applied sciences

MDPI

Article

Fall Recognition Based on Time-Level Decision Fusion Classification

Juyoung Kim [1], Beomseong Kim [2] and Heesung Lee [1,*]

[1] Department of Railroad Electrical and Electronics Engineering, Korea National University of Transportation, Uiwang-si 16101, Republic of Korea; myforte8@ut.ac.kr
[2] Department of Artificial Intelligence, Gyeonggi University of Science and Technology, Siheung-si 15073, Republic of Korea; beomseongkim@gtec.ac.kr
* Correspondence: hslee0717@ut.ac.kr; Tel.: +82-31-460-0574

Abstract: We propose a vision-based fall detection algorithm using advanced deep learning models and fusion methods for smart safety management systems. By detecting falls through visual cues, it is possible to leverage existing surveillance cameras, thus minimizing the need for extensive additional equipment. Consequently, we developed a cost-effective fall detection system. The proposed system consists of four modules: object detection, pose estimation, action recognition, and result fusion. Constructing the fall detection system involved the utilization of state-of-the-art (SOTA) models. In the fusion module, we experimented with various approaches, including voting, maximum, averaging, and probabilistic fusion. Notably, we observed a significant performance improvement with the use of probabilistic fusion. We employed the HAR-UP dataset to demonstrate this enhancement, achieving an average 0.84% increase in accuracy compared to the baseline, which did not incorporate fusion methods. By applying our proposed time-level ensemble and skeleton-based fall detection approach, coupled with the use of enhanced object detection and pose estimation modules, we substantially improved the robustness and accuracy of the system, particularly for fall detection in challenging scenarios.

Keywords: fall recognition; object detection; pose estimation; action recognition; fusion; skeleton-based

Citation: Kim, J.; Kim, B.; Lee, H. Fall Recognition Based on Time-Level Decision Fusion Classification. *Appl. Sci.* **2024**, *14*, 709. https://doi.org/10.3390/app14020709

Academic Editors: Yutaka Ishibashi, El-Sayed El-Alfy, Motaz Alfarraj and Abdul Jabbar Siddiqui

Received: 17 October 2023
Revised: 25 December 2023
Accepted: 10 January 2024
Published: 14 January 2024

1. Introduction

As perceptions of safety in society change and legal regulations for safety become stronger, the demand for safety management is increasing. However, there is currently a shortage of personnel for safety management, and efficiency is not at its optimum level. An efficient safety management system is essential for addressing these challenges. A recent and increasing trend in research focuses on the incorporation of artificial intelligence technologies into safety management systems [1–4]. By utilizing artificial intelligence, it is possible to detect and respond to accidents involving onsite workers and facility users in real time, thereby minimizing additional damage. In South Korea, accidents caused by falls are the most common type of industrial disasters [5]. Therefore, in this study, we propose an algorithm for real-time fall detection using state-of-the-art deep learning models for a smart safety management system.

In parallel with our research, a range of investigations are underway to detect falls using diverse sensor technologies [6–27]. These include wearable devices, accelerometers, radars, and video sensors. Furthermore, researchers are actively exploring the possibility of fall detection in multimodal scenarios by utilizing a combination of different sensors [23–27]. However, when employing a wearable sensor [6–9], an individual must wear the sensor for fall monitoring, which is inconvenient, and concerns about battery life arise. Moreover, additional equipment is required to use accelerometers or radar sensors [10–13], and, especially in the radar case, the falls of objects may be mistaken as human falls. Utilizing multimodal approaches to detect falls can enhance performance; however, a potential drawback

is processing time degradation owing to the additional tasks required for aligning data from multiple sensors.

Currently, numerous cameras are installed in various locations. Hence, adopting video sensors for fall detection allows seamless integration with already established camera systems, eliminating the requirement for additional hardware and ensuring cost-effectiveness. Moreover, this approach eliminates the need for users to fulfill specific conditions, resulting in reduced user resistance. Therefore, we aim to detect falls using video sensors.

Recently, with the emergence of deep learning, significant advancements have been made in computer vision technology, leading to an increasing number of attempts to solve various issues using computer vision techniques. In particular, the utilization of video data has made it possible to detect falls, and several studies have shown promising outcomes [14–22]. In line with this trend, we propose an algorithm for fall detection using deep learning-based computer vision technology.

The proposed fall detection system comprises four modules: object detection, pose estimation, action recognition, and result fusion. The first module, the object detection module, determines the presence of a person and, if detected, provides an output regarding the person's location in the video. In the second module, the pose estimation module obtains the output from the first module and processes the image section containing the person (rather than the entire image). This process generates skeleton data for pose estimation. The third module, the action recognition module, recognizes the pattern of changes in the joint points of the individual in the video to output the fall detection results. In the fourth module, the result fusion module, the results from the third module are fused using various methods to enhance the robustness and performance, ultimately producing the final result. The main contributions of this study are as follows:

1. Using a time-level ensemble, we aggregate the continuous outcomes of the modules. Thus, the system has enhanced robustness and overcomes challenges under difficult conditions. Various fusion methods, including voting-, averaging-, maximum-, and probability-based approaches, are employed.
2. Our fall detection system is based on skeleton data, which means that the accuracy of person detection and pose estimation significantly influences the fall detection performance. Through comprehensive performance comparisons with several detection models and pose estimation models, we propose suitable person detection and joint point estimation algorithms for fall detection. In the experiment, YOLOv3 [28] and YOLOv7 [29] were employed to detect individuals. Moreover, AlphaPose [30], a convolutional neural network (CNN)-based approach, and ViTPose [31], which is based on a vision transformer (ViT) [32], were utilized to estimate joint points.

This paper is organized as follows: In Sections 2 and 3, we provide a comprehensive overview of relevant studies that have attempted to detect falls and the details of the entire system process and of each module used to determine falls, respectively. The dataset, experimental setup, and experimental results are presented in Section 4. Finally, conclusions are drawn in Section 5.

2. Related Work

Fall can result in physical and mental injuries, and numerous studies have proposed fall detection methods. Automatic fall detection prevents situations in which injuries caused by falls result in an inability to request aid. Moreover, a reduction in recovery time can be anticipated through prompt first aid. In previous research, various sensors, including accelerometers, radars, and vision sensors, have been employed for fall detection, and the concept of multimodal detection systems has also been introduced.

2.1. Accelerometer-Based Methods

Some studies aim to detect falls by attaching wearable accelerometers to the body and utilizing the values obtained from these sensors [6–9]. Ref. [6] compared numerous previous studies that used accelerometers for fall detection. In [7], a triaxial accelerometer

was worn on the head to analyze acceleration patterns, and if the threshold was exceeded, it was classified as a fall. In [8,9], features were extracted from the values obtained from a triaxial accelerometer, and machine learning-based classifiers were employed to detect falls. In [8], an accelerometer was attached to the chest, and its values were used as inputs for a classifier to classify falls and activities of daily living (ADL). The authors compared the outcomes by employing various machine learning-based classifiers. In [9], a pendant-style triaxial accelerometer that can be worn around the neck was proposed. The authors used long short-term memory (LSTM) to detect falls. Using an accelerometer for fall detection can capitalize on the sensitivity of the sensor itself, resulting in a high recall performance, which classifies actual falls as falls. However, when using an accelerometer attached to a limited body area, it may be challenging to reflect the overall body movement accurately, leading to a higher likelihood of misidentifying activities similar to falls as falls. Additionally, the requirement for wearing an accelerometer necessitates user participation and consent, thereby introducing potential constraints on user convenience.

2.2. Radar-Based Methods

Research is also underway to detect falls by installing radars in the surrounding environment and analyzing the obtained signals [10–13]. Studies on fall detection using radar sensors were presented in [10]. A custom-made radar sensor and support vector machine (SVM) were employed in [11] to detect falls. In [12,13], a frequency-modulated continuous-wave (FMCW) radar was installed on the ceiling to capture signals for fall detection. In [12], signal preprocessing was applied, followed by the utilization of an SVM as a classifier to detect falls. In contrast, [13] employed a random forest algorithm as a classifier. Radar-based fall detection offers the advantage of detecting falls even in low-light conditions without performance degradation. However, it faces drawbacks related to the installation cost of radar sensors and the potential misclassification of non-human falls, such as objects dropping, as falls.

