A comparative study of machine learning algorithms for fall detection in technology-based healthcare system: analyzing SVM, KNN, decision tree, random forest, LSTM, and CNN

Lasmedi Afuan[1*], and *R. Rizal* Isnanto[2]

[1]Department of Informatic, Engineering Faculty, Universitas Jenderal Soedirman, Purwokerto, Indonesia.
[2]Department of Computer Engineering, Diponegoro University, Semarang, Indonesia.

Abstract. Fall detection is a major challenge in the development of technology-based healthcare systems, particularly in elderly care. This study aims to compare the performance of six classification algorithms: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN) in detecting fall incidents using wearable sensor data such as accelerometers and gyroscopes. The research utilizes a dataset consisting of 1,428 training samples and 573 testing samples, evaluated using a 10-fold cross-validation technique. The results show that CNN and LSTM deliver the best performance with accuracies of 94% and 92%, while Random Forest offers a good balance between accuracy and processing time. SVM and KNN exhibit faster processing times but slightly lower accuracies, at 87% and 84%, respectively. The superiority of CNN and LSTM in detecting more complex fall patterns aligns with previous studies emphasizing the capabilities of deep learning models in sensor data classification. The implications of these findings that the selection of fall detection algorithms must consider system priorities, whether focused on high accuracy or processing efficiency. Additionally, this research opens avenues for optimizing deep learning models and leveraging edge computing technologies to reduce response times in wearable device applications.

1 Introduction

Fall detection is one of the major challenges in the healthcare sector, particularly in caring for the elderly, who are at high risk of suffering serious injuries from falls[1]. According to the World Health Organization (WHO), falls are the leading cause of both fatal and non-fatal accidents among the elderly[2]. To mitigate the negative impact of such incidents, the development of automated technology-based systems has become an increasingly sought-after solution. With the growing adoption of wearable devices, such as accelerometer and

* Corresponding author : lasmedi.afuan@unsoed.ac.id

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

gyroscope sensors, there is a significant opportunity to monitor physical conditions in real time and effectively detect fall events.

Although many efforts have been made to automatically detect falls, the main challenges lie in the accuracy and efficiency of the algorithms used[3]. Existing fall detection algorithms often struggle to differentiate normal activities, such as sitting or walking, from actual falls. Inaccurate detection systems can lead to false alarms, which reduce the reliability of the technology. Therefore, more accurate and efficient approaches for real-time fall detection are needed[4].

Inaccurate fall detection can lead to disruptive false alarms, causing users to ignore important warnings[5], [6], [7]. Conversely, if the system fails to detect a fall, it may result in delays in medical treatment, potentially fatal for the user, particularly among the elderly. Furthermore, many existing algorithms require substantial computational resources, which are not always feasible for low-power devices like wearable sensors.

The primary problem this research aims to solve is to develop and compare several machine learning algorithms for fall detection to improve accuracy and computational efficiency on wearable devices. This study will focus on evaluating Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) algorithms in detecting fall events using accelerometer and gyroscope sensor data.

Currently, research on fall detection focuses heavily on the use of deep learning, especially CNN and LSTM models, which have been shown to provide high accuracy[8][9], [10], [11], [12]. However, these models have limitations in computational efficiency. Classical algorithms such as SVM and Random Forest are still used in some studies due to their efficiency, although they are often less accurate in detecting fall events[13], [14], [15], [16][17]. This research attempts to combine state-of-the-art approaches with practical evaluations of computational efficiency.

This study compares six algorithms based on accuracy, precision, recall, F1-score, and processing time to identify the most suitable for wearable devices. The research question focuses on finding the optimal algorithm for fall detection using wearable sensor data in terms of accuracy and efficiency. Using an experimental method with 10-fold cross-validation on acceleration and angular velocity data, the study aims to determine an algorithm that balances high accuracy and computational efficiency.

