Received: 15 August 2024 | Revised: 12 October 2024 | Accepted: 7 November 2024 | Published online: 13 November 2024

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2024, Vol. 00(00) 1–8
DOI: 10.47852/bonviewJDSIS42024100

BON VIEW PUBLISHING

Vision-Based Fall Detector for Elderly Based on Sliding Window Approach and Feature Engineering

Boris Assanovich[1],* **and Katsiaryna Kosarava**[2]

[1]*Information Systems and Technologies Department, Yanka Kupala State University of Grodno, Belarus*

[2]*Faculty of Mathematics and Natural Sciences, Cardinal Stefan Wyszynski University in Warsaws, Poland*

Abstract: A new vision-based fall detector is proposed that uses the tsfresh tool to generate features from the bounding box motion parameters of an object and performs classification in a sliding window mode. The efficiency of the generated features is demonstrated compared to the primary ones. Using the auto-sklearn library and a generalized dataset compiled from the UR Fall Detection and CAUCAFall datasets, the best human fall detection model is found. This model based on a gradient boosting classifier achieved 96% accuracy, which is not inferior to well-known detection algorithms, but uses only two primary motion parameters to generate secondary features. A principal component analysis-based class separability study showed that for secondary features, 99% of the variance is captured by the first 4 principal components, while for primary features, the first 10 principal components contain only 80% of the data variance. Furthermore, the processing time for generating secondary features and making predictions was found to be relatively short, taking only a few seconds per sequence, highlighting the practical applicability of the proposed approach in real-time fall monitoring systems.

Keywords: machine learning, sliding window, human fall detection, classification model, tsfresh, auto-sklearn

1. Introduction

Monitoring the activity of the elderly has gained significant attention in recent years. One significant concern for individuals over 70 is losing balance and falling, which is particularly perilous for those living alone as they may not receive timely assistance. This issue has sparked a surge in studies focused on remote fall detection enabling fast and qualified help for the elderly.

Falls are indeed a significant public health concern, particularly among the elderly population. According to the World Health Organization, falls are the second-leading cause of injury-related death worldwide among people aged 65 and older [1].

Fall detection systems can use both invasive, based on wearable sensor devices, and non-invasive methods, based on computer vision (CV) or so-called vision-based technology. It is obvious that in many practical situations, remote CV control is preferable. Additionally, vision-based fall detection can provide feedback and training to individuals to improve their balance and reduce the risk of falls. This technology has immense potential to revolutionize fall control and improve the lives of vulnerable individuals.

This paper proposes a new vision-based fall detector using tsfresh tool to generate secondary features from motion parameters and perform classification in a sliding window mode. This will allow detecting a person's fall based on a limited sequence of frames, which will reduce the time of detection and response to the fact of a fall. The implementation of the detector is based on machine learning methods using freely available datasets and setting up a computational experiment. The choice of the fall detector model is based on the use of the AutoML tool, which is used in the comparison of the proposed method for the primary and secondary features of the motion parameters during a fall.

*Corresponding author:** Boris Assanovich, Information Systems and Technologies Department, Yanka Kupala State University of Grodno, Belarus. Email: bas@grsu.by

2. Literature Review

A comprehensive revision of many published papers regarding this area has been made by Gutiérrez et al. [2], where the main characteristics of more than 80 similar systems were described including their classification and performance. The review of Jesús Gutiérrez et al. analyzed almost 500 papers, most of which were based on machine learning and the use of neural networks. Recently, Alam et al. [3] presented a survey discussing deep learning-based fall detection methods in which they evaluate the performance of fall detection systems, their performance metrics, and the related datasets. Analysis has shown that many modern systems have a sensitivity of over 90% for fall detection. In general, current system *State of Art* that perform fall detection includes three main procedures: (1) video signal preprocessing including image transformation and its segmentation; (2) extracting features (descriptors) by highlighting local or global features of the image; (3) actual fall detection based on the obtained features and classification algorithms [3]. Research on fall detection has explored various approaches including CV and image processing. Redmon et al. [4] utilized a convolutional neural network (CNN) to analyze images from videos, leveraging optical flow to detect motion between frames. Several cameras were used to detect changes from frame to frame. While this method achieved high accuracy, it requires special cameras or multiple devices. Lezzar et al. [5] proposed a simpler algorithm for determining falls by comparing them to normal activities of daily living (ADL). They used a 2D camera and occlusion recognition achieving 93.94% accuracy with the SVM classifier by increasing the number of supervised states.

Researchers from the University of Burgundy prepared so-called ImViA Fall detection dataset, which contained 191 different activities including ADLs and 143 falls. Charfi et al. [6] performed a fairly detailed analysis to identify low-level features and evaluate fall detection using the SVM classifier, which was later compared with Adaboost-based classification in another work from the same university [7]. In this work, researchers experimentally proposed to construct special spatiotemporal human fall descriptors. For this purpose, they used to find the best combination of several features (horizontal and vertical projection histograms, parameters of the rectangle that bounds the object, a number of suitable parameters of the fitting ellipse, etc.,) and their transformations. The proposed fall detection protocols used the combinations of selected features and the same dataset, which was later called L2i, allowed achieving a resulting classification error of about 1%. However, the approaches to feature selection in the publications discussed above are quite resource-intensive and require a manual selection procedure.