2.3. Vision-Based Methods

Advancements in deep learning have resulted in significant progress in computer vision technology. Consequently, there has been a notable increase in research aimed at utilizing RGB and RGB-D cameras to recognize human actions and detect falls [14–22]. These studies utilized the capabilities of deep learning algorithms to process visual data, facilitating the recognition of human behavior and detection of falls with enhanced accuracy and efficiency. In [14], human behavior was recognized by utilizing estimated human poses from an RGB-D sensor. In [15], fall detection was performed by extracting human silhouettes from RGB-D images through background removal. In [16], fall detection was performed by obtaining 3D skeleton values from RGB-D images. With the advancement of pose estimation algorithms for 2D images, the authors of [17–21] employed RGB images for fall detection. They estimated human poses from RGB images and used these estimations to detect falls. In [17], a recurrent neural network (RNN) utilizing a pose vector as the input was employed for fall detection. In [18], fall detection was conducted using random forest, SVM, and multilayer perceptron (MLP), and their respective performances were compared. The authors of [19,20] employed a spatial temporal graph convolutional network (ST-GCN) for fall detection. Especially in [20], a pre-trained ST-GCN on action recognition data, not fall data, was utilized to extract features for action. Subsequently, the extracted features were used as input to the encoder–decoder model, and if the restored features from the decoder did not exceed a specified threshold, they were recognized as a fall. Furthermore, refs. [21,22] detected falls using feature-level fusion and a 3D-CNN. Vision-based fall detection can utilize the global posture of a person in an image to detect falls. Long-distance fall detection using a vision sensor is advantageous for user convenience, and the integration with existing surveillance cameras allows for cost reduction by minimizing the need for additional equipment.

2.4. Multimodal-Based Methods

Other studies have utilized data from various sensors such as cameras, accelerometers, radars, vital signs, gyroscopes, and magnetometers to detect falls or recognize actions [23–27]. In these studies, falls were detected in a multimodal context by extracting features from signals acquired from two or more sensors and then aggregating these features to be used as input for the classifiers. Multimodal-based approaches can achieve high accuracy by integrating data from various sensors. However, processing time may be compromised as values from different sensors need to be aligned and handled, and the system design can become more complex. The advantages and disadvantages of detecting falls using various sensors, as well as the strengths and weaknesses of various vision-based fall detection algorithms under investigation in this paper, are summarized in Tables 1 and 2, respectively.

Table 1. Summary of pros and cons of fall detection methods using various sensors.

Sensor	Pros	Cons
Accelerometer	Sensor sensitivity	Limitation in catching the global movements of the body caused by the number of attached sensors. Devices have to be worn.
Radar	Low-light condition performance Privacy friendliness	Limitation in recognizing different postures. Difficulty in distinguishing between the fall of objects and humans.
Vision	Capture of global body posture is possible No requirements are imposed on users	Vulnerability in low-light conditions.
Multimodal	Enhancement in performance through the utilization of abundant information	Decreased inference time due to processing of various sensor values.

Table 2. Vision-based fall detection studies with some pros and cons.

Author	Pros	Cons	Dataset
Panahi, L., 2018 [15]	Fit a person's silhouette on the ellipses and observe the changes in the ellipses and detect fall. In general, the extraction accuracy of silhouette is higher than skeleton coordinate prediction accuracy, so the performance degradation due to the silhouette step is less.	It is difficult to detect detailed body movements because the method creates an ellipse by fitting silhouettes and observes changes in the ellipse to detect falls.	UR-Fall
Lie, W. N., 2018 [17]	It is possible to extract the temporal characteristics of the skeleton sequence because the predicted skeleton is used as an input for the LSTM	There is less temporal information using only 8 frames of input. And it is difficult to convey the relationship between the joints because the skeleton coordinate value itself is used as the input.	Private dataset
Keskes, O., 2021 [16]	This study uses 3D skeleton sequences as inputs and classifies them using ST-GCN models, so all spatial–temporal information can be considered.	Since 3D skeleton predictions are less accurate than 2D skeleton predictions, incorrect 3D coordinate predictions are likely to occur in a real-world environment. This may result in performance degradation.	TST v2 Fallfree dataset
Ramirez, H., 2021 [18]	The authors present a comparison of various ML models as classifiers.	Because it detects falling in frame, input data do not contain temporal information.	HAR-UP UR-FALL

Table 2. *Cont.*

Author	Pros	Cons	Dataset
Galvão, Y. M., 2021 [20]	The authors trained an encoder–decoder model using action features extracted from the pre-trained ST-GCN. By using only data on activities of daily living (ADL) as the dataset for training the encoder–decoder model, it can address the scarcity of fall data.	It is trained on ADL data, not fall data, and if the likelihood of an action as an ADL is below a certain level, it is classified as a fall. So there is a possibility of classifying actions as falls that are neither falls nor ADL.	PRECIS HAR, HAR-UP UR-Fall
Kim, J., 2023 [19]	Using the ST-GCN model as a classifier, spatial–temporal information can be considered.	Person detection that precedes the extraction of skeleton data does not work well.	HAR-UP
Alanazi, T., 2023 [22]	The proposed image fusion method converts multiple frames into a richly informative image to detect falls.	Since it is based on segmentation, a person's clothes and belongings can be reflected in the silhouette and post-processing is required for incorrect segmentation	Le2i fall detection dataset

3. Fall Detection System

3.1. Person Detection

The complete process of the proposed system is depicted in Figure 1. The initial step in the proposed fall detection method is person detection. The inference speed of each module is important for real-time fall detection. Therefore, in this study, the YOLO model [28,29,33], a highly regarded real-time detector, was employed for person detection. This allows for the rapid transfer of person detection results to the pose-estimation module. YOLO [33] was designed to overcome the slow inference speed of two-stage detectors. It performs bounding-box regression and classification of the feature maps obtained by passing the CNN without requiring separate region proposals. This approach addresses the low detection speed observed in previous models. In this system, YOLO, which is known for its fast detection speed, was used to construct a real-time fall detection system. A pre-trained YOLOv3 was first employed to detect persons. However, the pre-trained YOLOv3 struggled to effectively detect fallen persons. Consequently, YOLOv7, an enhanced version, was used to conduct additional experiments. YOLOv3 [28] employs a residual connection-based backbone for feature extraction. It regresses the bounding box onto feature maps of various resolutions. YOLOv7 [29] employs a concatenation-based backbone that reuses features learned from previous layers, thus increasing the parameter efficiency. YOLOv7 also embraces the neural architecture search (NAS) to achieve an efficient model through scaling. YOLOv7 exhibits a better Pareto front than YOLOv3. The results of person detection using YOLOv3 and YOLOv7 are shown in Figure 2. Only the region containing the participants was defined as a region of interest, and person detection was performed in this area, as shown in Figure 2.

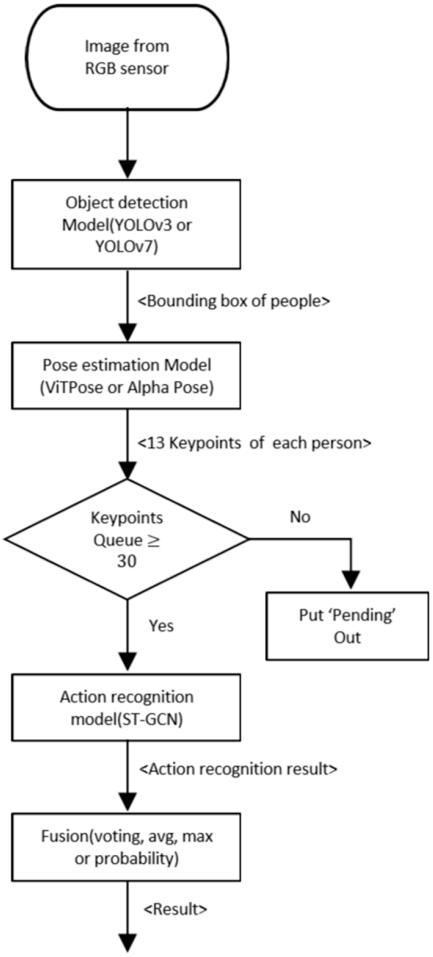
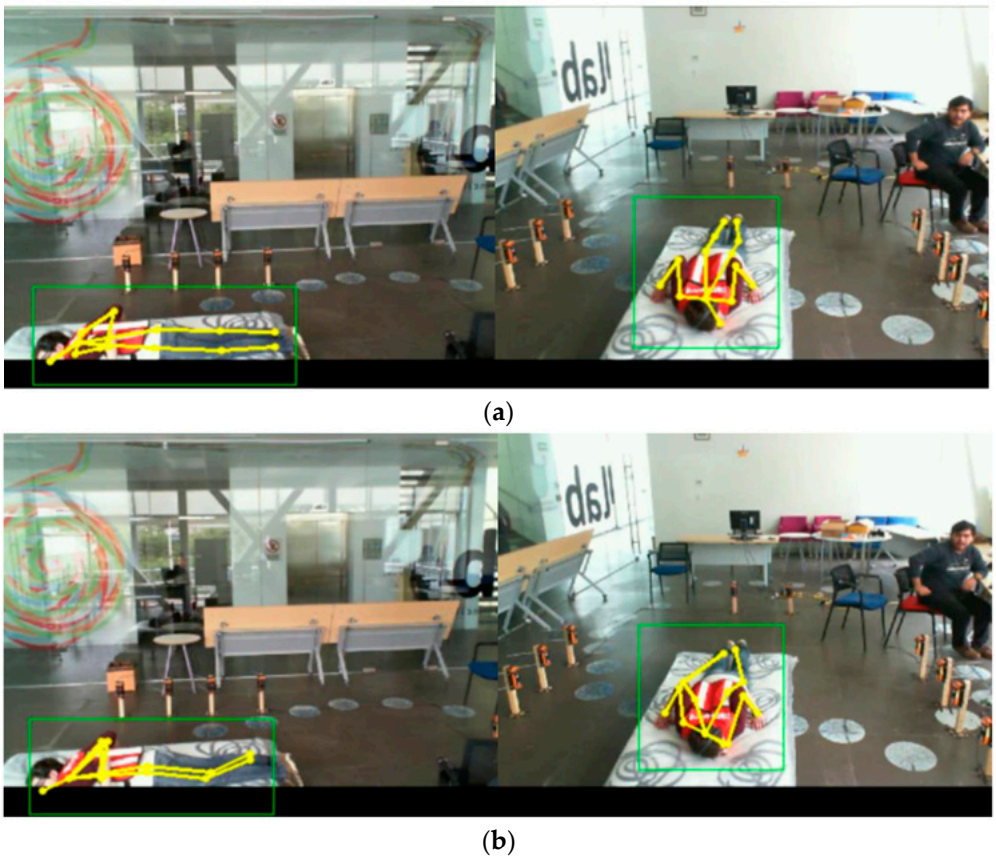


