Sensors and Materials, Vol. 37, No. 4 (2025) 1657–1679
MYU Tokyo

S & M 4011

Comparative Analysis of Lightweight OpenPose and MoveNet AI Models for Real-time Fall Detection and Alert Systems

Yuh-Shihng Chang,* Yi-Xiang Zheng, and Zheng-Yu Ku

Department of Information Management, National Chin-Yi University of Technology,
No. 57, Sec. 2, Zhongshan Rd., Taiping Dist., Taichung 411030, Taiwan (R.O.C.)

(Received October 28, 2024; accepted April 4, 2025)

Keywords: AI, human pose estimation, fall detection, human activity recognition and behavior understanding

Falls are the leading cause of injury-related deaths among adults aged 65 and older. The age-adjusted fall mortality rate increased by 41%, from 0.0553% in 2012 to 0.0780% in 2021. The models OpenPose and MoveNet can be used to detect human movements associated with falls in dynamic images. In this study, we utilized these AI models to independently identify falls among elderly individuals living alone and compared the efficiency of these two AI fall detection models in capturing dynamic images. In this study, we employed a web camera as a sensor and integrated it with the two systems mentioned above to detect human limb movements, determine whether the monitored individual has fallen, and send emergency notifications to family members, caregivers, and nursing staff via communication software. AI-based posture detection technology is vital for elderly individuals who live alone, have poor health, or have limited mobility. Our evaluation of the detection efficiency of different AI fall detection models provides valuable references for applications in healthcare systems.

1. Introduction

Falls are the leading cause of injury-related death among adults aged 65 and older. The age-adjusted fall death rate increased by 41%, from 0.0553% of older adults in 2012 to 0.0780% in 2021.(1,2) According to the definition by the World Health Organization, a fall refers to an event where a person unintentionally comes to rest on the ground or floor.(3) Following a fall, aside from potentially causing fatal or nonfatal injuries, it can also have negative psychological effects on the individual and their family members. For instance, the person who fell may experience anxiety about falling again, along with feelings of guilt and self-blame. These emotions can impact their daily life, potentially leading to reduced outdoor activities and, ultimately, a gradual loss of physical function and mobility. When a fall results in severe injury and there is no immediate means to notify others or receive emergency assistance, it can lead to irreversible harm and loss of independence, necessitating long-term reliance on others for assistance and care. This situation not only affects the quality of life of the individual but also

*Corresponding author: e-mail: eric_chang@ncut.edu.tw
https://doi.org/10.18494/SAM5420

imposes additional burdens on their family and society. To mitigate the potential harm caused by falls, developing a real-time fall detection and alert system is crucial. Such a system not only aids in promptly responding to fall incidents and reducing the risk of harm but also offers potential psychological benefits. It enables individuals to receive timely support and care when they experience a fall, thereby maintaining their physical and mental well-being as well as their overall quality of life.

In this study, we argue for the necessity of a reliable fall detection system that can immediately notify family members or healthcare providers upon detecting a fall event. Currently, human motion detection technologies primarily fall into two categories: wearable sensors and image recognition techniques. However, wearable devices may pose inconvenience for older adults or young children who may not be familiar with how to use them or may easily forget to wear them. Particularly for some elderly individuals, they may find it challenging to adapt to the constant requirement of wearing sensors, not to mention young children who may not understand their purpose and could inadvertently damage the wearable devices.

Therefore, we chose to use lightweight pose estimation models, OpenPose(4) and MoveNet,(5) with a focus on fast, real-time motion estimation. These models can directly output human keypoint positions and, in some applications, may be paired with specialized motion classifiers [such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs)] to classify specific actions, enabling the real-time detection of falls and utilizing the LINE Notify communication software for message notifications. We aim to achieve the following objectives: (1) establish a real-time fall detection and notification system, (2) successfully identify current actions of individuals, and (3) promptly notify family members or healthcare providers when a fall occurs, thereby achieving immediate alerting effects. Compared with IoT sensing technology, AI image recognition technology can provide a convenient, precise, and non-intrusive solution without the need for additional wearable devices, a fact supported by multiple studies.(6–9) This further ensures timely detection and notification of fall events, ultimately enhancing the safety and quality of life for individuals. In this paper, we focus on machine learning techniques, specifically the lightweight OpenPose and MoveNet AI models, using a web cam as a sensor for real-time human pose estimation and fall detection. We leverage AI techniques, including deep learning, image recognition sensing, and posture estimation perception, to support critical healthcare applications. The aforementioned sensors and AI technologies are integrated with systems such as LINE Notify to provide instant alerts in emergency situations, emphasizing their practical and social importance.

2. Theoretical Foundation

In this section, the image recognition technology of AI open sources and human motion classification methods will be discussed. This discussion aims to facilitate a better understanding of the Lightweight OpenPose(4) and MoveNet(5) models proposed in this study and the human motion classification model. These are crucial for real-time detection capabilities, as the aforementioned AI algorithms and models help ensure that fall events can immediately trigger notification processes, providing prompt responses and reducing potential harm.

2.1 Human posture recognition

Image recognition technology is a rapidly growing field in recent years, which is based on computer vision and machine learning, utilizing deep learning models to achieve automatic recognition and classification of images. This technology finds extensive applications in areas such as security surveillance,(10,11) medical image analysis,(12–14) and autonomous driving.(15,16) From facial recognition to traffic sign detection, it demonstrates its importance in enhancing security, medical diagnosis, and traffic systems. Human pose recognition is also a critical aspect of this technology, further expanding its application domains, significantly impacting quality of life and technological advancement, and playing a crucial role in technological development.