Later, a number of other methods appeared based on searching for key points (KP) of the human body, analyzing their relationships, and obtaining a number of associated features to perform classification and detect a person's fall. In Beddiar et al. [8], authors showed that the combination of an SVM classifier with LSTM neural networks using such features as distances and angles between different points of the human body can improve the overall performance of a fall detection system. At the same time, for training and testing, both the L2i dataset and the UR Fall Detection Dataset were used, which contained data received from the accelerometer and Microsoft Kinect cameras [9].

Recently, in Zhang et al. [10], the authors constructed a new inverted pendulum model for the human body and the corresponding spatiotemporal pose evolution map. The authors proposed several fall detection algorithms based on Lagrangian mechanics of human body motion and achieved excellent detection accuracy of 0.958–0.979 for their proposed algorithms. Unfortunately, the authors did not disclose the implementation details except that they used a multi-stage CNN consisting of two branches to extract the KP of the human body or skeleton showing spatial relationships between body parts used for pose estimation systems.

Today, the application of new computer technologies has made it possible to improve the algorithms for human KP data processing, which is confirmed by the implementation of a series of YOLO models including specially trained structures for detecting falls [11]. However, as we know, these models are quite complex and resource-intensive.

To reduce the time of the fall detection process, the sliding window algorithm can be effective which is widely used in machine learning together with feature engineering. Recent publications have shown their use not just in detecting the state of a person by wearable sensors [12] but also in diagnosing the abnormal state of machine elements [13]. However, according to our information, this technique is not sufficiently represented by researchers in vision-based monitoring of the activity of the elderly. In this paper, we make an attempt to fill this gap and consider its application in detecting falls based on the groundwork we have previously made.

In our previous paper [14], we have used both the L2i dataset and the UR Fall Detection Dataset to compare the performance of a LSTM neural network detector working with the basic CV features: Bounding box (BB) height to width ratio or so-called aspect ratio (AR) and speed or velocity of BB movement with several other classification methods. The other classifiers (support vector machine, decision tree, and random forest) were trained using secondary features generated by the tsfresh library for the time series analysis [15]. The results allowed achieving accuracy up to 1.0. However, the experiments were carried out on the basis of a rather small set of 70 selected video sequences with a length of 300 frames obtained from L2i dataset performing the analysis within 10 s, which may be unacceptable for practical applications.

On the other hand, with the advent of new machine learning tools such as AutoML, it has also been applied to fall detection. Based on the analysis of data from wearable sensors, it was stated in Kausar et al. [16] that it is possible to detect falls in elderly individuals with an accuracy of about 99%, while spending about 6 s on processing at best. However, in publications covering the results of code implementations of these methods, the detection accuracy was 95–97% [3]. In this regard, we decided to verify the use of AutoML for vision-based technology.

In this study, we improved a previously proposed fall detector that uses tsfresh technology to obtain secondary features from motion parameters and performs classification in a sliding window mode. In addition, we compared the effectiveness of using primary and secondary parameters for fall detection.

Our contribution to improving the detector is threefold. Firstly, we proposed a sliding window algorithm to increase the amount of both train and test data and speed up fall detection within a few seconds. Secondly, we prepared the CAUCAFall dataset for training and testing the proposed algorithm, on the basis of which we generalized a dataset that combines sequences also from the UR Fall dataset. And thirdly, an approach for automatically selecting the most effective classification algorithm with its optimal hyperparameters based on the auto-sklearn library for vision-based fall detection has been exploited.

The article is organized as follows. The following section describes the basic physics of human fall and the structure of the system. Next, the description of the datasets, the selection of the sliding window parameters, and the optimization of the fall detector model are presented along with the experimental results, their discussion, and their performance comparison. The last section concludes the study.

3. Problem Overview and System Design

3.1. Theoretical framework

The location of the human body's center of mass (COM) is critical to maintaining its equilibrium both while standing and while moving. The change in a person's position during a fall is a dynamic process that is determined by the transition from a stable to an unstable state. However, the moment of a fall is uncertain: the human body may be unstable due to the intervention of external forces, or the balance may be disturbed due to the body's own causes. Fall behavior is time-varying and continuous: after a fall, the human body's posture changes dynamically, and each part of the body can change its position in its own way. The connections of individual body parts correspond to a skeletal structure consisting of a set of interconnected KP. In this regard, recently there have been more and more studies examining methods for fall detection and fall prevention based on the analysis of the human body KPs [2]. However, the complexity of such algorithms increases greatly compared to analyzing the movement of COM only. Studies have shown that the dynamic stability of a person while walking is determined by a sacral marker (a point located at the base of the spine) which can be replaced with fairly high accuracy by COM [17]. According to the representation of the human body, consisting of individual elements around a COM, the canonical shape of an object is determined by the external forces acting on it $\frac{d^2R}{dt^2} = F^{ext}$ [18]. However, in practice, the determination of external forces $\boldsymbol{F^{ext}}$ to find the radius-vector $\boldsymbol{R}$ is virtually impossible and the calculation of indirect parameters, such as speed (velocity) and acceleration of COM, may be applied. At the same time, based on CV technology, the location of a person can be detected assuming that his body is inside the BB. Next, to simplify the development of a fall control practical model, in this study we analyze the fall of a person whose geometric center coincides with COM. For the most simple and symmetrical figures, the geometric center coincides with COM [19] and is called their centroid. Calculating the centroid speed or the velocity of BB instead of COM motion characteristics defines the following broad approach for a fall detection as described below.