Figure 1. The complete process of the proposed system.

Figure 2. Results of person detection using YOLOv3 and YOLOv7: (**a**) detection result of YOLOv3 for normal activity; (**b**) detection result of YOLOv3 for fall activity; (**c**) detection result of YOLOv7 for normal activity; and (**d**) detection result of YOLOv7 for fall activity.

3.2. Skeleton-Based Feature Extraction

Detecting falls using a skeleton-based approach has the advantage of preventing the learning of irrelevant information such as the background, thus leading to exceptional generalization performance. Furthermore, skeleton-based methods offer the advantage of enabling individualized fall detection for multiple individuals within a single video. In this study, the pre-trained AlphaPose [30] and ViTPose [31] were used to obtain skeleton data, and their respective results were compared. AlphaPose uses a CNN-based architecture to generate skeleton data as its output, whereas ViTPose employs the encoder structure of ViT [32] as its backbone. CNNs are among the most effective approaches for extracting features from images. However, very recently, research has demonstrated better performance in image feature extraction using the transformer [34] architecture than that achieved with models based on CNNs. This trend has prompted the development of ViTPose, which leverages the ViT structure as a complementary solution to the CNN-based AlphaPose to enhance pose estimation. These models produce 17 key points as their outputs. We utilized 13 key points, excluding the eyes and ears from both sides as these have a relatively minor impact on fall detection results. We compared the experimental outcomes of AlphaPose and ViTPose using the selected key points. Figure 3 shows an example of the skeleton output obtained using AlphaPose v0.2.0 and ViTPose.



(**a**)

(**b**)

Figure 3. Skeleton data output of ViTPose and AlphaPose: (**a**) result of ViTPose, and (**b**) result of AlphaPose.

3.3. Classification

In the proposed system, the classifier uses the key points obtained from the pose estimation module as inputs to classify the actions of the detected persons in the image as either falling or as normal behavior. Using these key points as inputs instead of all RGB pixels in the video prevents overfitting, thereby improving generalization performance. Falls occur because of a series of movements in which balance is lost and the body makes

contact with the ground. To recognize such temporal changes in movements, it is crucial for the classifier to understand not only individual images but also how changes unfold in a video. In each video frame, the classifier should learn the configuration of the person's body and understand how the body arrangement changes across consecutive frames. Therefore, we employed ST-GCN [35] as the classifier. This choice was motivated by its ability to grasp the formation of the body in each video frame and capture how the body's structure evolves over successive frames. In ST-GCN, the skeleton coordinates are organized into graph-formatted data for input. This model recognizes the spatial patterns between the key points in the skeleton graph data obtained from each frame, and detects the temporal patterns between skeleton graph data acquired from multiple frames. By utilizing both spatial and temporal information, ST-GCN can detect falls effectively. The operating process of the ST-GCN is shown in Figure 4.

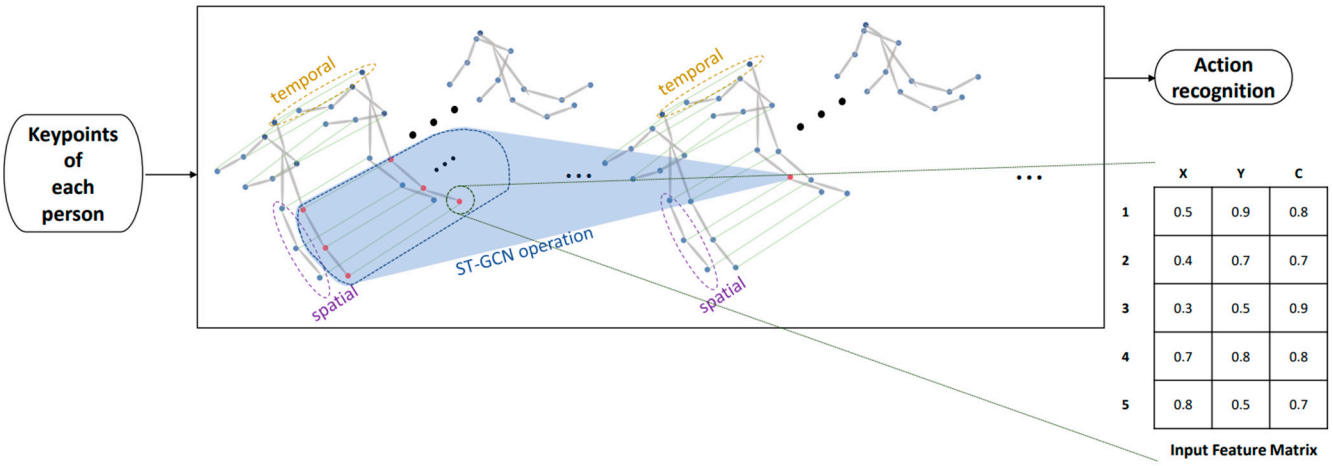


Figure 4. Structure of ST-GCN.

In one layer of the ST-GCN, the spatial kernel size, which represents the distance considering the connections between adjacent joints, is set to 1. This means that only the most adjacent nodes are included in the spatial operation range. The spatial receptive field of one ST-GCN operation is considered to involve three nodes. As shown in Figure 4, three nodes get involved in one ST-GCN operation. For instance, for the elbow node, the spatial receptive field of the ST-GCN operation includes the wrist, elbow, and shoulder nodes. Next, the parameter for the temporal kernel size, which determines how many frames participate in the temporal operation, is set to 9. In other words, the corresponding frame and the surrounding four frames are included in the temporal receptive field. It is not the length of the skeleton sequence but rather the size of the temporal kernel that is 9. Therefore, conducting ST-GCN operations layer by layer enables the recognition of patterns in spatial–temporal information. By utilizing the ST-GCN, we can overcome the limitations of traditional classifiers in handling unseen data and enhance generalization performance.

