3
	30.3	97.0	94.4	96.2	98.3	95.3	4
	25.1	98.0	89.7	96.3	99.0	92.9	0
	19.4	85.0	94.7	69.2	99.8	80.0	1
MultiRocketHydra	69.7	92.0	81.8	90.0	94.7	85.7	2
	44.9	94.0	94.1	88.9	99.2	91.4	3
	33.3	96.0	96.1	92.5	98.9	94.2	4
	5.4	100.0	80.7	92.6	98.0	86.2	0
	7.9	99.0	94.7	69.2	99.8	80.0	1
QUANT	18.2	98.0	86.4	95.0	96.0	90.5	2
	8.3	100.0	89.5	94.4	98.5	91.9	3
	10.5	100.0	98.0	92.5	99.4	95.2	4
	3.9	95.0	78.1	92.6	97.7	84.8	0
	2.8	86.0	82.6	73.1	99.2	77.6	1
Rocket	12.6	87.0	93.8	75.0	98.7	83.3	2
	8.4	94.0	94.1	88.9	99.2	91.4	3
	5.9	90.0	91.7	83.0	97.7	87.1	4

Table A8. Performance of Time Series Models on the FallAllD Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
	15.9	100.0	97.8	96.8	99.0	97.3	0
	15.5	99.0	84.6	98.7	93.1	91.1	1
Catch22	16.4	98.0	88.7	87.3	96.8	88.0	2
	16.0	98.0	89.9	92.5	96.3	91.2	3
	35.4	100.0	100.0	100.0	100.0	100.0	4
	63.8	97.0	100.0	93.6	100.0	96.7	0
	61.7	98.0	91.7	98.7	96.5	95.1	1
Hydra	60.1	98.0	98.4	96.8	99.6	97.6	2
	66.3	92.0	87.0	89.6	95.2	88.2	3
	126.4	100.0	100.0	100.0	100.0	100.0	4
	67.9	98.0	98.9	96.8	99.5	97.8	0
	75.3	98.0	88.6	100.0	95.1	94.0	1
MultiRocketHydra	69.6	98.0	98.4	96.8	99.6	97.6	2
	74.6	99.0	95.7	100.0	98.4	97.8	3
	143.6	99.0	97.7	100.0	98.8	98.9	4
	9.4	100.0	98.9	95.7	99.5	97.3	0
	13.5	100.0	89.5	98.7	95.5	93.9	1
QUANT	12.3	100.0	96.7	93.7	99.1	95.2	2
	17.3	100.0	94.4	100.0	97.9	97.1	3
	29.9	100.0	100.0	97.7	100.0	98.8	4
	11.9	96.0	100.0	91.4	100.0	95.5	0
	12.1	97.0	94.9	96.2	98.0	95.5	1
Rocket	12.2	97.0	98.4	95.2	99.6	96.8	2
	13.4	93.0	92.2	88.1	97.4	90.1	3
	26.2	99.0	95.6	100.0	97.5	97.7	4

Table A9. Performance of Time Series Models on the SisFall Dataset

Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score	Fold
Catch22	12.7	99.0	83.5	90.3	98.1	86.8	0
	11.5	100.0	97.5	97.7	99.5	97.6	1
	12.5	99.0	95.6	83.6	99.6	89.2	2
	16.9	100.0	95.9	92.9	99.8	94.4	3
	13.2	99.0	100.0	84.0	100.0	91.3	4
Hydra	36.6	97.0	95.8	93.9	99.6	94.9	0
	35.5	99.0	99.7	98.9	100.0	99.3	1
	32.8	94.0	96.8	87.4	99.7	91.9	2
	35.9	99.0	95.5	99.2	99.7	97.3	3
	37.0	96.0	95.7	92.8	99.4	94.2	4
MultiRocketHydra	41.3	96.0	98.9	91.3	99.9	95.0	0
	38.1	100.0	100.0	100.0	100.0	100.0	1
	38.4	95.0	96.9	90.3	99.7	93.5	2
	40.4	100.0	100.0	99.2	100.0	99.6	3
	42.1	96.0	100.0	91.3	100.0	95.4	4
QUANT	4.4	100.0	94.3	93.4	99.4	93.9	0
	4.4	100.0	97.7	97.7	99.6	97.7	1
	3.5	99.0	87.4	87.4	98.8	87.4	2
	5.1	100.0	100.0	96.1	100.0	98.0	3
	5.9	100.0	98.4	90.9	99.8	94.5	4
Rocket	9.2	95.0	88.9	90.3	98.8	89.6	0
	8.7	99.0	100.0	98.6	100.0	99.3	1
	8.5	94.0	97.8	87.4	99.8	92.4	2
	8.7	99.0	90.7	99.2	99.4	94.7	3
	8.9	93.0	100.0	85.9	100.0	92.4	4

Table A10. Results of Cross-Dataset Evaluation

Training Set	Model	T (ms)	AUC	Precision	Recall	Specificity	F_1 Score
FARSEEING	Hydra	47.7	96.0	91.4	94.1	97.2	92.8
	Rocket	8.7	93.0	86.1	91.2	95.3	88.6
	MultiRocketHydra	49.9	96.0	94.1	94.1	98.1	94.1
	Catch22	18.5	97.0	88.9	70.6	97.2	78.7
	QUANT	43.4	100.0	91.7	97.1	97.2	94.3
FallAllD	Hydra	40.2	76.0	90.0	52.9	98.1	66.7
	Rocket	9.0	74.0	64.5	58.8	89.7	61.5
	MultiRocketHydra	43.3	82.0	85.2	67.7	96.3	75.4
	Catch22	13.5	86.0	73.5	73.5	91.6	73.5
	QUANT	41.5	91.0	85.2	67.7	96.3	75.4
FallAllD+	Hydra	58.9	91.0	96.6	82.4	99.1	88.9
	Rocket	13.6	86.0	83.9	76.5	95.3	80.0
	MultiRocketHydra	76.8	90.0	93.3	82.4	98.1	87.5
	Catch22	29.9	98.0	93.3	82.4	98.1	87.5
	QUANT	69.7	98.0	96.3	76.5	99.1	85.3
SisFall	Hydra	150.4	69.0	38.0	79.4	58.9	51.4
	Rocket	47.5	76.0	39.5	100.0	51.4	56.7
	MultiRocketHydra	195.0	69.0	37.3	82.4	56.1	51.4
	Catch22	93.4	88.0	31.4	97.1	32.7	47.5
	QUANT	81.7	84.0	46.2	88.2	67.3	60.6
SisFall+	Hydra	180.8	93.0	80.0	94.1	92.5	86.5
	Rocket	52.5	94.0	78.6	97.1	91.6	86.8
	MultiRocketHydra	218.7	94.0	88.6	91.2	96.3	89.9
	Catch22	109.7	95.0	77.1	79.4	92.5	78.3
	QUANT	54.9	98.0	83.3	88.2	94.4	85.7
All	Hydra	197.9	91.0	87.9	85.3	96.3	86.6
	Rocket	59.1	88.0	74.4	85.3	90.7	79.5
	MultiRocketHydra	233.9	88.0	90.0	79.4	97.2	84.4
	Catch22	141.6	94.0	84.9	82.4	95.3	83.6
	QUANT	122.8	95.0	82.4	82.4	94.4	82.4