3.2. Research design

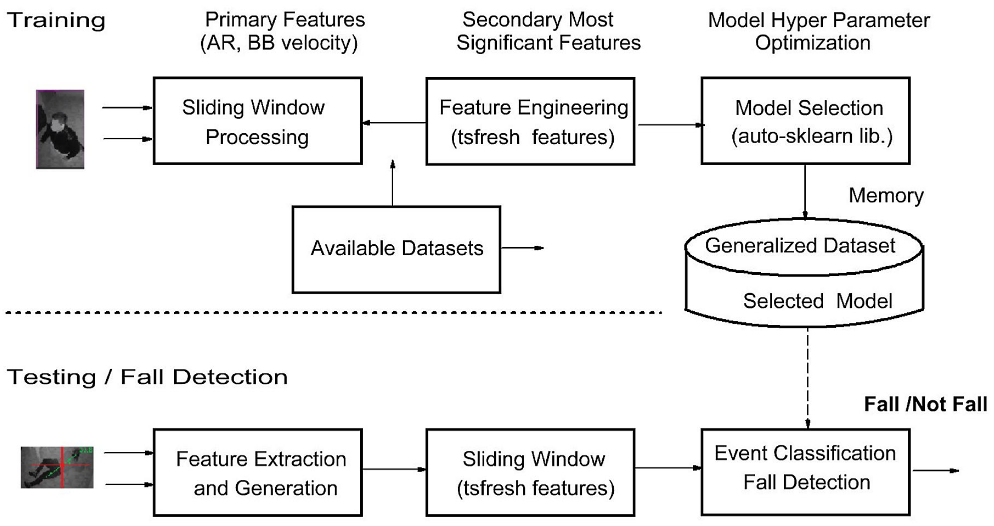
With the use of vision-based algorithm, the coordinates of the BB within a frame as the fall occurs can be consistently updated. By considering the uniform motion of an object from frame to frame, we can easily calculate the speed of the BB centroid based on the changes in position over time. This can be done by calculating the displacement between the consecutive frames and then dividing by the time interval between frames. It is important to note that while this calculation provides insights into the BB movement, it may not reflect the actual position of the human body during a fall. The AR representing the relationship between the BB width and height is another important parameter for the human body tracking. The AR of a BB is a valuable feature in fall detection, as it can indicate the orientation of person and help distinguish between falls and non-falls. For example, a sudden change in AR can indicate a person falling, as the body tends to stretch out in the direction of the fall.

Thus, the AR and centroid velocity (speed) are two valuable features in fall detection. The AR is sensitive to changes in orientation and COM velocity depends on the change in the center of gravity. Combining these 2 features can provide a more comprehensive understanding of the fall event. This is confirmed by a number of previous studies. Gutiérrez et al. [2] and Lezzar et al. [5] studied the human body COM trajectory that moves along an arc during each step and proved that it can be an index of both balance and the neural maturation of walking.

Consider for a while the free fall of a standing person with a loss of balance. Then, knowing the distance from COM to ground (0.45 m on average) and neglecting air resistance, it is easy to see that the fall time can be approximately 0.3 s. Further, if the frame rate (fps) of video camera is 30 fps, then for this time we would get only about 10 frames. This simplification reinforces the need to quickly respond to a fall with an alarm signal.

Therefore, this study proposes a fall detection approach based on the use of a sliding window algorithm. In this case, the window is a time interval containing a set of values that are used to form a training or test example when performing classification. For this purpose, the most informative parameters are selected from the time series corresponding to the video surveillance data for the elderly. Next, video sequences are splitted using a sliding window algorithm that selects the data subsequences allowing detecting a fall. For every obtained subsequence, secondary features are generated with the use of tsfresh library. It should be noted that the classification model is selected using the auto-sklearn tool based on the analysis of an ensemble of supervised machine learning models. The block diagram of the proposed detector and the corresponding data processing steps are shown in Figure 1. The principle of data preparation, processing, and model selection is shown in Figure 1 in the form of separate steps and includes two stages: detector training, and detector testing. First, the video surveillance data (testing stage) are pre-processed, then primary features are extracted and buffered for the sliding window algorithm, secondary features are generated, and fall detection is performed based on the selected model. The model is built at the training stage using the auto-sklearn tool. The details of choosing the best parameters for the sliding window algorithm and the classifier type are presented in the next section.