3.4. Time-Level Decision Fusion

Assembling multiple models to enhance generalization performance is a common practice [36–39]. Combining the outcomes of separately trained models to improve the overall performance is a typical ensemble approach. However, falls involve temporal changes in posture. Therefore, if the skeleton data at time intervals $[t, t+1, \ldots, t+n]$ indicate a fall, it is likely that the skeleton data at time intervals $[t-1, t, \ldots, t+n-1]$ and $[t+1, t+2, \ldots, t+n+1]$ also correspond to a fall. Therefore, in this study, the results from different time points of the classifier were assembled. Fluctuations in the skeleton sequence can occur due to incorrect bounding boxes in the person detection stage, leading to a degradation in pose estimation performance or performance issues within the pose estimation stage itself. Fusion can help alleviate the impact of skeleton sequence fluctua-

tions on the classification performance of the ST-GCN. Additionally, situations where falls are challenging to detect, such as actions similar to falling, can be overcome by fusing the results from multiple perspectives of the ST-GCN. To aggregate the classifiers at the time level, we denote the keypoint values obtained at time t as Y_t. The input features X_t for a classifier are defined as follows:

$$X_t = [Y_{t-w+1}\ Y_{t-w+2}\ \ldots\ Y_t] \tag{1}$$

where w denotes the key points' list size. The output result ω_t of the classifier is defined as

$$\omega_t = f(X_t) \tag{2}$$

where $f(\cdot)$ represents the classifier function. ω_t is represented as a vector indicating the confidence scores for each class and can be expressed as follows:

$$\omega_t = \left[\omega_t^1\ \omega_t^2\ \ldots\ \omega_t^c\right]^T, \text{ where } 0 \le \omega_t^i \le 1 \tag{3}$$

Here, c represents the number of predefined classes. The final result ω^{out}, obtained by fusing ω_t from time t to $t - N + 1$, is defined as follows:

$$\omega^{out} = g([\omega_{t-N+1}\ \omega_{t-N+2}\ \ldots\ \omega_t]) \tag{4}$$

where N denotes the number of feature windows, and $g(\cdot)$ represents the function that fuses the results of N classifiers. The fusion module process is summarized in Figure 5.

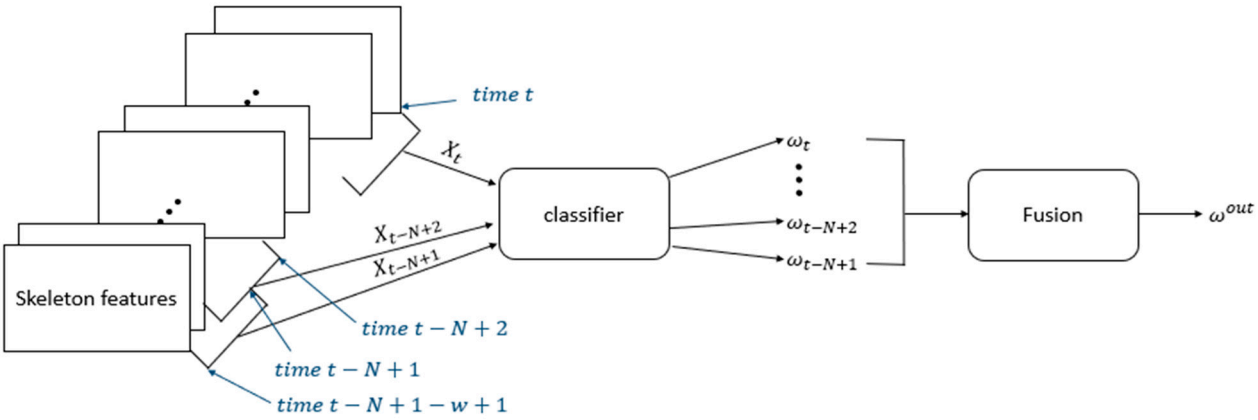


Figure 5. Skeleton features are fed successively into the classifier, and the classifier outputs are aggregated in the fusion step to obtain ω^{out}.

We used the following methods to obtain $g(\cdot)$.

3.4.1. Voting Method

The final output ω^{out} is obtained by using Equations (5)–(7) to determine the most frequent class among the results of N classifiers:

$$B_k^i = \begin{cases} \omega_k^i = 1,\ if max\left[\omega_k^1\ \omega_k^2\ \ldots\ \omega_k^c\right] = \omega_k^i \\ \omega_k^i = 0,\ otherwise \end{cases} \tag{5}$$

$$\omega^{voting,\ i} = \sum_{k=t-N+1}^{t} B_k^i \tag{6}$$

$$\omega^{out} = \underset{i=1,2,\ldots,C}{\text{argmax}}\,\omega^{voting,\ i} \tag{7}$$

3.4.2. Average Method

The final output ω^{out} is determined by selecting the class with the highest average confidence score among the results of N classifiers as defined in Equation (8):

$$\omega^{out} = \underset{i=1,2,\ldots,C}{\operatorname{argmax}} \left[\frac{1}{N} \sum_{k=t-N+1}^{t} \omega_k^i \right] \tag{8}$$

3.4.3. Maximum Method

The final output ω^{out} is obtained by selecting the class with the highest confidence score among the results of N classifiers. The equation used is as follows:

$$\omega^{out} = \underset{i=1,2,\ldots C}{\operatorname{argmax}} \left[max \left[\omega_{t-N+1}^i \; \omega_{t-N+2}^i \; \ldots \; \omega_t^i \right] \right] \tag{9}$$

3.4.4. Probabilistic Fusion Method

A probabilistic approach is employed to aggregate the results from different time-point classifiers. In the fusion stage, the N results are fused using belief. As a new measure, we define belief as the posterior probability of $\omega^{out} = i$ $(i = 1, 2, \ldots, C)$ conditioned on all time-level classifiers and combine the classifiers by using belief [39]. The posterior probability $bel_n(\omega^{out} = i)$ at time levels from 1 to n is defined as follows:

$$bel_n(\omega^{out} = i) = p(\omega^{out} = i \,|I_{1:n}) \tag{10}$$

In this case, $p(\omega^{out} = i)$ can be obtained from the confidence score of the ith class of a classifier, and I_k represents the event where the classifier produces an output using the skeleton features in the kth frame. Motivated by [39,40], we calculate the current posterior distribution from the previous time step's posterior distribution as shown in the following equation:

$$bel_n(\omega^{out} = i) = \frac{p\left(I_n | \omega^{out} = i, I_{1:n-1} \right) p(\omega^{out} = i | I_{1:n-1})}{p(I_n | I_{1:n-1})} \tag{11}$$

where $p\left(I_n | \omega^{out} = i, I_{1:n-1} \right)$ is the probability of the nth classifier when previous $n-1$ classifiers are given. Because each time-level classifier makes inferences separately and independently from the other classifiers, we obtain

$$bel_n(\omega^{out} = i) = \frac{p\left(I_n | \omega^{out} = i \right) p(\omega^{out} = i | I_{1:n-1})}{p(I_n | I_{1:n-1})} \tag{12}$$

Applying Bayes' rule to $p(I_n | \omega^{out} = i)$ in (12), the following equation can be derived:

$$p(I_n | \omega^{out} = i) = \frac{p(\omega^{out} = i | I_n) p(I_n)}{p(\omega^{out} = i)} \tag{13}$$

Therefore, Equation (12) becomes the following equation:

$$bel_n(\omega^{out} = i) = \frac{p(\omega^{out} = i | I_n) p(I_n)}{p(\omega^{out} = i)} \frac{p(\omega^{out} = i | I_{n-1})}{p(I_n | I_{n-1})} \tag{14}$$

where $p(I_n)$ is the probability of the nth classifier, and $p(\omega^{out} = i)$ is the probability that, when an unknown action is given, the class of the action is i. Analogously, we use the following equation for the opposite event:

$$1 - bel_n(\omega^{out} = i) = \frac{p(\omega^{out} \neq i | I_n) p(I_n)}{p(\omega^{out} \neq i)} \frac{p(\omega^{out} \neq i | I_{n-1})}{p(I_n | I_{n-1})} \tag{15}$$

By dividing Equation (14) by Equation (15), Equation (16) is obtained:

$$\frac{bel_n(\omega^{out}=i)}{1-bel_n(\omega^{out}=i)} = \frac{p(\omega^{out}=i|I_n)}{p(\omega^{out}\neq i|I_n)} \frac{p(\omega^{out}=i|I_{n-1})}{p(\omega^{out}\neq i|I_{n-1})} \frac{p(\omega^{out}\neq i)}{p(\omega^{out}=i)} \tag{16}$$

We can express Equation (16) as follows:

$$\frac{bel_n(\omega^{out}=i)}{1-bel_n(\omega^{out}=i)} = \frac{p(\omega^{out}=i|I_n)}{1-p(\omega^{out}=i|I_n)} \frac{p(\omega^{out}=i|I_{n-1})}{1-p(\omega^{out}=i|I_{n-1})} \frac{1-p(\omega^{out}=i)}{p(\omega^{out}=i)} \tag{17}$$