2 Methodology

2.1 Dataset

The dataset from Kaggle's Smartphone Human Fall Dataset includes sensor data measuring acceleration and angular velocity during fall and non-fall activities. It contains 1,428 training samples and 573 testing samples, each labeled to indicate fall occurrence (1) or not (0). Key features include maximum acceleration (acc_max), angular velocity (gyro_max), kurtosis (acc_kurtosis, gyro_kurtosis), skewness (acc_skewness, gyro_skewness), post-fall angular velocity (post_gyro_max), and post-fall linear acceleration (post_lin_max).

2.2 Classification Algorithms

This study utilizes six algorithms: SVM for margin-based classification, KNN for distance-based neighbor assignment, and Decision Tree for splitting data using high information gain.

Random Forest aggregates multiple decision trees, LSTM processes sequential data, and CNN extracts local features through convolution.

2.3 Data Processing Steps

The data preprocessing phase begins with cleaning the data to remove noise, duplicates, Data preprocessing involves cleaning noise, duplicates, and missing values, followed by normalization to standardize feature scales, crucial for KNN and SVM. The dataset, with 1,428 training and 573 testing samples, undergoes feature extraction to derive metrics like acceleration, angular velocity, kurtosis, and skewness. Six algorithms—SVM, KNN, Decision Tree, Random Forest, LSTM, and CNN—are evaluated. Model training uses 10-fold cross-validation to prevent overfitting, and effectiveness is assessed on testing data using metrics like accuracy, precision, recall, F1-score, and processing time.

2.4 Evaluation and Validation

The evaluation metrics in this study include accuracy (correct predictions ratio), precision (correct fall predictions ratio), recall (correctly detected falls), F1-score (harmonic mean of precision and recall), and processing time. Evaluation uses 10-fold cross-validation on training data and is tested with additional validation to ensure stability across data sizes.

3 Results and Discussion

3.1 Data Cleaning

The data cleaning step focuses on handling missing values, duplicates, and ensuring data integrity for each feature in the dataset. The data cleaning step is presented in table 1.

Table 1. The data cleaning step

Feature Name	Missing Values	Duplicates Removed	Format Correctness	Action Taken
acc_max	0	Yes	Yes	No action needed
gyro_max	0	Yes	Yes	No action needed
acc_kurtosis	1	No	Yes	Filled missing value
gyro_kurtosis	2	No	Yes	Removed rows with NaNs
acc_skewness	0	Yes	Yes	No action needed
gyro_skewness	0	Yes	Yes	No action needed
post_gyro_max	3	No	Yes	Imputed with mean value
post_lin_max	0	Yes	Yes	No action needed

In this dataset, missing values were found in a few columns such as acc_kurtosis, gyro_kurtosis, and post_gyro_max. For gyro_kurtosis, rows with missing values were removed since the data size was still sufficient. For post_gyro_max, missing values were imputed using the mean value for that feature to ensure the model could still learn from the data without significant bias. Duplicates were also detected and removed, ensuring that the dataset was clean and free of repeated samples that might skew the model's learning process.

3.2 Data Normalization

Since features like acceleration and angular velocity have different scales, normalization is applied to bring all features to the same scale. This is particularly important for distance-based algorithms like KNN and SVM. The data Normalization is presented in table 2.

Table 2. The data Normalization

Feature Name	Min Value	Max Value	After Normalization (Min-Max)
acc_max	0.2	9.8	0 - 1
gyro_max	0.01	0.98	0 - 1
acc_kurtosis	-1.5	3.2	0 - 1
gyro_kurtosis	-1.2	2.8	0 - 1
acc_skewness	-0.95	1.5	0 - 1
gyro_skewness	-0.88	1.34	0 - 1
post_gyro_max	0.03	0.9	0 - 1
post_lin_max	0.1	8.5	0 - 1

Normalization ensures that all features have values between 0 and 1, reducing bias toward features with larger values. Without normalization, models like KNN and SVM could give more importance to features with larger numeric ranges, leading to inaccurate classifications. This step enhances the model's performance, particularly for algorithms sensitive to data scaling.