Figure 1
Data processing by fall detector


Training
Primary Features (AR, BB velocity)
Secondary Most Significant Features
Model Hyper Parameter Optimization
Sliding Window Processing
Feature Engineering (tsfresh features)
Model Selection (auto-sklearn lib.)
Memory
Available Datasets
Generalized Dataset
Selected Model
Testing / Fall Detection
Fall /Not Fall
Feature Extraction and Generation
Sliding Window (tsfresh features)
Event Classification Fall Detection

4. Research Methodology

4.1. Dataset contents and its cleaning

The UR Fall Detection Dataset [9], commonly known as the UR Fall Dataset, is extensively utilized in machine learning and is particularly valuable for those focusing on human activity recognition, especially in healthcare and elderly care scenarios. This dataset comprises 70 sequences (30 falls and 40 ADL). Fall incidents are captured using two Microsoft Kinect cameras along with corresponding accelerometer data, while ADL events are recorded using only one device (camera 0) and an accelerometer. The dataset includes files with features extracted from depth maps, stored in CSV format. Each row represents a data sample corresponding to a single depth image and includes features such as the sequence name (e.g., 'fall-01', 'adl-01'), frame number, label ('-1' indicates the person is not lying down, '1' indicates the person is lying on the ground, '0' indicates a temporary pose during a fall), HeightWidthRatio (AR) – the ratio of BB height to width, MajorMinorRatio – the ratio of the major to minor axis calculated from the segmented person's BLOB, among others. More about dataset can be found in Kwolek and Kepski [9] as well as in Yhdego et al. [12]. The features extracted from the videos are crucial for distinguishing between falls and other activities.

CAUCAFall [20] is a database designed for recognizing human falls in uncontrolled home environments featuring diverse conditions such as occlusions, lighting changes, varied clothing, background movement, different fall angles, and distances from the camera. It includes segmentation labels for each image, facilitating the implementation of human fall recognition methods using YOLO detectors. The dataset includes simulations of five different types of falls and five ADLs. Falls are labeled as "fall", while ADLs are labeled as "no-fall". The data are structured into 10 main directories, each representing a different subject. Within each directory, there are 10 folders corresponding to the various activities performed by the participants. Each folder contains a video of the respective action in.avi format, images of the actions in.png format, and frame segmentation labels in.txt format.

For the UR Fall dataset, we combined the recordings into a single data frame and added a variable-indicator "*Fall*" which takes the values **true/false** for records with falls and without them respectively. Each frame in the dataset is labeled $-1; 0; 1$ defining that person is "not lying", "lying", and "falling/lying down" correspondently. The data cleaning was also performed. Firstly, for ADL videos the sequences were trimmed where the label 0 appeared after the label 1, indicating that person got up after lying down since our system must immediately output signal about fall not waiting for the end of record. Secondly, the sequences with missing frames were also found as soon as we apply frame numbering for calculation of BB velocity. In the experiments conducted, this parameter was called *"FrameSpeed"* and was calculated in one millisecond using the following formula:

$$FrameSpeed = \frac{fps}{1000df}\sqrt{(dx^2 + dy^2)},$$

where *dx, dy* define corresponding coordinate change of BB center;

df – video frame number increment while their numbering (records had missing frames);

fps – frame rate, which is 30 fps for the UR Fall dataset.

For CAUCAFall dataset, the given information from files of all folders has been combined into one dataset, and a variable "*fall*" was assigned **true** for video records with falls. Then, the BB coordinates of each frame were converted from YOLO format to absolute values. The features *HeightWidthRatio* and *FrameSpeed* were calculated for *fps* = 23 in this dataset. For video recordings of CAUCAFall dataset for the sequences corresponding to the fall process ("fall" dataset folders), we selected the beginning of sliding window starting from 5th frame, since the visual analysis of videos showed that fall process has started from the very beginning of recordings. The sliding window transformation procedure was then performed similarly to that used for the UR Fall Dataset.

In this paper, we conducted experiments to train the above model using primary video features (*HeghtWidthRatio* and *FrameSpeed*) and secondary features generated by the tsfresh library [21], which is used for systematic feature engineering from time series and other sequential data.

Next, to prepare data to apply the fall detector parameters setup (Figure 1), it was necessary to convert all records into output sequences of a certain length. To do it, the sliding window algorithm was used with the following parameters:

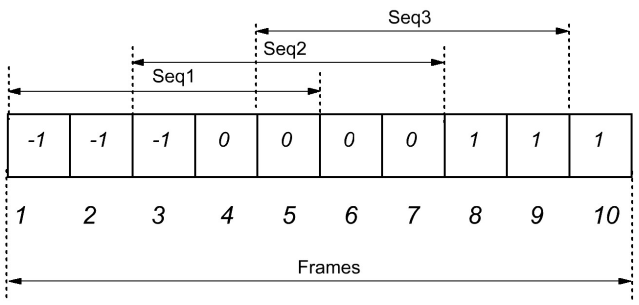
Window – width of the sliding window (equal to the length of the output sequence);

Stride is the step by which the frames were converted into sequences;

Fall_Frames – number of falling frames (elements in the output sequence marked as 0).

The operation of the method for *Window*=5, *Stride*=2, *Fall_Frames*=2 is depicted in Figure 2 below. For this example, our transformation algorithm returns only those sequences with length 5 in which at least 2 elements have a label of 0 (fall).