$p(\omega^{out}=i|I_{1:n-1})$ in Equation (17) can be expressed as $bel_{n-1}(\omega^{out}=i)$ according to the definition of belief. Therefore, Equation (17) can be expressed as follows:

$$\frac{bel_n(\omega^{out}=i)}{1-bel_n(\omega^{out}=i)} = \frac{p(\omega^{out}=i|I_n)}{1-p(\omega^{out}=i|I_n)} \frac{bel_{n-1}(\omega^{out}=i)}{1-bel_{n-1}(\omega^{out}=i)} \frac{1-bel_0(\omega^{out}=i)}{bel_0(\omega^{out}=i)} \tag{18}$$

If we assume an equal ratio between falling and normal actions, we can define $bel_0(\omega^{out}=i) = 0.5$ and $bel_0(\omega^{out}\neq i) = 0.5$. The values of $bel_0(\omega^{out}=i)$ and $bel_0(\omega^{out}\neq i)$ can be determined based on the ratio of the occurrence of falling and normal actions. Let us denote the expression $\frac{bel_n(\omega^{out}=i)}{1-bel_n(\omega^{out}=i)}$ in Equation (18) as δ_i^n. Considering N time steps, we can calculate δ_i^N, and consequently, the final value of $bel_N(\omega^{out}=i)$ can be computed using the following equation:

$$bel_N(\omega^{out}=i) = \frac{\delta_i^N}{1+\delta_i^N} \tag{19}$$

Finally, the classification is made based on the individual belief:

$$\omega^{out} = \underset{i=1,2,\ldots,C}{\operatorname{argmax}}\left[bel_N(\omega^{out}=i)\right] \tag{20}$$

4. Experimental Results

4.1. Graph Construction

In this study, we employed ST-GCN [35] as a classifier. To utilize the spatial and temporal connectivity information in ST-GCN, the skeleton data were structured into a graph format and used as the input for ST-GCN. To represent the skeleton data as a graph, each key point was treated as a node, and the connections between the nearest key points linked by bones, such as the knees and ankles, were represented as edges. An example of graph construction is shown in Figure 6.

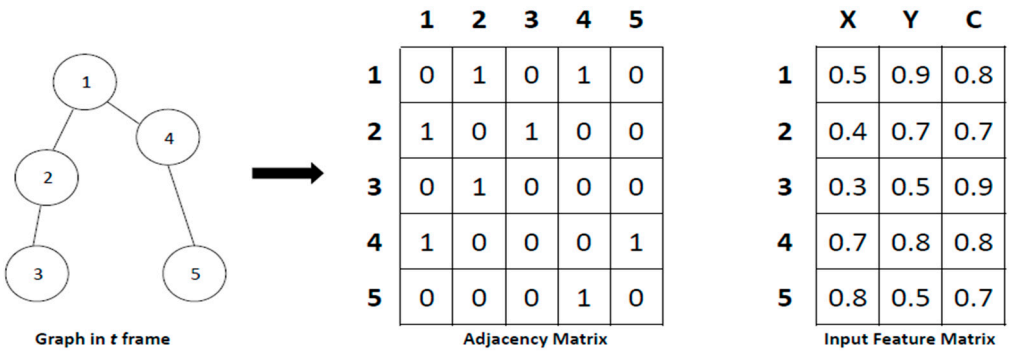


Figure 6. Example of graph construction.

The adjacency matrix represents the connection status between nodes. In Figure 6, nodes 1 and 2 are connected, so the values of the adjacency matrix in the second row and

first column and in the first row and second column are both 1. In the skeleton, if the ankles and knees are connected, the corresponding value in the adjacency matrix becomes 1. The input feature matrix represents the features of the nodes. The features are the x-coordinate, y-coordinate, and confidence score of each node, which are computed in the pose estimation module. In the input feature matrix of Figure 6, we can see that for node 1, the x-coordinate is 0.5, the y-coordinate is 0.9, and the confidence score is 0.8. To simplify graph construction, the example used in Figure 6 illustrates a graph with five nodes. However, we utilized 13 key points, and thus, the graph data were constructed using 13 nodes.

4.2. Dataset and Experimental Setup

The HAR-UP dataset [27] was used to test the proposed system. The HAR-UP dataset consists of 11 categorized actions, each performed by 17 subjects in three trials and captured from two camera viewpoints. Among the 11 actions, 1–5 represent falls, whereas 6–11 depict different actions. In this study, the remaining seven actions, excluding falls, were categorized as normal actions, and the experiments were conducted accordingly. Furthermore, the sixth action in the dataset involves subjects moving out of the camera's field of view and returning, making it challenging to define the region of interest (ROI). Hence, this behavior was excluded from the experiments. A total of 1016 videos were analyzed. Figure 7 shows examples of images from the HAR-UP dataset with fall and normal actions. To facilitate the understanding of the subjects' actions, their behaviors were depicted using a sequence of eight images with a 10-frame interval, as shown in Figure 7.

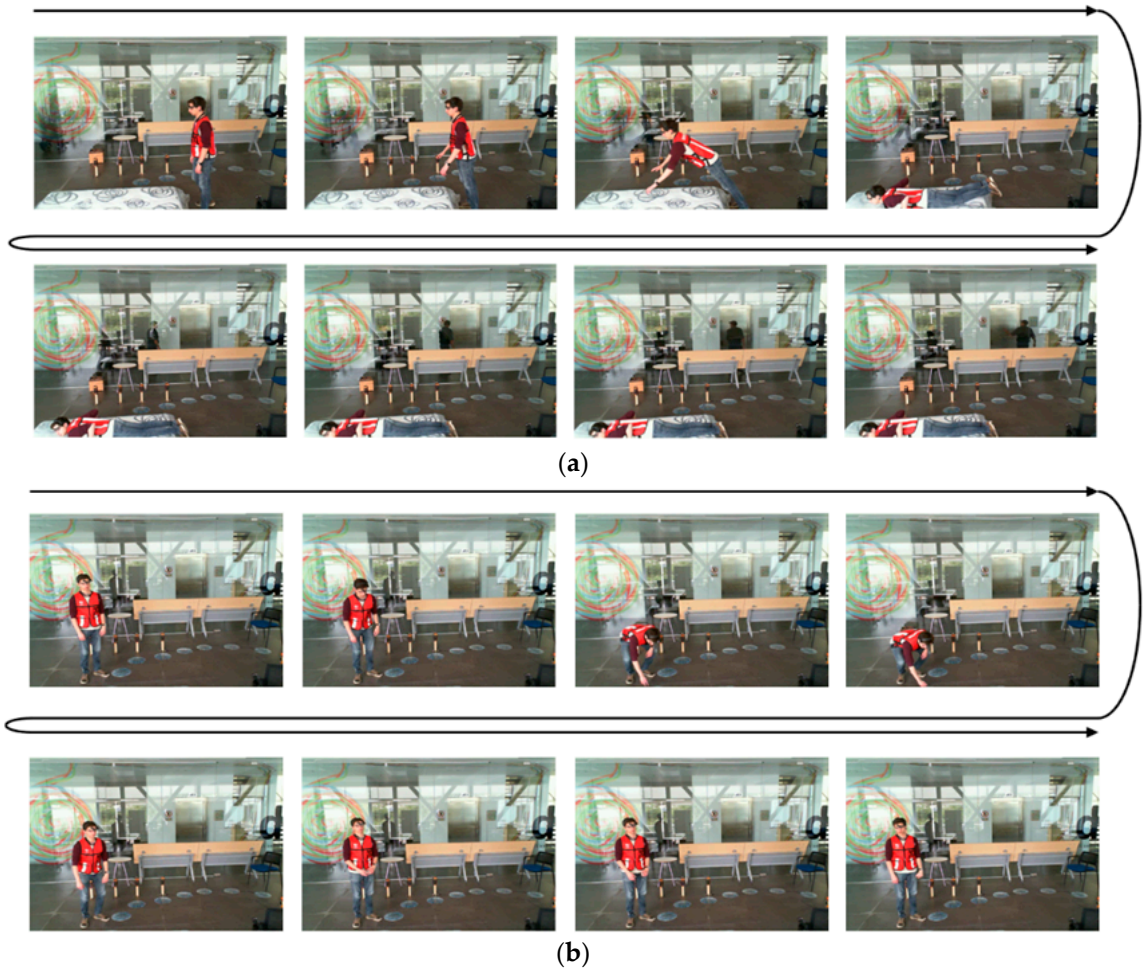


Figure 7. HAR-UP dataset: (**a**) example of fall action, and (**b**) example of normal (picking up) action.