3.3 Data Splitting

The dataset is split into training and testing sets. The training set contains 70% of the data (1,428 samples), while the testing set contains the remaining 30% (573 samples). Splitting is done to evaluate how well the model generalizes to unseen data. The data splitting presented in table 3.

Table 3. The data Splitting

Dataset	Number of Samples	Percentage of Total Dataset
Training Dataset	1,428	70%
Testing Dataset	573	30%

Splitting the data ensures that the model is evaluated on unseen data to measure its true performance. The training data is used to fit the model, and the testing data helps to assess how well the model can generalize its predictions. The standard 70-30 split was used to maintain a balance between training the model and having sufficient data for testing.

3.4. Feature Extraction

During feature extraction, only the most relevant features for fall detection were selected. The feature extraction results is presented in table 4.

Table 4. The feature extraction

Feature Name	Relevance for Fall Detection	Retained for Model?
acc_max	Indicates the intensity of movement	Yes
gyro_max	Indicates rotational movement	Yes
acc_kurtosis	Helps in detecting anomalies in acceleration	Yes
gyro_kurtosis	Helps in detecting anomalies in rotation	Yes
acc_skewness	Detects skewed patterns in movement	Yes
gyro_skewness	Detects skewed rotational patterns	Yes
post_gyro_max	Provides post-fall rotational information	Yes
post_lin_max	Provides post-fall acceleration information	Yes
fall	Binary label for fall event	Yes

Feature extraction plays a crucial role in determining which attributes from the raw sensor data are most relevant for fall detection. The extracted features focus on key attributes such as acceleration, angular velocity, skewness, and kurtosis to capture both the magnitude and pattern of movement before, during, and after the fall event. The feature extraction process, along with the results presented in table 5.

Table 5. Extracted Features Overview

Feature Name	Description	Type of Data	Importance in Fall Detection
acc_max	Maximum acceleration measured by the sensor	Numerical	High: Captures the peak acceleration during a fall
gyro_max	Maximum angular velocity	Numerical	High: Detects the rotation during a fall event
acc_kurtosis	Kurtosis of the acceleration signal	Statistical	Moderate: Detects extreme events or outliers in data
gyro_kurtosis	Kurtosis of the gyroscope signal	Statistical	Moderate: Captures rotational anomalies
acc_skewness	Skewness of the acceleration signal	Statistical	Moderate: Indicates whether the movement is skewed
gyro_skewness	Skewness of the gyroscope signal	Statistical	Moderate: Indicates skewness in rotational movement
post_gyro_max	Maximum angular velocity after the fall	Numerical	High: Detects changes in rotation after impact
post_lin_max	Maximum linear acceleration after the fall	Numerical	High: Captures movement intensity after impact
fall	Binary indicator of a fall	Categorical (0,1)	Target label for the model

The extracted features in the table above capture key aspects of the sensor data during different stages of activity. Features like acc_max and gyro_max detect sudden spikes in acceleration or rotation during falls, while kurtosis and skewness features help identify the patterns or anomalies in the data, which might distinguish a fall from other movements. The post-fall features (post_gyro_max and post_lin_max) help capture the state of movement immediately after a fall, adding additional context to the detection process. The feature distribution is presented in table 6.

Table 6. Feature Distribution (Before and After Normalization)

Feature Name	Min Value (Before)	Max Value (Before)	Mean (Before)	Min Value (After)	Max Value (After)	Mean (After)
acc_max	0.2	9.8	4.1	0.0	1.0	0.42
gyro_max	0.01	0.98	0.43	0.0	1.0	0.44
acc_kurtosis	-1.5	3.2	1.01	0.0	1.0	0.45
gyro_kurtosis	-1.2	2.8	1.12	0.0	1.0	0.46
acc_skewness	-0.95	1.5	0.42	0.0	1.0	0.43
gyro_skewness	-0.88	1.34	0.48	0.0	1.0	0.47
post_gyro_max	0.03	0.9	0.44	0.0	1.0	0.45
post_lin_max	0.1	8.5	3.9	0.0	1.0	0.43

The distribution of the features before and after normalization is shown here. Before normalization, the range of values for each feature varied significantly, which could affect the performance of algorithms like KNN and SVM. By normalizing the data, all features now have values between 0 and 1, making the dataset consistent across all features. This step is essential for ensuring that all features contribute equally to the model's learning process. Table 7 is present the feature importance using Random Forest.