Figure 2
Sliding window



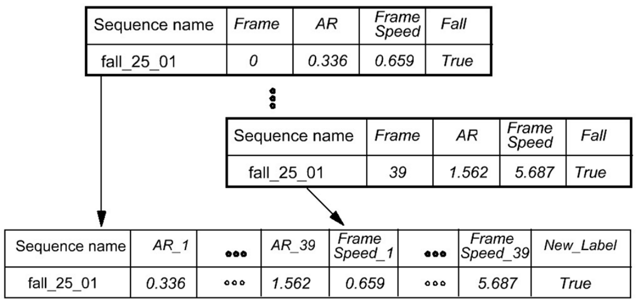
4.2. Sliding window parameters configuration

To select the values of parameters *Window, Stride, Fall_Frames,* we trained a random forest model using cross-validation. The parameter values were selected from the following ranges: *Window* = (30, 40, 50), *Stride* = (3,5,7,10), *Fall_Frames* = (10,15,20).

We used a brute-force approach to find the optimal combination of parameters. For each combination, we applied a sliding window over the data based on the values of the *Window, Stride,* and *Fall_Frames* parameters generating multiple sequences. Since the number of 'non-fall' sequences was much larger than the number of 'fall' sequences (in some parameter combinations, the AR was as high as 4-to-1), we applied random sampling to the 'non-fall' sequences after splitting the data. The size of the random sample was set equal to the number of 'fall' sequences to balance the dataset.

Then, for each obtained sequence, we transposed the *HeightWidthRatio* and *FrameSpeed* columns into a feature row

Figure 3
Transformation of features from frames



vector and added a class label *new_label* that corresponds to whether a "*fall*" was observed in that frame sequence. In this way, we obtained a dataset in which each sequence is represented by a vector of length 2**Window*+1, see Figure 3.

Next, we split the data into training and test sets in a ratio of 4:1. To evaluate the performance of each combination of *Window, Stride,* and *Fall_Frames* parameters, we trained a RandomForest classifier and used grid search to determine the best number of estimators (trees) with 5-fold cross-validation. The best combination was chosen based on the performance of the trained RandomForest model on the test data. The best parameters are presented in Table 1. The worst performance is highlighted in red, and the best is highlighted in green.

For the generalized dataset with primary features, we chose the parameters to ensure that the number of output sequences for the UR Fall and CAUCAFall datasets was not too small and that the test accuracy remained high. Therefore, for the generalized dataset, we set the values *Window*=40, *Stride*=3, *Fall_Frames*=20.

We performed the same procedure of selecting the best parameters *Window, Stride, Fall_Frames* for secondary features extracted from the sequences generated by the sliding window using the tsfresh library. In this case, the vector of secondary features was generated automatically for each sequence. For generating secondary features, we left only *HeightWidthRatio* and *FrameSpeed*, since previous studies [14] have shown that these features have the greatest impact on the classification accuracy. The best parameters are presented in Table 2. For the CAUCAFall dataset with *Window* >= 40 and any values of *Stride, Fall_Frames,* we were able to train the model with 100% accuracy on the test set.

For a generalized dataset with secondary features, the best values *Window=40, Stride=3, Fall_Frames=20.*

Table 1
The best parameters for primary features

Window size	Stride	# Fall frames	Accuracy, %	#Output sequences
		UR Fall Dataset		
40	3	20	95	100
40	3	15	94	124
30	7	20	97	30
50	3	20	92	104
50	3	15	93	120
		CAUCAFall Dataset		
40	3	20	81	272
40	3	15	72	119
30	7	20	80	304
50	3	20	83	272
50	3	15	82	304

Table 2
The best parameters for secondary features

Window size	Stride	# Fall frames	Accuracy, %	#Output sequences
		UR Fall Dataset		
40	3	20	96	77
40	3	20	95	100
30	3	15	94	120
50	3	10	91	138
50	3	10	93	143
		CAUCAFall Dataset		
40	3	20	85	272
40	3	15	100	272
30	7	20	90	304
50	3	20	100	284
50	3	15	100	275

4.3. Primary and secondary feature analysis

Correlation analysis showed that primary and secondary features weakly correlate with the data class ("fall"/"not fall"): the maximum correlation value for secondary features is 0.55 and for primary features – 0.5. At the same time, as expected for primary features, *HeightWidthRatio* and *FrameSpeed* values in later frames of the sequence correlate more strongly with the class label. Thus, the cross-correlation of the primary features *HeightWidthRatio* in neighboring frames is very high (more than 0.9 for ten neighboring frames). Statistical analysis of primary and secondary features within different classes also revealed large differences in secondary features. However, the Kolmogorov-Smirnov and Mann-Whitney tests showed that the differences between classes in terms of features are statistically significant, both for primary and secondary features. In addition, to test the hypothesis that secondary features are better than primary ones for class separability, we performed their principal component analysis (PCA). A 3D visualization of the PCA transformation is shown in Figure 4. In the dataset with secondary features, 99% of the variance is captured by the first four principal components, while for primary features, the first ten principal components account for only 80% of the variance. This indicates that secondary features explain the behavior of the target variable better than primary features.