In this work, we employed modules pre-trained on external datasets. For person detection, we utilized YOLO models trained by [28,29] on the MS COCO detection dataset. Additionally, for the pose estimation module, we utilized AlphaPose and ViTPose trained by [30,31] on the MS COCO keypoint dataset. For the classification module, we utilized the trained ST-GCN obtained from [41] using the Le2i Fall Detection dataset. Therefore, the HAR-UP dataset was utilized entirely as test data without being partitioned into separate training and testing datasets. Tables 3 and 4 present the input and output data sizes, data type, and number of parameters of the object detection, pose estimation, and action recognition modules used in this experiment, as well as the specifications of the employed system.

Table 3. Input and output sizes of each module.

Model	Data Type	Parameters	Input	Output
YOLOV3	RGB image	8,669,876	(640, 480)	(1, 7)
YOLOV7	RGB image	36,905,341	(640, 480)	(1, 7)
AlphaPose	RGB image	40,577,745	(256, 192)	(3, 13)
ViTPose	RGB image	89,994,513	(256, 192)	(3, 13)
ST-GCN	Skeleton data	6,144,323	(3, 13, 30)	(1, 2)

Table 4. System configuration.

Type	Main Specification
OS	Ubuntu 20.04LTS
GPU	NVIDIA GeForce RTX 3090
CPU	Intel Core i9 13900KF
RAM	DDR5, 64GB

4.3. Experimental Results

Figures 8 and 9 present the recognition accuracy for fall and normal actions obtained by varying the number of classifiers of the fusion method using the HAR-UP dataset. As a baseline, we considered a scenario where if the ST-GCN result indicated a fall at least once in a video, it was considered a detected fall occurrence.

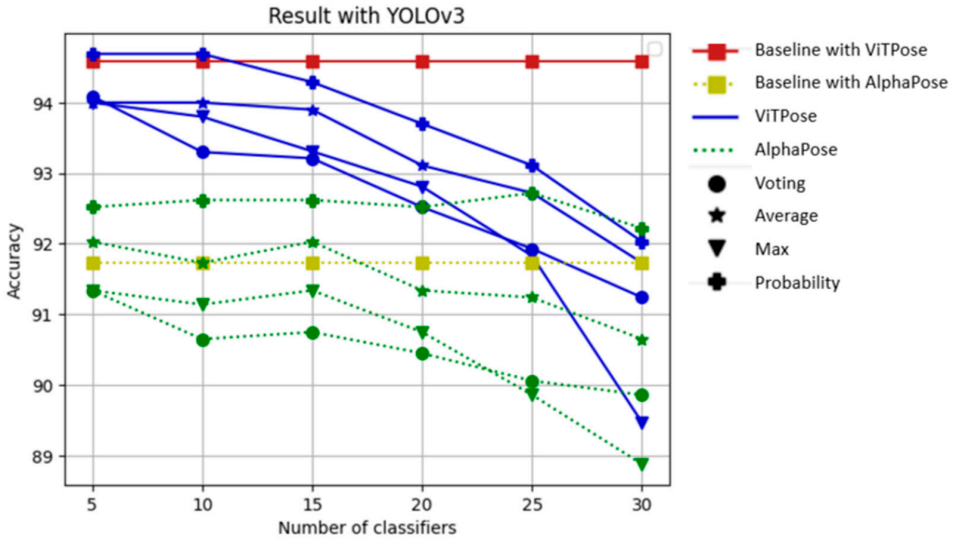


Figure 8. Accuracy graph of each fusion method using YOLOv3.

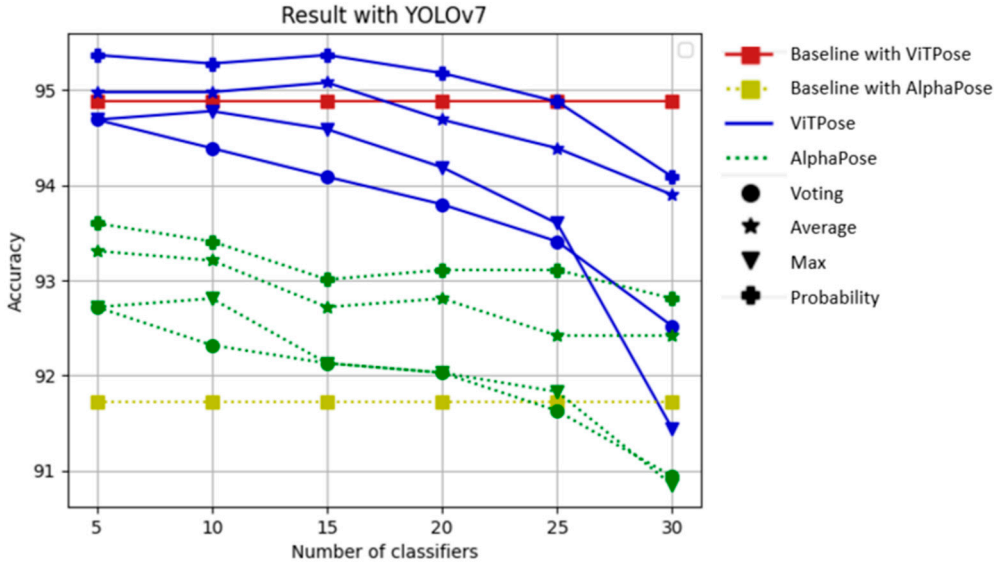


Figure 9. Accuracy graph of each fusion method using YOLOv7.

Figure 8 shows the results obtained using YOLOv3 for person detection, and Figure 9 shows the results obtained using YOLOv7. It can be observed that using both YOLOv3 and YOLOv7, the performance improved compared to that before fusion. With the use of YOLOv3 and YOLOv7, we observed instances where the fusion model exhibited superior performance compared to the baseline model. Specifically, the application of time-level fusion contributed to performance enhancement. When AlphaPose was used, the enhancement in the performance of the fusion model was significantly more substantial compared to the utilization of ViTPose. This signifies the efficacy of the fusion technique in effectively mitigating fluctuations in the skeleton sequence or classifier results arising from certain inaccuracies in AlphaPose estimation. Notably, the performance was consistently higher with the following fusion methods: probabilistic fusion, average, maximum, and voting. When YOLOv3 was used, the combination of ViTPose and probabilistic fusion with the number of classifiers being 10 exhibited the highest performance. However, when YOLOv7 was employed, the best performance was achieved using ViTPose and probabilistic fusion with the number of classifiers being five. YOLOv7 performed better than YOLOv3, and ViTPose performed better than AlphaPose. In general, accuracy tended to decrease when the number of classifiers exceeded 15. The results for all 1016 videos using the optimal numbers of classifiers for each fusion method are presented in Tables 5 and 6.

In a binary classification problem, true (T) and false (F) indicate whether the predicted result is correct or not. Positive (P) and negative (N) indicate cases in which the model classifies the state as falling and normal, respectively. Precision, recall, and accuracy, presented in Tables 5 and 6, are the most widely used indices for data learning of classifiers and are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{21}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{22}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{23}$$

Table 5. Results of fall detection using YOLOv3 with the optimal number of classifiers.

Fusion Method	Optimal Number of Classifiers	Model	Data Label	Precision (%)	Recall (%)	Accuracy (%)
Baseline	-	ViTPose	Fall	97.49	91.57	94.59
			Normal	91.99	97.63	
		AlphaPose	Fall	93.65	89.61	91.73
			Normal	89.96	93.87	
Probability	10	ViTPose	Fall	97.90	91.37	94.69
			Normal	91.85	98.02	
		AlphaPose	Fall	93.94	91.18	92.62
			Normal	91.36	94.07	
Average	5	ViTPose	Fall	97.26	90.59	94.00
			Normal	91.13	97.43	
		AlphaPose	Fall	93.87	90.00	92.03
			Normal	90.32	94.07	
Max	5	ViTPose	Fall	97.46	90.39	94.00
			Normal	90.98	97.63	
		AlphaPose	Fall	93.78	88.63	91.34
			Normal	89.14	94.07	
Voting	5	ViTPose	Fall	98.28	89.80	94.09
			Normal	90.55	98.42	
		AlphaPose	Fall	93.78	88.63	91.34
			Normal	89.14	94.07	

With the utilization of YOLOv3, the optimal precision for detecting fall cases was achieved by combining ViTPose with the voting method. Conversely, for normal cases, the baseline performance with ViTPose demonstrated the highest precision. Regarding recall, the baseline with ViTPose exhibited superior performance for fall cases, while the optimal result for normal cases was obtained by employing ViTPose and the voting method. In the context of YOLOv7 utilization, the highest precision for fall cases was observed with ViTPose and the voting method, whereas for normal cases, the highest precision was attained through the combination of ViTPose and the averaging method. Concerning recall, ViTPose combined with the averaging method resulted in the highest performance for fall cases, while ViTPose combined with the voting method proved to be the most effective strategy for normal cases. Table 7 presents a comparison of the recognition results for each action between ViTPose and AlphaPose when using the probabilistic fusion method and YOLOv7, which is the most effective.