The method chosen for this study involves using human pose estimation within image recognition technology to effectively detect human movements. Human pose estimation can be divided into two main approaches: top-down and bottom-up. In the top-down approach, the image first detects potential locations containing humans and labels bounding boxes around them. Then, it identifies the human joint points within these bounding boxes. Models representing this approach include AlphaPose(17) and Hourglass.(18) The bottom-up approach, on the other hand, identifies the joint positions of each person in the image and constructs complete human poses on the basis of the relationships between these joints. Models representing this approach include PifPaf,(19) OpenPose,(20) Lightweight OpenPose,(4) and MoveNet.(5)

In this study, considering the need for real-time action detection, we adopted a bottom-up human pose recognition approach, specifically utilizing the Lightweight OpenPose model and MoveNet models. Given the potentially large quantity of image datasets, which could otherwise decrease detection speeds, the bottom-up approach involves identifying all joint points in the image at once, without imposing additional computational burden as the number of individuals increases. In contrast, a top-down approach would increase joint detection computations with more people in the scene, potentially affecting detection efficiency in multiperson scenarios. The advantages of the Lightweight OpenPose(4) and MoveNet(5) models lie in their lightweight design, maintaining accuracy while reducing computational load, which is suitable for real-time action detection applications. Through this method, we can more accurately capture and analyze human movements in images, providing a reliable foundation for real-time fall detection and notification systems.

The overall process of OpenPose(20) is shown in Fig. 1, which includes taking the whole image as input, co-predicting the confidence maps for body part detection and part affinity fields for part association via CNNs, and then performing a series of bipartite matches to associate body part candidates, which are finally assembled into a complete body pose for all the people in the image. OpenPose(20) was developed by the Computer Vision Center team, consisting of researchers from the Technical University of Catalonia.

In this study, we considered that the Lightweight OpenPose model, improved by Osokin,(4) is a powerful pose estimation model capable of accurately identifying multiple human joint points, making it suitable for handling complex postures. This is achieved through a highly optimized network design and postprocessing code. It utilizes a dilated MobileNet V1(21) feature extractor with depthwise separable convolutions, along with a lightweight refinement stage design, resulting in an increase in the accuracy-to-network complexity ratio of over 6.5 times.

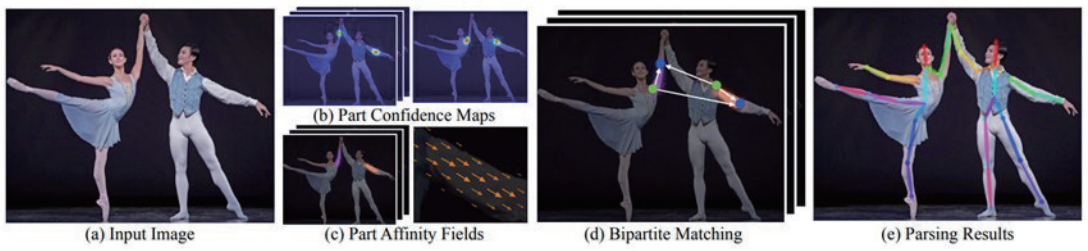


Fig. 1. (Color online) Overall process of OpenPose.(20)

To assess the efficiency of AI models in detecting body movements, we also employed another lightweight model, MoveNet, for a comparative evaluation of its performance in real-time fall detection scenarios. The architecture of MoveNet uses the CenterNet(22) method as its underlying technology. CenterNet is a bottom-up method whose main idea is to transform the problem of target detection into predicting the centroid of each target in the image and simultaneously predicting the target's bounding box. This method simplifies the process of target detection and improves accuracy and efficiency.

MoveNet has a total of four models, mainly divided into two variants, MoveNet-SinglePose and MoveNet-MultiPose, which are suitable for single-person and multiperson pose estimations, respectively. MoveNet-SinglePose includes three models: PoseNet,(23) MoveNet-Lightning, and MoveNet-Thunder. PoseNet is a pose estimation model developed by Google, which is mainly based on the MobileNet architecture. Although PoseNet and MoveNet are different pose estimation models, they are both developed by Google and built and deployed on the basis of the TensorFlow framework, so they are often discussed and compared together.

The overall flow of MoveNet is shown in Fig. 2. In the first step, the centroid of the individual is identified using the body-centered heatmap, which is the arithmetic mean of all the keypoints, and the point with the smallest distance from the center of the frame is selected. In the second step, an initial set of key points for the body is generated from the key point regression output. In the third step, we multiply each key point heat map pixel by a weight that is inversely proportional to the distance of the regressed key points to filter out interference from the background figure. In the fourth step, we select the point with the largest heat map value from each key point channel and add a 2D offset to accurately predict the key point.

MoveNet-Lightning focuses on providing low latency and fast inference, whereas MoveNet-Thunder emphasizes accuracy for complex scenarios. Although the Lightning model is faster in inferring, it is slightly inferior to Thunder in terms of accuracy. As shown in Table 1, these three models are tested on a subset of the Common Objects in Context (COCO) dataset to evaluate their accuracy. In the COCO dataset, each image is filtered and cropped to ensure that only one person's pose is included.(24)

The application scenario of this study is focused on elderly individuals living alone. Therefore, a comparison was made between the MoveNet-Lightning and Lightweight OpenPose models to evaluate whether they meet the requirements of real-time performance in detecting fall postures during elderly activities under low-computation conditions, focusing on efficiency and low latency.