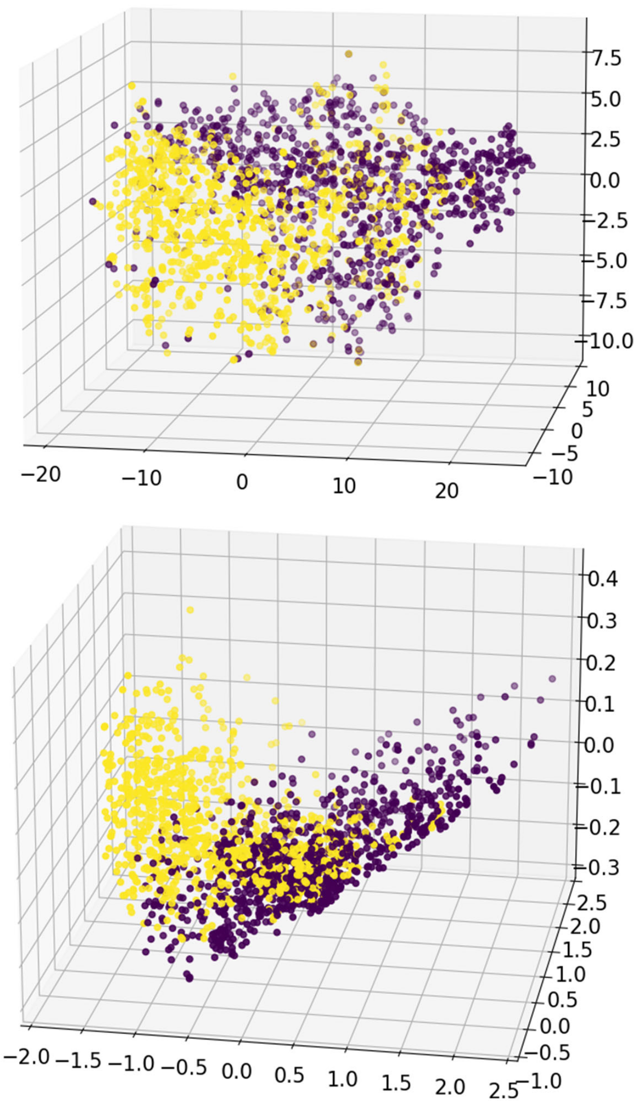
4.4. Training and selecting the best model with auto-sklearn

Then, we create 2 separate datasets with primary and secondary features as described in Section 4.2.

After applying dataset construction and class balancing, we obtained 2 datasets of size 1856 sequences each, where 498 sequences were from the UR Fall dataset and 1358 sequences were from the CAUCAFall dataset.

The dataset with primary features has 80 features generated by sliding window itself), and the dataset with secondary features contains 585 features (generated by applying tsfresh tool to each sequence). The datasets were split into train/test in the ratio of 80% to 20%.

Figure 4
PCA transformation for primary features (top) and secondary features (bottom)



To select the best model from the ensemble, we used the auto-sklearn library, which provides ready-made supervised machine learning. Built on top of the scikit-learn machine learning library, auto-sklearn automatically searches for the correct learning algorithm for a new machine learning dataset and optimizes its hyperparameters [22].

Auto-sklearn starts by using meta-learning where it leverages a database of previously evaluated datasets to make educated guesses about good models for a new dataset. It extracts statistical properties (meta-features) from the dataset, such as the number of features, data sparsity, and correlations between features, to determine similarity to past datasets. Based on these meta-features, it initializes the search process by considering models and hyperparameter settings that worked well on similar datasets. Auto-sklearn doesn't rely on random or grid search for hyperparameters; instead, it uses Bayesian optimization to make intelligent guesses about which hyperparameters will work best. As part of the pipeline search, auto-sklearn also optimizes preprocessing steps and feature engineering techniques. It can automatically handle missing values, scale data, and select best features. After finding multiple well-performing models, auto-sklearn uses a technique called ensemble selection to create a weighted combination of these models. This enhances the robustness and accuracy of predictions.

The optimal model trained on the primary feature dataset is a VotingClassifier utilizing the sklearn gradient boosting algorithm, specifically HistGradientBoostingClassifier, with the following hyperparameters:

early_stopping=False,
l2_regularization=0.0015999096167997723,
learning_rate=0.024544266632179432,
max_iter=512, max_leaf_nodes=28,
min_samples_leaf=5,
random_state=1, validation_fraction=None, warm_start=True.

The model achieved an accuracy of 80% on the test set.

For the secondary feature dataset, the same model architecture was identified as the best, but with adjusted hyperparameters:

early_stopping=True, l2_regularization=2.22710531072698e-08, learning_rate=0.02095996930896952,
max_iter=512, max_leaf_nodes=20,
min_samples_leaf=1,
random_state=1,
validation_fraction=None, warm_start=True. T

This model attained a test accuracy of 96%.

4.5. Feature importance

In this paper, the secondary feature importance has been estimated using *permutation feature importance technique*. Permutation feature importance is a model validation technique that measures the contribution of each feature to the statistical performance of a fitted model on a given tabular dataset. This approach is especially valuable for models that are non-linear or difficult to interpret. It works by randomly shuffling the values of one feature at a time and then measuring the resulting decrease in the model's performance score [23]. By breaking the link between a feature and the target, we can determine how much the model relies on that particular feature.

The 6 most significant secondary features generated by the tsfresh tool and ranked by their importance are shown in Table 3 below. The first two columns contain the primary features and the names of secondary features, and the last 2 columns show their average values and standard deviations.

We measured the time of secondary features generation and class prediction for 1 sequence and 5 sequences of length 40 frames which took 1.3 s and 2.8 s, respectively. Thus, the processing time of feature engineering and prediction is quite fast and takes a few seconds.