Table 6. Results of fall detection using YOLOv7 with the optimal number of classifiers.

Fusion Method	Optimal Number of Classifiers	Model	Data Label	Precision (%)	Recall (%)	Accuracy (%)
Baseline	-	ViTPose	Fall	96.73	92.94	94.88
			Normal	93.16	96.84	
		AlphaPose	Fall	93.65	89.61	91.73
			Normal	89.96	93.87	

Table 6. *Cont.*

Fusion Method	Optimal Number of Classifiers	Model	Data Label	Precision (%)	Recall (%)	Accuracy (%)
Probability	5	ViTPose	Fall	97.15	93.53	95.37
			Normal	93.71	97.23	
		AlphaPose	Fall	95.69	91.37	93.60
			Normal	91.68	95.85	
Average	15	ViTPose	Fall	91.96	98.12	95.08
			Normal	98.22	92.38	
		AlphaPose	Fall	88.82	96.38	92.72
			Normal	96.64	89.56	
Max	10	ViTPose	Fall	91.76	97.70	94.78
			Normal	97.83	92.18	
		AlphaPose	Fall	89.22	96.19	92.81
			Normal	96.44	89.87	
Voting	5	ViTPose	Fall	97.70	91.57	94.69
			Normal	92.01	97.83	
		AlphaPose	Fall	95.61	89.61	92.72
			Normal	90.15	95.85	

Table 7. Number of action recognition results of the probabilistic fusion method with the optimal number of classifiers.

Model	Action	Actual Value \ Predicted Value	Fall	Normal	Recall (%)
ViTPose	1	Fall	98	4	96.08
AlphaPose			101	1	99.02
ViTPose	2	Fall	88	14	86.27
AlphaPose			81	21	79.41
ViTPose	3	Fall	96	6	94.12
AlphaPose			93	9	91.18
ViTPose	4	Fall	96	6	94.12
AlphaPose			92	10	90.20
ViTPose	5	Fall	99	3	97.06
AlphaPose			99	3	97.06
ViTPose	7	Normal (Standing)	0	102	100
AlphaPose			0	102	100
ViTPose	8	Normal (Sitting)	0	102	100
AlphaPose			0	102	100
ViTPose	9	Normal (Picking up)	4	98	96.08
AlphaPose			5	97	95.10
ViTPose	10	Normal (Jumping)	0	102	100
AlphaPose			0	102	100

Table 7. *Cont.*

Model	Action	Actual Value \ Predicted Value	Fall	Normal	Recall (%)
ViTPose	11	Normal (Lying down)	10	88	89.80
AlphaPose			16	82	83.67

In the experiments, the misclassification of falling incidents as normal situations mainly occurred due to failures in person detection. To address this issue, we constructed an additional training dataset with a focus on detecting individuals who were either obscured or in a fallen state, utilizing the HAR-UP dataset, and conducted subsequent experiments. During this process, the training dataset and the test dataset were divided in a 7:3 ratio, and 4375 images were selected from the training videos for training YOLOv7. Figure 10 shows the results obtained using the improved YOLOv7.

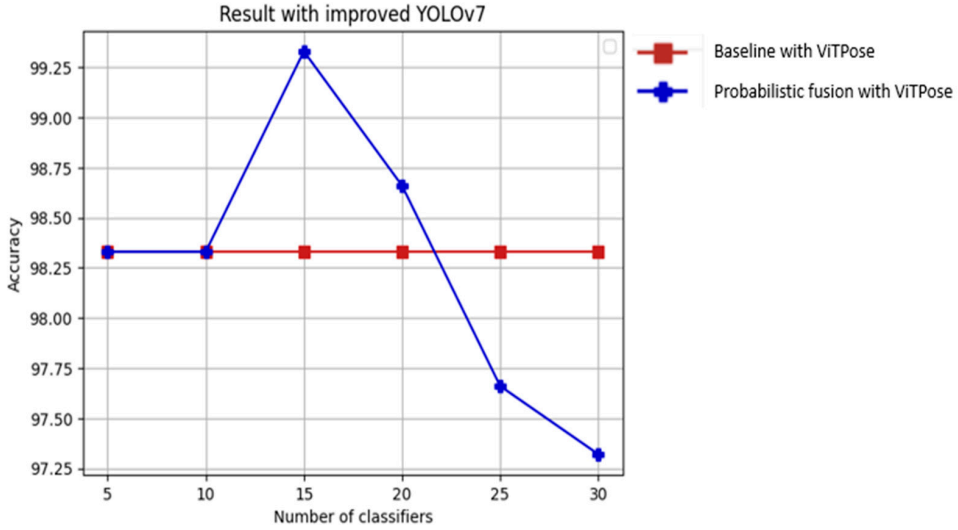


Figure 10. Accuracy graph of the proposed method using improved YOLOv7.

In Table 8, the performance of the proposed method is compared with that of previous methods using the HAR-UP dataset. The performance of these previous methods is directly cited from other research studies [18,20,27,42,43].

Table 8. Comparison of several algorithms using the HAR-UP dataset.

Author	Classifier	Accuracy (%)
Martínez-Villaseñor, L., 2019 [27]	CNN	95.10
Ramirez, H., 2021 [18]	RF	99.34
Galvão, Y. M., 2021 [20]	Encoder–decoder	98.62
Yadav, S. K., 2022 [42]	ARFDNet	96.70
Raza, A., 2023 [43]	ViT	97.36
Proposed method	ST-GCN with probabilistic fusion	99.33

5. Conclusions

In this study, we utilized deep learning-based computer vision techniques to quickly address accidents caused by falls. To enhance the fall detection performance, we employed the aggregation of classifier result sequences, thereby overcoming scenarios that are challenging for the model to classify. Consequently, we were able to improve the fall detection performance by effectively using a light fusion module. In this study, we used various fu-

sion methods, including voting, maximum, averaging, and probabilistic fusion. Compared to the other fusion methods, the probabilistic fusion method showed improvements in precision, recall, and accuracy. In our fall detection system, the accuracy of person detection and pose estimation significantly affects performance because the detection of falls is based on skeleton data. It was observed that using YOLOv7 for person detection and ViTPose led to an improved fall detection performance.

In the experiments, false negatives that classified falls as normal were primarily attributed to failures in person detection. Among these, false negatives occurred particularly when the entire form of a person was not visible in the video or when a person in a fallen state was not detected. Therefore, efforts to improve person detection are crucial to enhance the performance of the proposed system. The proposed fall detection system allows the easy replacement of its modules, making it possible to incorporate state-of-the-art object detection models that outperform YOLOv7. Alternatively, constructing additional training datasets that focus on detecting obscured persons or persons in a fallen state and training the detection model on these datasets could lead to an improvement in person detection performance. Not only the detection module but also the classifier module is replaceable. Therefore, it is worth considering modifying models that have achieved state-of-the-art performance in action recognition for the fall detection task. These are expected to significantly enhance the overall performance of the proposed system. Additionally, conducting experiments on various datasets [44,45] would be beneficial to demonstrate the general effectiveness of the proposed fusion method in enhancing fall detection performance.

Author Contributions: Conceptualization, J.K., B.K. and H.L.; methodology, J.K., B.K. and H.L.; software, J.K.; validation, J.K., B.K. and H.L.; formal analysis, B.K. and H.L.; investigation, J.K.; writing—original draft preparation, J.K.; writing—review and editing, B.K.; visualization, J.K.; supervision, H.L.; project administration, H.L.; funding acquisition, B.K. and H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP; Ministry of Science, ICT & Future Planning) (NRF-2021R1F1A1052074), and this work was also supported by another National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2022-00166346).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: HAR-UP dataset is available at https://github.com/jpnm561/HAR-UP (accessed on 9 January 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Seydi, S.T.; Saeidi, V.; Kalantar, B.; Ueda, N.; Halin, A.A. Fire-Net: A Deep Learning Framework for Active Forest Fire Detection. *J. Sens.* **2022**, *2022*, 8044390. [CrossRef]
2. Xie, Y.; Zhu, J.; Cao, Y.; Zhang, Y.; Feng, D.; Zhang, Y.; Chen, M. Efficient video fire detection exploiting motion-flicker-based dynamic features and deep static features. *IEEE Access* **2020**, *8*, 81904–81917. [CrossRef]
3. Zhang, Y. Safety management of civil engineering construction based on artificial intelligence and machine vision technology. *Adv. Civ. Eng.* **2021**, *2021*, 1–14. [CrossRef]
4. Degas, A.; Islam, M.R.; Hurter, C.; Barua, S.; Rahman, H.; Poudel, M.; Ruscio, D.; Ahmed, M.; Begum, S.; Rahman, M.; et al. A survey on artificial intelligence (ai) and explainable ai in air traffic management: Current trends and development with future research trajectory. *Appl. Sci.* **2022**, *12*, 1295. [CrossRef]
5. *Industrial Accident Analysis Booklet*; Ministry of Employment and Labor of South Korea: Sejong, Republic of Korea, 2021; p. 12.
6. Shany, T.; Redmond, S.J.; Narayanan, M.R.; Lovell, N.H. Sensors-based wearable systems for monitoring of human movement and falls. *IEEE Sens. J.* **2011**, *12*, 658–670. [CrossRef]
7. Lindemann, U.; Hock, A.; Stuber, M.; Keck, W.; Becker, C. Evaluation of a fall detector based on accelerometers: A pilot study. *Med. Biol. Eng. Comput.* **2005**, *43*, 548–551. [CrossRef]