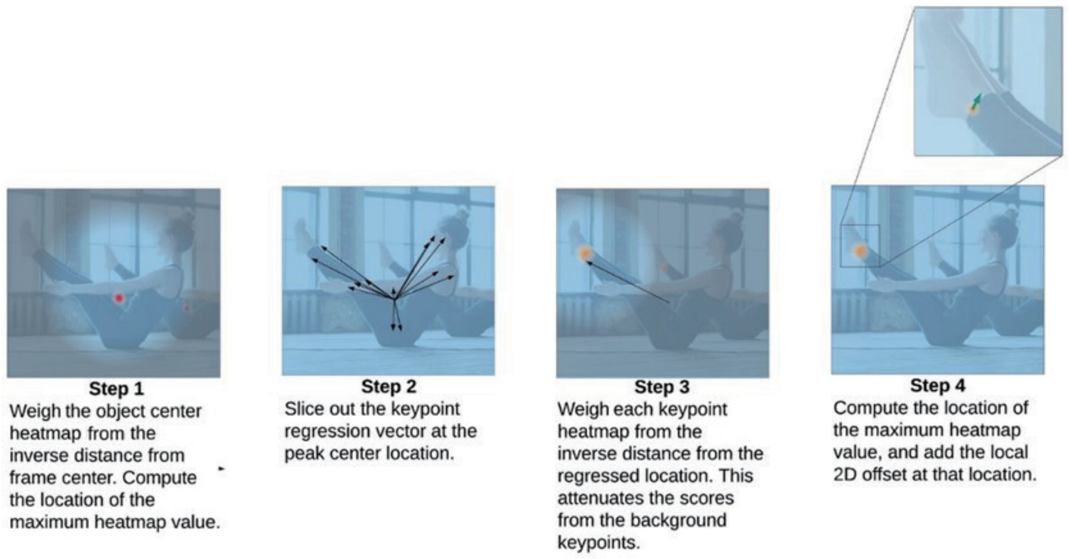

Step 1
Weigh the object center heatmap from the inverse distance from frame center. Compute the location of the maximum heatmap value.
Step 2
Slice out the keypoint regression vector at the peak center location.
Step 3
Weigh each keypoint heatmap from the inverse distance from the regressed location. This attenuates the scores from the background keypoints.
Step 4
Compute the location of the maximum heatmap value, and add the local 2D offset at that location.

Fig. 2. (Color online) Process of MoveNet.[24]

Table 1
Comparison of accuracies of PoseNet, MoveNet-Lightning, and MoveNet-Thunder on COCO dataset.

Model	Size (MB)	mAP	Latency (ms)		
			Pixel 5 - CPU 4 threads	Pixel 5 - GPU	Raspberry Pi 4 - CPU 4 threads
MoveNet Thunder (FP16 quantized)	12.6	72.0	155	45	594
MoveNet Thunder (INT8 quantized)	7.1	68.9	100	52	251
MoveNet Lightning (FP16 quantized)	4.8	63.0	60	25	186
MoveNet Lightning (INT8 quantized)	2.9	57.4	52	28	95
PoseNet (MobileNetV1 backbone FB32)	13.3	45.6	80	40	338

2.2 Classification of human body movements

Common methods for classifying human body movements include using CNNs,[25] RNNs,[26] and Multilayer Perceptron (MLP).[27] CNNs are widely used for extracting features from static postures, whereas RNNs excel in capturing dynamic time-series aspects. Some studies further incorporate attention mechanisms to emphasize key body parts, enhancing the model's capability to recognize complex movements. MLP models, as basic neural network structures, demonstrate their utility in learning complex nonlinear relationships. Overall, these methods provide a solid foundation for human motion detection and are widely applied in areas such as visual perception and behavior analysis. The MLP mixer developed on the basis of MLP features an architecture that includes two types of MLP: channel-mixing MLP and token-mixing MLP. Each MLP block contains two fully connected layers and a nonlinearity applied independently to each row of its input data tensor. Mixer layers can be written as (omitting layer indices)

$$\begin{aligned} U_{*,\,i} &= X_{*,\,i} + W_2\sigma(W_1 LayerNorm(X)_{*,\,i}), \text{ for } i = 1 \ldots C, \\ Y_{j,*} &= U_{j,*} + W_4\sigma(W_3 LayerNorm(U)_{j,*}), \text{ for } j = 1 \ldots S. \end{aligned} \tag{1}$$

As mentioned above, the overall complexity is linear in the number of pixels in the image, as for a typical CNN.(27) The same channel-mixing MLP (token-mixing MLP) is applied to every row (column) of X. Tying the parameters of the channel-mixing MLP (within each layer) is a natural choice—it provides positional invariance, a prominent feature of convolutions.

The MLP model typically consists of multiple fully connected layers,(28) where each neuron in these layers is connected to every neuron in the previous layer. This structure helps MLP capture relationships between input features. Additionally, each fully connected layer in an MLP typically includes a nonlinear activation function such as the rectified linear unit (ReLU) or sigmoid,(29) which aids the model in distinguishing between different classes. In MLP models, the Gaussian error linear unit (GELU) nonlinear activation function and the ReLU nonlinear activation function exhibit significant differences:

- ReLU: The ReLU activation function is defined as $f(x) = max(0, x)$, which outputs zero when the input is less than or equal to zero, and otherwise remains unchanged. ReLU is piecewise linear and discontinuous at $x = 0$ (with a discontinuous derivative).
- GELU: The GELU activation function is a smooth nonlinear function defined as $f(x) = x \cdot \Phi(x)$, where $\Phi(x)$ is the Gaussian cumulative distribution function. GELU is smooth and differentiable across the entire real number range, making it popular in various applications, particularly in neural networks.

The GELU function can approximate the ReLU function in specific cases, particularly when the input values are very large or very small. This approximation property sometimes offers additional advantages. Generally, GELU may converge slightly faster than ReLU during training owing to its smooth nature, which aids in faster gradient propagation. Finally, MLP often uses the Softmax function(30) to generate a probability distribution over classes. This means that the model informs us of the likelihood that the input data belongs to each class. During training, MLP uses the backpropagation algorithm(31) and optimizers to adjust their settings and improve classification accuracy.

This classifier model can be represented by a mathematical formula. Assuming that the input feature vector is x, the computation process of the model is as follows:

1. First Fully Connected Layer: The input x undergoes linear transformation in the first fully connected layer: $h^{(1)} = W^{(1)}x + b^{(1)}$, where $W^{(1)}$ is the weight matrix of the first layer and $b^{(1)}$ is the bias vector.
2. ReLU Nonlinear Activation Function: The output $h^{(1)}$ from the first layer is passed through the ReLU activation function: $h^{(2)} = \max(0, h^{(1)})$, where max performs element-wise comparison and takes the maximum.