The general meaning of the most significant secondary features generated, ranked by their importance, is as follows:

- **HeightWidthRatio_minimum** – represents the lowest value in the time series *x*.
- **HeightWidthRatio_energy_ratio_by_chunks__num_segments_10__segment_focus_9** – calculates the ratio of the sum of squares for a specific segment *i* within *N* total segments to the sum of squares for the entire series. In this case, the series is divided into 10 segments, and the feature is focused on the 9th segment (counting starts from zero).
- **HeightWidthRatio_variation_coefficient** – measures the coefficient of variation, which is the ratio of the standard

Table 3
Top 6 most important secondary features generated by tsfresh tool

Primary features	Secondary features	Importance mean	Importance Std
AR	minimum	0.032	0.006
AR	energy_ratio_by_chunks__num_segments_10__segment_focus_9	0.029	0.006
AR	variation_coefficient	0.010	0.007
BB velocity	autocorrelation__lag_4	0.010	0.004
AR	fft_coefficient__attr_“abs”__coeff_20	0.008	0.003
BB velocity	change_quantiles__f_agg_“mean”__isabs_“ru”__qh_0.4__ql_0.0	0.008	0.003

deviation to the mean, providing a relative measure of variability around the mean.

- **FrameSpeed_autocorrelation__lag_4** – computes the value of an aggregation function (like variance or mean) over the autocorrelation of the time series *x* with a lag of 4.
- **HeightWidthRatio_fft_coefficient__attr_“abs”__coeff_20** – refers to the Fourier coefficients obtained from the one-dimensional discrete Fourier Transform of the time series *x*, computed using the fast Fourier transform algorithm. This specific feature corresponds to the 20th coefficient and captures the absolute value.
- **FrameSpeed_change_quantiles__f_agg_“mean”_*isabs*_“ru”__ qh_0.4__ql_0.0** – first defines a range using the quantiles ql and qh of the distribution of *x*. Then, it calculates the mean of the absolute values of consecutive changes in the series *x* that fall within this defined range.

4.6. Results discussion

In this study, we explored the effectiveness of primary and secondary features in detecting falls, applying a sliding window approach to enhance the size and variability of the training data. The datasets were created by segmenting sequences from the UR Fall and CAUCAFall datasets, resulting in two feature sets: the primary dataset with 80 features generated directly by the sliding window, and the secondary dataset with 585 features generated using the tsfresh tool.

After preprocessing and balancing the data, we employed the AutoML approach via the auto-sklearn library to select the optimal models for both datasets. This approach proved effective, with the best model for the primary feature dataset achieving an 80% test accuracy using a HistGradientBoostingClassifier. For the secondary features, the same model setup with adjusted hyperparameters improved accuracy to 96%, demonstrating the advantages of secondary feature extraction.

The feature importance analysis for secondary features showed a critical role of the first 6 features in the model’s performance. These findings also indicate that secondary features derived from just two primary features – BB AR and velocity – are highly informative for building robust machine learning model for fall detection.

Moreover, the processing times for generating secondary features and making predictions were found to be relatively fast, taking only a few seconds per sequence, which highlights the practical applicability of this approach in real-time fall monitoring systems.

Overall, the use of secondary features and AutoML significantly improved model accuracy compared to our previous work.

5. Conclusion

A new vision-based method for monitoring falls of elderly people using machine learning techniques and an approach based on the use of a sliding window and secondary feature generation is proposed. The results obtained in this study are more reliable compared to our previous work [14] due to the use of 2 orders of magnitude more data for training and selection of the best model. In addition, using a sliding window approach allowed us to increase the number of sequences analyzed. In this paper, it was shown that the proposed methodology for creating a detector using a sliding window for data transformation is promising. This is confirmed by recent publications that use a similar approach to build a classifier for data obtained from wearable sensors [24] and remote radar monitoring [25]. Our results prove that the use of tsfresh tool, which generates new secondary features from only 2 primary ones, related to the AR and velocity of the BB covering a person, makes it possible to build an ensemble of good machine learning models. Further use of the auto-sklearn tool allows to select the best ensemble model with histogram-based gradient boosting classifiers and achieve test accuracy of 96%, that is slightly better compared to using a similar model for sensor data in [24]. In our study, the effectiveness of tsfresh and auto-sklearn tools was confirmed using data from two datasets UR Fall Dataset and CAUCAFall Dataset as well as their combination.

The experiments conducted showed that secondary features describe the behavior of the target variable better than primary features, which actually led to an improvement in the accuracy of the optimized model of 96% compared to using primary ones, where it had a value of 80%.

This study underscores the potential of combining the sliding window approach with automated feature extraction and model selection techniques for fall detection providing a highly efficient and accurate solution suitable for real-time applications.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

Data available on request from the corresponding author upon reasonable request.

Author Contribution Statement

Boris Assanovich: Conceptualization, Methodology, Validation, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Katsiaryna Kosarava:** Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – review & editing, Visualization.