8. Gibson, R.M.; Amira, A.; Ramzan, N.; Casaseca-de-la-Higuera, P.; Pervez, Z. Multiple comparator classifier framework for accelerometer-based fall detection and diagnostic. *Appl. Soft Comput.* **2016**, *39*, 94–103. [CrossRef]
9. Jeong, S.S.; Kim, N.H.; Yu, Y.S. Fall Detection System Based on Simple Threshold Method and Long Short-Term Memory: Comparison with Hidden Markov Model and Extraction of Optimal Parameters. *Appl. Sci.* **2022**, *12*, 11031. [CrossRef]
10. Cippitelli, E.; Fioranelli, F.; Gambi, E.; Spinsante, S. Radar and RGB-depth sensors for fall detection: A review. *IEEE Sens. J.* **2017**, *17*, 3585–3604. [CrossRef]
11. Garripoli, C.; Mercuri, M.; Karsmakers, P.; Soh, P.J.; Crupi, G.; Vandenbosch, G.A.; Pace, C.; Leroux, P.; Schreurs, D. Embedded DSP-based telehealth radar system for remote in-door fall detection. *IEEE J. Biomed. Health Inform.* **2014**, *19*, 92–101. [CrossRef]
12. Wang, B.; Guo, Y. Soft fall detection using frequency modulated continuous wave radar and regional power burst curve. In Proceedings of the 2022 Asia-Pacific Microwave Conference (APMC), Yokohama, Japan, 29 November–2 December 2022; pp. 240–242.
13. Takabatake, W.; Yamamoto, K.; Toyoda, K.; Ohtsuki, T.; Shibata, Y.; Nagate, A. FMCW radar-based anomaly detection in toilet by supervised machine learning classifier. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6.
14. Cippitelli, E.; Gasparrini, S.; Gambi, E.; Spinsante, S. A human activity recognition system using skeleton data from RGBD sensors. *Comput. Intell. Neurosci.* **2016**, *2016*, 4351435. [CrossRef]
15. Panahi, L.; Ghods, V. Human fall detection using machine vision techniques on RGB–D images. *Biomed. Signal Process. Control* **2018**, *44*, 146–153. [CrossRef]
16. Keskes, O.; Noumeir, R. Vision-based fall detection using st-gcn. *IEEE Access* **2021**, *9*, 28224–28236. [CrossRef]
17. Lie, W.N.; Le, A.T.; Lin, G.H. Human fall-down event detection based on 2D skeletons and deep learning approach. In Proceedings of the 2018 International Workshop on Advanced Image Technology (IWAIT), Chiang Mai, Thailand, 7–9 January 2018; pp. 1–4.
18. Ramirez, H.; Velastin, S.A.; Meza, I.; Fabregas, E.; Makris, D.; Farias, G. Fall detection and activity recognition using human skeleton features. *IEEE Access* **2021**, *9*, 33532–33542. [CrossRef]
19. Kim, J.; Lee, J.H.; Lee, H. Fall down detection using vision transformer and graph convolutional network. *J. Korean Soc. Railw.* **2023**, *26*, 251–259. [CrossRef]
20. Galvão, Y.M.; Portela, L.; Ferreira, J.; Barros, P.; Fagundes, O.A.D.A.; Fernandes, B.J. A framework for anomaly identification applied on fall detection. *IEEE Access* **2021**, *9*, 77264–77274. [CrossRef]
21. Alanazi, T.; Muhammad, G. Human fall detection using 3D multi-stream convolutional neural networks with fusion. *Diagnostics* **2022**, *12*, 3060. [CrossRef]
22. Alanazi, T.; Babutain, K.; Muhammad, G. A Robust and Automated Vision-Based Human Fall Detection System Using 3D Multi-Stream CNNs with an Image Fusion Technique. *Appl. Sci.* **2023**, *13*, 6916. [CrossRef]
23. Lara, O.D.; Pérez, A.J.; Labrador, M.A.; Posada, J.D. Centinela: A human activity recognition system based on acceleration and vital sign data. *Pervasive Mob. Comput.* **2012**, *8*, 717–729. [CrossRef]
24. Li, H.; Shrestha, A.; Heidari, H.; Le Kernec, J.; Fioranelli, F. Bi-LSTM network for multimodal continuous human activity recognition and fall detection. *IEEE Sens. J.* **2019**, *20*, 1191–1201. [CrossRef]
25. Chahyati, D.; Hawari, R. Fall detection on multimodal dataset using convolutional neural network and long short term memory. In Proceedings of the 2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Depok, Indonesia, 17–18 October 2020; pp. 371–376.
26. Wang, X.; Penta, R.; Sehgal, B.; Chen-Song, D. Human fall detection-multimodality approach. *arXiv* **2023**, arXiv:2302.00224.
27. Martínez-Villaseñor, L.; Ponce, H.; Brieva, J.; Moya-Albor, E.; Núñez-Martínez, J.; Peñafort-Asturiano, C. UP-fall detection dataset: A multimodal approach. *Sensors* **2019**, *19*, 1988. [CrossRef]
28. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
29. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 7464–7475.
30. Fang, H.S.; Xie, S.; Tai, Y.W.; Lu, C. Rmpe: Regional multi-person pose estimation. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2334–2343.
31. Xu, Y.; Zhang, J.; Zhang, Q.; Tao, D. Vitpose: Simple vision transformer baselines for human pose estimation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 38571–38584.
32. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
33. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 21–26 June 2016; pp. 779–788.
34. Vaswani, A.; Shazeer, N.M.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing System, Long Beach, CA, USA, 4–9 December 2017; pp. 6000–6010.
35. Yan, S.; Xiong, Y.; Lin, D. Spatial temporal graph convolutional networks for skeleton-based action recognition. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LO, USA, 2–7 February 2018.

36. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
37. Hu, J.; Shen, L.; Sun, G. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 24–27 June 2018; pp. 7132–7141.
38. Yao, L.; Miller, J. Tiny imagenet classification with convolutional neural networks. *CS 231n* **2015**, *2*, 8.
39. Lee, H.; Hong, S.; Kim, E. Neural network ensemble with probabilistic fusion and its application to gait recognition. *Neurocomputing* **2009**, *72*, 1557–1564. [CrossRef]
40. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics*; MIT Press: Cambridge, MA, USA, 2005; p. 28.
41. Human-Falling-Detect-Tracks. Available online: https://github.com/GajuuzZ/Human-Falling-Detect-Tracks (accessed on 1 December 2023).
42. Yadav, S.K.; Luthra, A.; Tiwari, K.; Pandey, H.M.; Akbar, S.A. ARFDNet: An efficient activity recognition & fall detection system using latent feature pooling. *Knowl. Based Syst.* **2022**, *239*, 107948.
43. Raza, A.; Yousaf, M.H.; Velastin, S.A.; Viriri, S. Human fall detection from sequences of skeleton features using vision transformer. In Proceedings of the 18th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, Lisbon, Porutugal, 8–10 February 2023; pp. 591–598.
44. Kwolek, B.; Kepski, M. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Comput. Methods Programs Biomed.* **2014**, *117*, 489–501. [CrossRef]
45. Alzahrani, M.S.; Jarraya, S.K.; Salamah, M.A.; Ben-Abdallah, H. FallFree: Multiple fall scenario dataset of cane users for monitoring applications using kinect. In Proceedings of the International Conference on Signal-Image Technology & Internet-Based Systems, Jaipur, India, 4–7 December 2017; pp. 327–333.