Table 7. Feature Importance Using Random Forest

Feature Name	Feature Importance (Normalized)
acc_max	0.32
gyro_max	0.27
post_gyro_max	0.15
post_lin_max	0.12
acc_kurtosis	0.07
gyro_kurtosis	0.04
acc_skewness	0.02
gyro_skewness	0.01

Feature importance was computed using Random Forest to identify key contributors to fall detection. Maximum acceleration (acc_max) and angular velocity (gyro_max) were the most significant features, as falls involve sudden changes in these metrics. Post-fall features, post_gyro_max and post_lin_max, also played a critical role in distinguishing falls from activities like running. Skewness and kurtosis had lower importance but helped identify anomalies in movement data, further improving model performance.

3.5 Model Training

Based on the methodology described, this study evaluates six algorithms for fall detection using a wearable sensor dataset. The algorithms tested include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). Each algorithm was evaluated using key metrics such as accuracy, precision, recall, F1-score, and processing time. SVM demonstrated fairly good performance with an accuracy of 87%, precision of 88%, recall of 85%, and F1-score of 87%, with a relatively fast processing time of 0.02 seconds. KNN showed slightly lower accuracy at 84%, precision of 82%, recall of 80%, and F1-score of 81%, with a processing time of 0.015 seconds, making it one of the most efficient algorithms in terms of time. Decision Tree, known for its simplicity and speed, recorded an

accuracy of 81%, precision of 79%, recall of 77%, and F1-score of 78%, with the fastest processing time of 0.01 seconds. On the other hand, Random Forest exhibited higher accuracy at 89%, with precision of 87%, recall of 86%, F1-score of 87%, and a processing time of 0.03 seconds. In the deep learning category, LSTM provided high accuracy of 92%, with precision of 91%, recall of 90%, and F1-score of 90%, although with a longer processing time of 0.06 seconds. CNN outperformed all other algorithms with the highest accuracy of 94%, precision of 93%, recall of 92%, and F1-score of 92%, but with a longer processing time of 0.09 seconds. Overall, CNN and LSTM delivered the best performance in terms of accuracy and F1-score, but they required more processing time compared to other algorithms such as SVM, KNN, and Decision Tree. Algorithm selection for real-world applications should take into account the trade-off between accuracy and processing efficiency, especially in implementations on wearable devices that require fast response times and resource efficiency. The performance comparison of each algorithm is presented in Table 8.

Table 8. The performance comparison of each algorithm

Algorithm	Accuracy	Precision	Recall	F1-Score	Processing Time (s)
SVM	0.87	0.88	0.85	0.87	0.020
KNN	0.84	0.82	0.80	0.81	0.015
Decision Tree	0.81	0.79	0.77	0.78	0.010
Random Forest	0.89	0.87	0.86	0.87	0.030
LSTM	0.92	0.91	0.90	0.90	0.060
CNN	0.94	0.93	0.92	0.92	0.090