References

[1] World Health Organization. (2021). *Falls*. Retrieved from: https://www.who.int/news-room/fact-sheets/detail/falls

[2] Gutiérrez, J., Rodríguez, V., & Martin, S. (2021). Comprehensive review of vision-based fall detection systems. *Sensors*, *21*(3), 947. https://doi.org/10.3390/s21030947

[3] Alam, E., Sufian, A., Dutta, P., & Leo, M. (2022). Vision-based human fall detection systems using deep learning: A review. *Computers in Biology and Medicine*, *146*, 105626. https://doi.org/10.1016/j.compbiomed.2022.105626

[4] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, *2016*, 779–788.

[5] Lezzar F., Benmerzoug, D., & Kitouni, I. (2020). Camera-based fall detection system for the elderly with occlusion recognition. *Applied Medical Informatics*, *42*(3), 169–179.

[6] Charfi, I., Miteran, J., Dubois, J., Atri M., & Tourki, R. (2012). Definition and performance evaluation of a robust SVM based fall detection solution. In *2012 Eighth International Conference on Signal Image Technology and Internet Based Systems*, 218–224. https://doi.org/10.1109/SITIS.2012.155

[7] Charfi, I., Miteran, J., Dubois, J., Atri, M., & Tourki, R. (2013). Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and Adaboost-based classification. *Journal of Electronic Imaging*, *22*(4), 17.

[8] Beddiar, D. R., Oussalah, M., & Nini, B. (2022). Fall detection using body geometry and human pose estimation in video sequences. *Journal of Visual Communication and Image Representation*, *82*, 103407.

[9] Kwolek, B., & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine*, *117*(3), 489–501. http://doi.org/10.1016/j.cmpb.2014.09.005

[10] Zhang, J., Wu, C., & Wang, Y. (2020). Human fall detection based on body posture spatio-temporal evolution. *Sensors*, *20*(3), 946. https://doi.org/10.3390/s20030946

[11] Zhao, D., Song, T., Gao, J., Li, D., & Niu, Y. (2024). YOLO-fall: A novel convolutional neural network model for fall detection in open spaces. *IEEE Access*, *12*, 26137–26149. http://doi.org/10.1109/ACCESS.2024.3362958

[12] Yhdego, H., Paolini, C., & Audette, M. (2023). Toward real-time, robust wearable sensor fall detection using deep learning methods: A feasibility study. *Applied Sciences*, *13*(8), 4988. https://doi.org/10.3390/app13084988

[13] Shukla, K., Holderbaum, W., Theodoridis, T., & Wei, G. (2024). Enhancing gearbox fault diagnosis through advanced feature engineering and data segmentation techniques. *Machines*, *12*(4), 261.

[14] Kosarava, K., & Assanovich, B. (2021). A simple indoor fall control system for the elderly based on the analysis of object bounding box parameters. In *Proceedings of the International Conference on Pattern Recognition and Information Processing*, 92–96.

[15] tsfresh. (n.d.). *Introduction-why tsfresh?* Retrieved from: https://tsfresh.readthedocs.io/en/latest/text/introduction.html

[16] Kausar, F., Awadalla, M., Mesbah, M., & AlBadi, T. (2022). Automated machine learning based elderly fall detection classification. *Procedia Computer Science*, *203*, 16–23. https://doi.org/10.1016/j.procs.2022.07.005

[17] Reimann, H., & Bruijn, S. M. (2024). The condition for dynamic stability in humans walking with feedback control. *PLOS Computational Biology*, *20*(3), e1011861. http://doi.org/10.1371/journal.pcbi.1011861

[18] Kamberaj, H. (2021). *Classical mechanics*. Germany: Walter de Gruyter GmbH & Co KG.

[19] Gahramanova, A. (2019). Locating centers of mass with image processing. *Undergraduate Journal of Mathematical Modeling: One+ Two*, *10*(1), 1. http://doi.org/10.5038/2326-3652.10.1.4906

[20] Eraso, J. C., Muñoz, E., Muñoz, M., & Pinto, J. (2022). Dataset CAUCAFall. *Mendeley Data*, *4*. http://doi.org/10.17632/7w7fccy7ky.4

[21] tsfresh. (n.d.). *Overview on extracted features*. Retrieved from: https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html

[22] Feurer, M., Eggensperger, K., Falkner, S., Lindauer, M., & Hutter, F. (2022). Auto-Sklearn 2.0: Hands-free AutoML via meta-learning. *Journal of Machine Learning Research*, *23*(261), 1–61.

[23] Breiman, L. (2001). Random forests. *Machine Learning*, *45*(2001), 5–32.

[24] Nishiyama, D., Arita, S., Fukui, D., Yamanaka, M., & Yamada, H. (2024). Accurate fall risk classification in elderly using one gait cycle data and machine learning. *Clinical Biomechanics*, *115*, 106262. https://doi.org/10.1016/j.clinbiomech.2024.106262

[25] Rodriguez, J., Mercuri, M., Karsmakers, P., Soh, P. J., Leroux, P., & Schreurs, D. (2013). Automatic fall detector based on sliding window principle. In *34th WIC Symposium on Information Theory in the Benelux and the Third joint WIC/IEEE SP Symposium on Information Theory and Signal Processing in the Benelux*, 215–219.

How to Cite: Assanovich, B., & Kosarava, K. (2024). Vision-Based Fall Detector for Elderly Based on Sliding Window Approach and Feature Engineering. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS42024100