3. Second Fully Connected Layer: The ReLU activated output $h^{(2)}$ is linearly transformed in the second fully connected layer: $h^{(3)} = W^{(2)}h^{(2)} + b^{(2)}$, where $W^{(2)}$ is the weight matrix of the second layer and $b^{(2)}$ is the bias vector.
4. Softmax Function: Finally, the output $h^{(3)}$ from the second layer is passed through the Softmax function to obtain the class probabilities: $\hat{y}$ = Softmax($h^{(3)}$) represents the model's output, indicating the probability distribution over classes.

In summary, the mathematical formula for this classifier model is

$$\hat{y} = Softmax\left(W^{(2)}max\left(0,\, W^{(1)}x + b^{(1)}\right) + b^{(2)}\right). \tag{2}$$

Here, $W^{(1)}$ and $b^{(1)}$are the weights and biases of the first layer, and $W^{(2)}$ and $b^{(2)}$ are the weights and biases of the second layer, respectively, and Softmax() denotes the Softmax function. The input layer has 16384 features, representing the feature size of a single input sample, with each input sample being a grayscale image of size 128 × 128. Initially, through the first fully connected layer, linear (16384, 100), input features are transformed into a 100-dimensional hidden representation, followed by a ReLU nonlinear activation function to enhance the model's learning and adaptation to the data. The second fully connected layer, linear (100, 2), then transforms the 100-dimensional hidden representation into an output with two classes. Finally, the Softmax function provides a probability distribution over each class. This structure is designed for classification tasks where the model aims to predict which class a given input sample belongs to. The architecture of the MLP model is illustrated in Fig. 3.

The classifier model used by MoveNet is also an MLP structure, which consists of two fully connected layers, two ReLU nonlinear startup functions, two dropout layers, and a Softmax function. The architecture is shown in Table 2.

The first layer is the input layer, which receives a one-dimensional vector of length 51 corresponding to 17 human body keypoints. Each keypoint consists of three values: x-coordinate, y-coordinate, and confidence score. The data is then processed by the second layer, a custom embedding layer, which handles the x and y coordinates before passing them to the first fully connected layer (Layer 3) with 128 neurons. This layer uses the ReLU6 activation function to

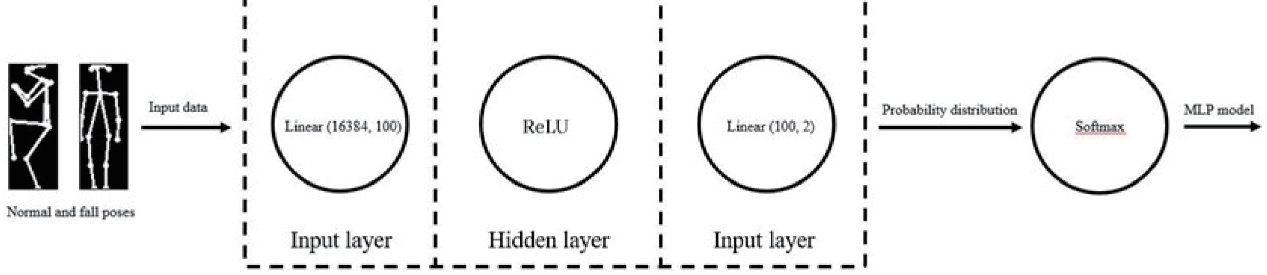


Fig. 3. (Color online) MLP architecture.

Table 2
Architecture of MoveNet.

Layer	Type	Input shape	Activation function
1	Input layer	(51,)	–
2	Custom layer	(51,)	–
3	Dense layer	(51,)	ReLU6
4	Dropout layer	(128,)	–
5	Dense layer	(128,)	ReLU6
6	Dropout layer	(64,)	–
7	Dense layer	(64,)	Softmax

capture nonlinear features and is followed by a dropout layer (Layer 4) that randomly drops 50% of the neurons to reduce the risk of overfitting. The second fully connected layer (Layer 5) has 64 neurons and also utilizes the ReLU6 activation function. It is followed by another dropout layer (Layer 6) to further enhance the model's generalization capability. Finally, the output layer (Layer 7) consists of len neurons (in this study, for binary classification, it is 2) and employs the Softmax activation function for multiclass classification.

3. System Construction Planning

3.1 System development

In terms of system setup, we have installed a video camera for real-time human pose detection. When the camera detects the presence of a person in the frame, the relevant image data is immediately transmitted to the system. The system not only receives the data but also actively performs action recognition to ensure real-time motion analysis of the person in the frame. When the system detects that a person has fallen, it promptly activates the notification procedure. Through the LINE Notify service, the system can instantly send notification messages, enhancing the response speed to fall incidents and further reducing the potential for harm. The overall design and implementation of the system architecture are illustrated in Fig. 4. With this system configuration, we ensure the operation of real-time detection and notification processes.

When the system's video camera detects a person in the frame, it first utilizes the Lightweight OpenPose and MoveNet models to perform joint detection on the person's body, displaying their joint posture and detection box in the frame. Through this process, the system can immediately capture the motion characteristics of the human body, providing detailed information for subsequent analysis.