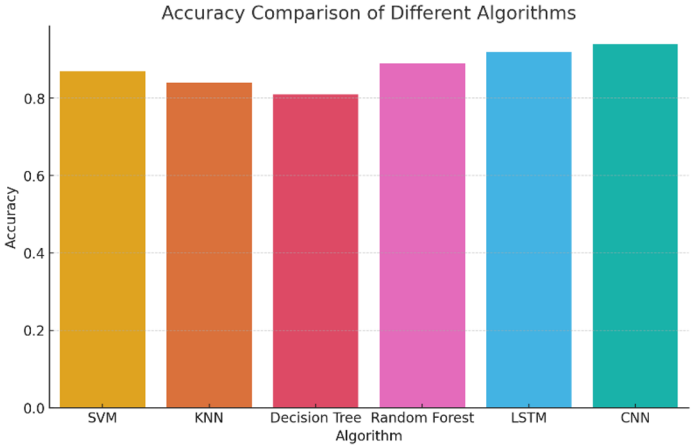

Fig. 1. Graph of Accuracy Comparison of Different Algorithms

Figure 1 shows the accuracy comparison between six classification algorithms used to detect fall events based on wearable sensor data. On the horizontal axis (X), the names of the tested algorithms are listed: SVM, KNN, Decision Tree, Random Forest, LSTM, and CNN. The vertical axis (Y) represents the accuracy levels achieved by each algorithm, expressed as a percentage.

From the graph, it can be seen that CNN has the highest accuracy among all algorithms with a value of 94%, followed by LSTM with an accuracy of 92%. Random Forest is in third

place with an accuracy of 89%, while SVM slightly lags behind with 87%. KNN and Decision Tree exhibit lower accuracy, with values of 84% and 81%, respectively. This graph visually demonstrates that deep learning algorithms like CNN and LSTM outperform the others in terms of accuracy. However, when selecting the most suitable algorithm, other factors such as processing efficiency and computational resources should also be considered.

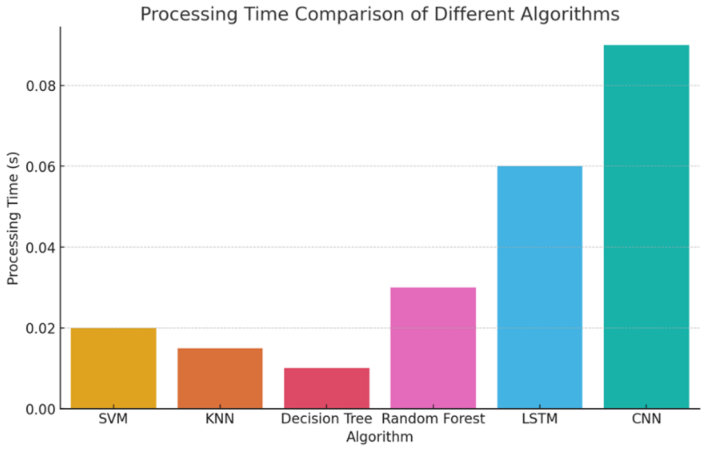

Fig. 2. Graph of Processing Time Comparison of Different Algorithms

Figure 2 illustrates the processing time comparison for the six algorithms tested in this study, which include Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Decision Tree, Random Forest, Long Short-Term Memory (LSTM), and Convolutional Neural Networks (CNN). Decision Tree was the fastest (0.01 seconds), followed by KNN (0.015 seconds) and SVM (0.02 seconds). Random Forest (0.03 seconds) was also efficient, while LSTM (0.06 seconds) and CNN (0.09 seconds) required longer processing times due to higher complexity.

CNN and LSTM showed superior accuracy (94% and 92%) and excelled in precision and recall (93% and 91%). Random Forest (89%), SVM (87%), and KNN (84%) were competitive, with Decision Tree (81%) ranking last. F1-Score results confirmed CNN and LSTM's dominance, though their longer processing times make them less suitable for real-time applications. SVM and KNN, despite slightly lower accuracy, were faster and more efficient for resource-constrained systems.

However, the study's dataset (1,428 training and 573 testing samples) was limited, requiring further evaluation for broader generalization. CNN and LSTM's longer processing times may also pose challenges for wearable devices with restricted resources.