Next, the system uses an MLP model to recognize the current actions of the detected person. Actions are classified into two categories: "normal" (normal actions) and "fall" (falling actions). If the system detects a fall for 3 s or longer, it promptly activates the LINE Notify service to send immediate notifications to family members or medical personnel. This design aims to implement

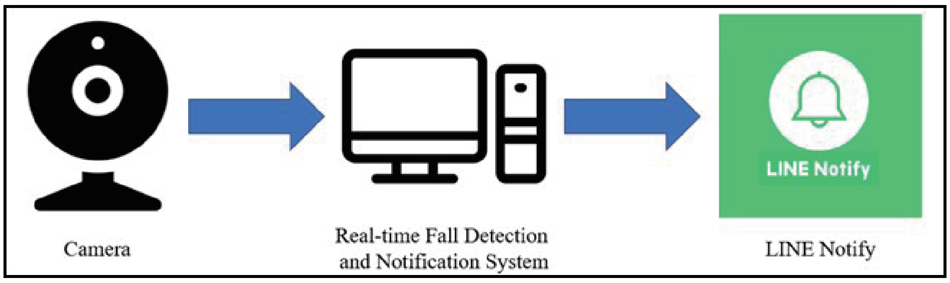


Fig. 4. (Color online) System architecture of real-time fall detection and alert system.

real-time alerting and notification functions to significantly reduce the potential harm caused by fall incidents.

Even in the absence of detected falls, the system will continue to monitor for movements, ensuring continuous surveillance. The overall flowchart of the system is illustrated in Fig. 5. Through this workflow, we ensure efficient and reliable motion detection and notification capabilities.

3.2 Environment setup

The model training and testing equipment used in this study is a laptop with an Intel Core i7-8565U 1.80 GHz processor and an NVIDIA GeForce MX150 graphics card. Additional information regarding related hardware and software versions of the development environment is provided in Table 3.

3.3 Dataset sources

The "UR Fall Detection Dataset"(32) was produced by Michal Kępski at the Interdisciplinary Centre for Computational Modelling at the University of Rzeszow. The dataset contains 70 sequences, including 30 fall events and 40 activities of daily living. Falls were recorded using two Microsoft Kinect cameras and corresponding acceleration data. Activities of daily living were recorded using only one device (Camera 0) and an accelerometer. Sensor data was collected using PS Move (60 Hz) and x-IMU (256 Hz) devices. The dataset is composed in the following way. Each row contains depth and RGB image sequences for Camera 0 and Camera 1 (floor parallel and ceiling-mounted, respectively), synchronization data, and raw acceleration data. Each movie is stored as a separate zip file in the form of a png image sequence. The depth data is stored in the PNG16 format and should be rescaled. In this paper, 30 movies from each of Camera 0 and Camera 1 are divided into a training set and a test set at a ratio of 8 to 2, and the movies are converted to photographs on a case-by-case basis. The exact number is shown in Table 4.

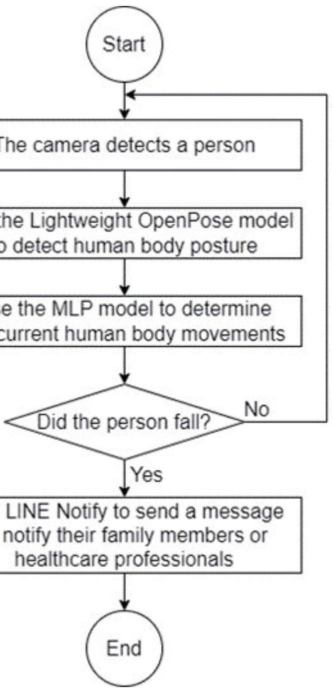

Start
The camera detects a person
Use the Lightweight OpenPose model to detect human body posture
Use the MLP model to determine the current human body movements
Did the person fall?
No
Yes
Use LINE Notify to send a message to notify their family members or healthcare professionals
End

Fig. 5. System flowchart of real-time fall detection and alert system.

Table 3
Hardware and software version information for AI model training in this study.

	Version, model, or specification
Software information	Integrated Development Environment (IDE): Anaconda 3 + PyCham 2023.1.4
	Python Version: 3.6.13
	PyTorch Version: 1.4.0
	TensorFlow Version: 2.13.0
	CUDA Version:12.2
Hardware information	Processor: Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz
	Graphics Card: NVIDIA GeForce MX150

Table 4
Dataset distribution table.[32]

Classification	Camera	Number of datasets	Total
Training dataset	Cam0	2397	4974
	Cam1	2577	
Test dataset	Com0	598	1200
	Cam1	602	

4. Experimental Phase and Discussion

4.1 Data processing for training

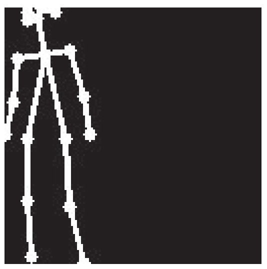
In this study, the training of the classifier is described as follows. First, we manually categorized the training images into two classes: “normal” and “fall”. Next, Lightweight

OpenPose split each category into training and testing sets and converted the images into grayscale representations of skeletal joint points, with each image sized at 128 × 128 pixels, preparing them for model training, as shown in Fig. 6.

The main reason for converting the training images to grayscale is to simplify the image data and reduce computational complexity. Grayscale images contain only brightness information, excluding color information, which is sufficient for human activity classification tasks. This approach enhances computational efficiency, reduces model complexity, and saves memory space. The normal class focuses on the grayscale skeletal joint point images for the "normal" class. The images are processed to highlight key joint positions and movements, facilitating effective training for the classification model. Each image maintains a resolution of 128 × 128 pixels, ensuring consistent input for the model while simplifying the visual data by removing color information. This allows the model to concentrate on the movement patterns essential for classification tasks.

The fall class focuses on the grayscale skeletal joint point images for the "fall" class. These images capture critical joint positions during fall scenarios, providing essential data for training the classification model. Each image is standardized to a resolution of 128 × 128 pixels, simplifying the input while eliminating color information. This allows the model to effectively analyze movement patterns related to falls, enhancing its capability to classify and detect such actions accurately. MoveNet recognizes the human skeleton by converting it into 17 keypoints, each of which contains the corresponding X and Y coordinates, confidence score, and joint point name. For example, information about a keypoint can be represented as "x: 230, y: 220, score: 0.9, name: 'nose'", which indicates the location of the keypoint and the system's confidence in its accuracy. These data points are combined and stored in the CSV format as the main input data for model training. This is shown in Figs. 7(a) and 7(b).

The benefits of using only coordinates for training in MoveNet are mainly in simplifying the data and improving the model performance. Unlike image categorization, the coordinate data only retains the key information of the movement, i.e., the position of human joints and their changes, which significantly reduces the data dimension and computational resource requirements. This approach removes irrelevant information such as background, color, and noise from the image, so that the model can focus more on the changes in movement patterns. In addition, coordinate data simplifies the training process, improves efficiency and model convergence, and is not dependent on specific camera angles or resolutions, making it more


(a)

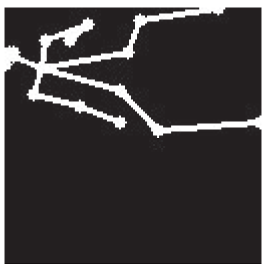

(b)

Fig. 6. Lightweight OpenPose preprocessed data for (a) normal and (b) fall categories.

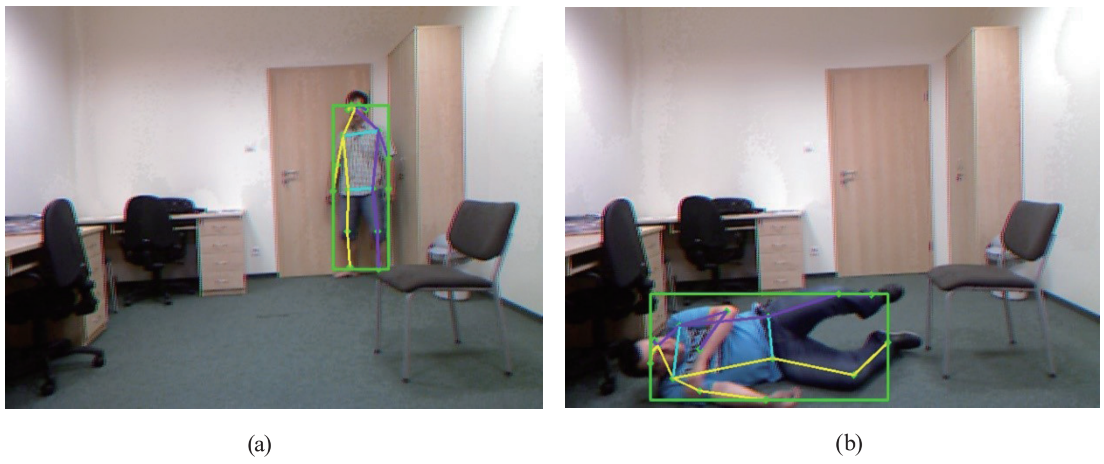


Fig. 7. (Color online) MoveNet preprocessed data for (a) normal and (b) fall categories.

adaptable. This approach is an efficient strategy for motion recognition tasks because it can focus on motion patterns, such as joint movements and posture changes, to improve classification accuracy.

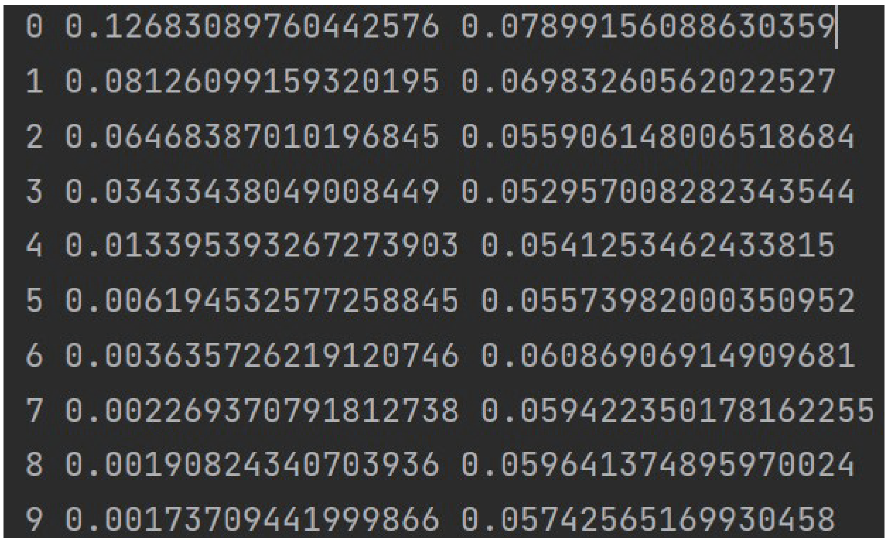
4.2 Training process

Before commencing training, we set the shuffle parameter to True. This setting ensures that the model is not affected by the order of the data during training, allowing it to learn features and patterns more effectively. We then input the training and testing sets into the MLP model to begin the training process, as shown in Fig. 8. In this figure, Lightweight OpenPose indicates the current training epoch, the second row shows the calculated average training loss (train_avg_loss), and the third row displays the calculated average testing loss (test_avg_loss). The current training epoch, training accuracy, average training loss, test accuracy (val_accuracy), and average test loss (val_loss) of MoveNet are shown in Fig. 9.

Calculating the average training loss and average testing loss is crucial for evaluating the model's performance during training and testing. These loss values provide indicators of how well the model is performing on the training and testing data. By monitoring these two loss values, we can assess whether the model is experiencing overfitting or underfitting. If the training loss continues to decrease while the testing loss increases, it may indicate overfitting, necessitating adjustments such as reducing model complexity or employing regularization techniques. Conversely, if both losses are high, it suggests that the model may require more training data or further parameter tuning.