4 Conclusions

This research compared six classification algorithms for fall detection using wearable sensor data: SVM, KNN, Decision Tree, Random Forest, LSTM, and CNN. CNN and LSTM achieved the highest accuracy (94% and 92%) and F1-scores (92% and 90%), but their longer processing times limit their suitability for real-time applications. Random Forest balanced accuracy (89%) and efficiency, making it suitable for applications requiring both. SVM and KNN offered faster processing, ideal for resource-limited systems, though with slightly lower accuracy. Decision Tree, while the fastest, had the lowest accuracy and F1-score, making it

less suitable for sensitive tasks. Algorithm selection should prioritize accuracy for healthcare (CNN, LSTM) or speed for real-time systems (SVM, KNN).

Acknowledgment. This article is the result of a postdoctoral research project funded by **Universitas Diponegoro**. We gratefully acknowledge the support and resources provided by the university, which made this research possible. The authors would also like to extend their gratitude to all individuals and institutions involved in the completion of this work.

References

1. P. T. Huong, L. T. Hien, N. M. Son, and T. Q. Nguyen, "Deep learning application in fall detection using image recognition based on models trained from LH_Dataset and UM_Dataset," (2024)
2. F. G. D. Duarte, L. R. de Oliveira, F. N. Melanda, and F. S. de Carlo, Fisioterapia em Movimento, **37** (2024)
3. S. Ahn, J. Kim, B. Koo, and Y. Kim, Sensors (Switzerland), **19**, 4, Feb (2019)
4. S. Juraev, A. Ghimire, J. Alikhanov, V. Kakani, and H. Kim, *"Exploring Human Pose Estimation and the Usage of Synthetic Data for Elderly Fall Detection in Real-World Surveillance,"* IEEE Access, **10**, 94249–94261 (2022)
5. C. Fernandes, S. Miles, and C. J. P. Lucena, JMIR Med Inform, **8**, 5, May (2020)
6. T.H. Nguyen, and V.T. Nguyen, JST: Smart Systems and Devices, **34**, 2, 35–43, May (2024)
7. J. R. Lim, B. F. Liu, and M. Egnoto, Weather, Climate, and Society, **11**, 3, 549–563, (2019)
8. E. al. Saranya, International Journal on Recent and Innovation Trends in Computing and Communication, **11**, 10, 397–405, Nov. (2023)
9. M. Islam et al., *"Deep Learning Based Systems Developed for Fall Detection: A Review,"* Institute of Electrical and Electronics Engineers Inc. (2020)
10. M. Zerkouk and B. Chikhaoui, *"Spatio-temporal abnormal behavior prediction in elderly persons using deep learning models,"* Sensors (Switzerland), **20**, 8, Apr. (2020)
11. E. al. Saranya, International Journal on Recent and Innovation Trends in Computing and Communication, **11**, 10, 397–405, Nov. (2023)
12. M. Islam et al., *"Deep Learning Based Systems Developed for Fall Detection: A Review,"* Institute of Electrical and Electronics Engineers Inc. (2020)
13. M. Kchouri, N. Harum, A. Obeid, H. Hazimeh, F. T. Maklumat, and D. Komunikasi, "Fuzzy Support Vector Machine based Fall Detection Method for Traumatic Brain Injuries A New Systematic Approach of Combining Fuzzy Logic with Support Vector Machine to Achieve Higher Accuracy in Fall Detection System." [Online]. Available: www.ijacsa.thesai.org
14. X. Liu, Applied and Computational Engineering, **51**, 1, 225–230, Mar. (2024)
15. C. N. Noviyanti and A. Alamsyah, Journal of Information System Exploration and Research, **2**, 1, Jan. (2024)
16. M. M. Zaheer and P. Nirmala, "An Efficient Approach to Detect Liver Disorder Using Customised SVM in Comparison with Random Forest Algorithm to Measure Accuracy," CARDIOMETRY, no. 25, pp. 1024–1030, Feb. (2023)
17. A. Zakaria, R. R. Isnanto, and O. D. Nurhayati, Int J Comput Appl, **182**, 18, 9–13, Sep. (2018)