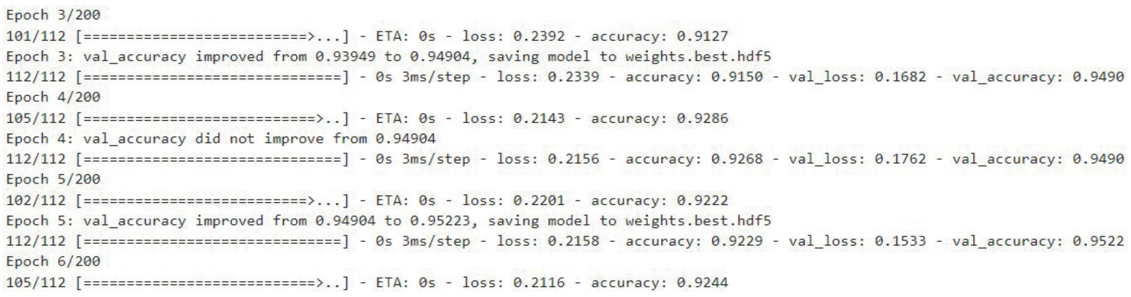
4.3 Performance evaluation metrics

In this study, we chose to use the precision–recall (PR) curve and receiver operating characteristic (ROC) curve as performance evaluation metrics, primarily because our task is a



```
0 0.12683089760442576 0.07899156088630359
1 0.08126099159320195 0.06983260562022527
2 0.06468387010196845 0.055906148006518684
3 0.03433438049008449 0.052957008282343544
4 0.013395393267273903 0.0541253462433815
5 0.006194532577258845 0.05573982000350952
6 0.003635726219120746 0.06086906914909681
7 0.002269370791812738 0.059422350178162255
8 0.00190824340703936 0.059641374895970024
9 0.00173709441999866 0.05742565169930458
```

Fig. 8. MLP model training process.



```
Epoch 3/200
101/112 [==========================>...] - ETA: 0s - loss: 0.2392 - accuracy: 0.9127
Epoch 3: val_accuracy improved from 0.93949 to 0.94904, saving model to weights.best.hdf5
112/112 [==============================] - 0s 3ms/step - loss: 0.2339 - accuracy: 0.9150 - val_loss: 0.1682 - val_accuracy: 0.9490
Epoch 4/200
105/112 [===========================>..] - ETA: 0s - loss: 0.2143 - accuracy: 0.9286
Epoch 4: val_accuracy did not improve from 0.94904
112/112 [==============================] - 0s 3ms/step - loss: 0.2156 - accuracy: 0.9268 - val_loss: 0.1762 - val_accuracy: 0.9490
Epoch 5/200
102/112 [==========================>...] - ETA: 0s - loss: 0.2201 - accuracy: 0.9222
Epoch 5: val_accuracy improved from 0.94904 to 0.95223, saving model to weights.best.hdf5
112/112 [==============================] - 0s 3ms/step - loss: 0.2158 - accuracy: 0.9229 - val_loss: 0.1533 - val_accuracy: 0.9522
Epoch 6/200
105/112 [===========================>..] - ETA: 0s - loss: 0.2116 - accuracy: 0.9244
```

Fig. 9. Training process of MoveNet MLP model.

binary classification problem between "normal" and "fall" events. The PR curve is particularly suitable for handling imbalanced datasets, as in our fall detection system, where fall events are the minority class compared with normal behavior. When dealing with this type of imbalanced data, the PR curve can more clearly demonstrate the trade-off between precision and recall, especially in detecting the critical minority class (such as falls), thus providing a better reflection of the model's performance.

On the other hand, the ROC curve is a measure of the model's capability to categorize the whole population, which is especially suitable for situations where the categories are more balanced. In binary classification problems, the ROC curve provides a more comprehensive assessment of the model's predictive performance at each threshold by considering both the true positive rate (TPR) and the false positive rate (FPR). Therefore, we use the PR curve to emphasize the detection performance for a small number of categories (fall events), whereas the ROC curve is used to comprehensively assess the overall performance of the model in handling all data categories.

The better performance of Lightweight OpenPose in terms of precision–recall area under the curve (PR AUC) and F1 scores, as shown in Fig. 10, and the better performance of MoveNet in terms of ROC AUC, as shown in Fig. 11, may be related to several factors as follows.

Modeling architecture differences:

OpenPose usually adopts multistage resolution refinement step by step, which can capture the pose points more finely, especially in small samples or unbalanced datasets. OpenPose may have better precision and recall in specific tasks (e.g., situations that require high-precision detections), which leads to better performances of PR AUC and $F1$-score.

Data characterization:

PR AUC and $F1$-score are more suitable for unbalanced datasets, especially in recognition tasks that focus on certain key poses or specific minority classes. If there is a category imbalance in the dataset, OpenPose may perform better on small sample categories.

ROC AUC assumes a more balanced dataset and can more fully evaluate the overall

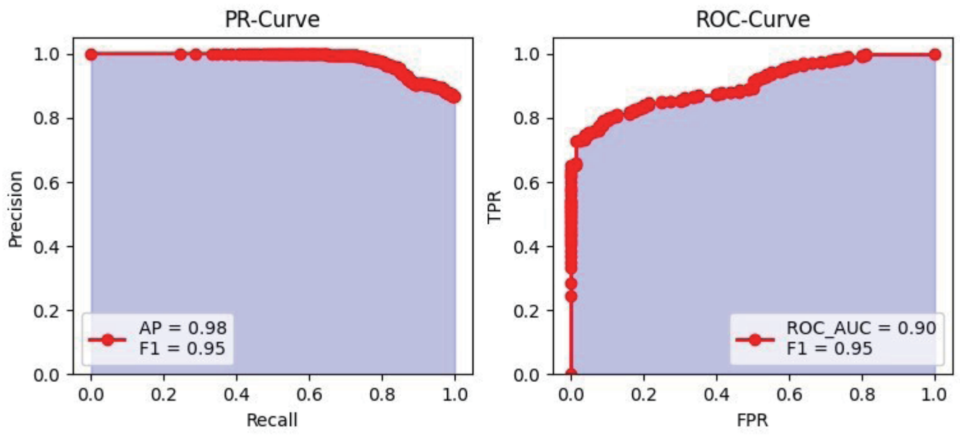


Fig. 10. (Color online) Lightweight OpenPose PR curve and ROC curve.

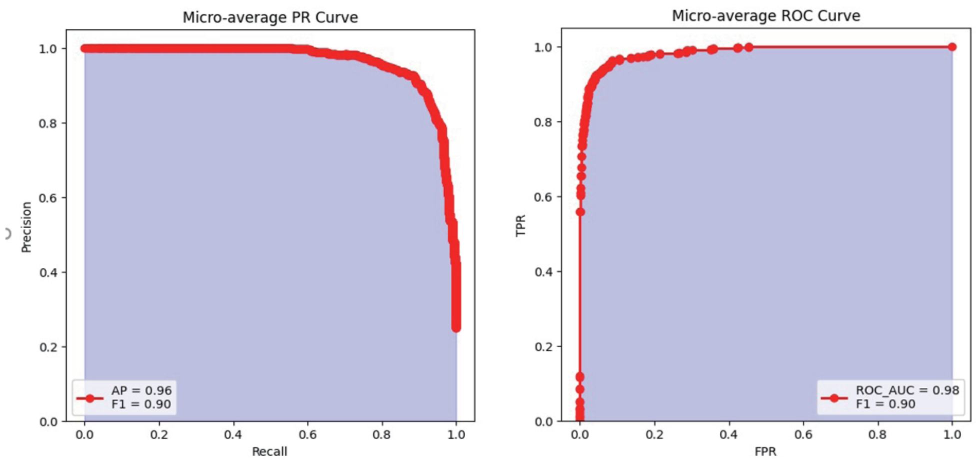


Fig. 11. (Color online) MoveNet PR curve and ROC curve.

performance of the model on all categories. MoveNet will have a better ROC AUC score if it can provide stable predictions on large or balanced datasets.

To summarize, Lightweight OpenPose performs well in refined prediction, so PR AUC and $F1$-score are likely to be higher, whereas MoveNet has an advantage in full-domain model performance; thus, ROC AUC is better.

4.4 Handling exceptions in data preprocessing

Both the fall category data sets after preprocessing using Lightweight OpenPose and MoveNet show a decrease in quantity, with Lightweight OpenPose experiencing a more significant reduction (as shown in Table 5). This may be due to some body parts being obscured during a fall, resulting in incomplete skeleton generation and leading to a class imbalance, which affects the model's performance. The occlusion issue is particularly pronounced during fall actions because, when the body falls, certain limbs (such as arms or legs) may be blocked from view, preventing Lightweight OpenPose from accurately detecting them, thus reducing the amount of fall category data. In Fig. 10, Lightweight OpenPose performs poorly on the ROC AUC since the model cannot effectively learn complete fall samples. However, because the model is trained on only unobstructed data, precision and recall are better in the absence of occlusion, leading to improved PR AUC and $F1$-score performances.

Additionally, even when the skeleton is successfully generated, especially during actions such as falling or sitting down, the system may sometimes misidentify nearby objects (such as chairs or tables) as part of the human body (as shown in Fig. 12). There may also be issues with incorrect human skeleton posture (as shown in Fig. 13). These misjudgments can affect the model's accuracy, increase the FPR, and reduce the reliability of practical applications.

These challenges highlight the limitations of fall detection systems when dealing with limb occlusion and environmental interference. The accuracy and completeness of data are crucial for model performance. Future work needs to further investigate how to improve occlusion issues and reduce the misidentification of objects to enhance the overall stability and accuracy of the model.

4.5 Notification system

4.5.1 Data preprocessing and experimental design

In Sect. 4.3, we found that using Lightweight OpenPose yields better accuracy predictions than MoveNet for the UR Fall Detection Dataset. Considering that elderly individuals only take

Table 5
Comparison of original and preprocessed dataset quantities.

	Original dataset quantity	OpenPose preprocessed data set quantity	MoveNet preprocessed data set quantity
Normal	2728	2602	2620
Fall	3446	2114	2582

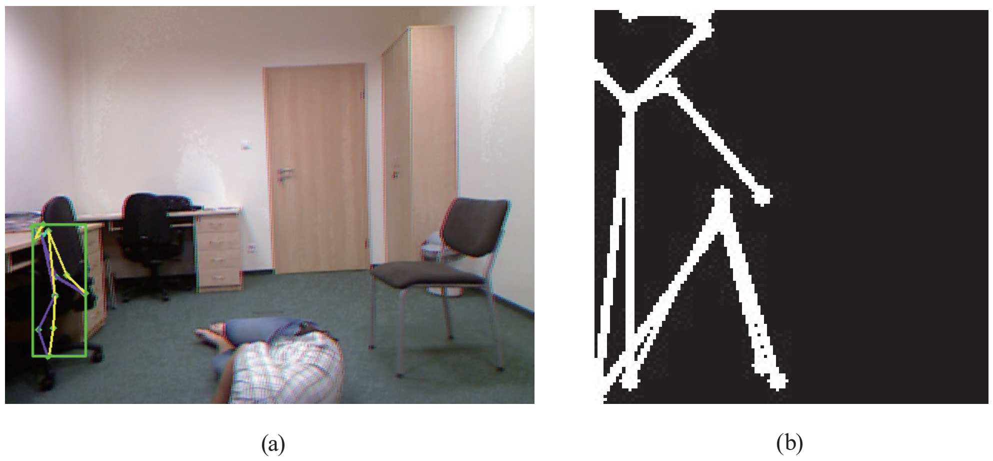


Fig. 12. (Color online) Misjudgment of limb detection results by the AI model: (a) MoveNet and (b) OpenPose.

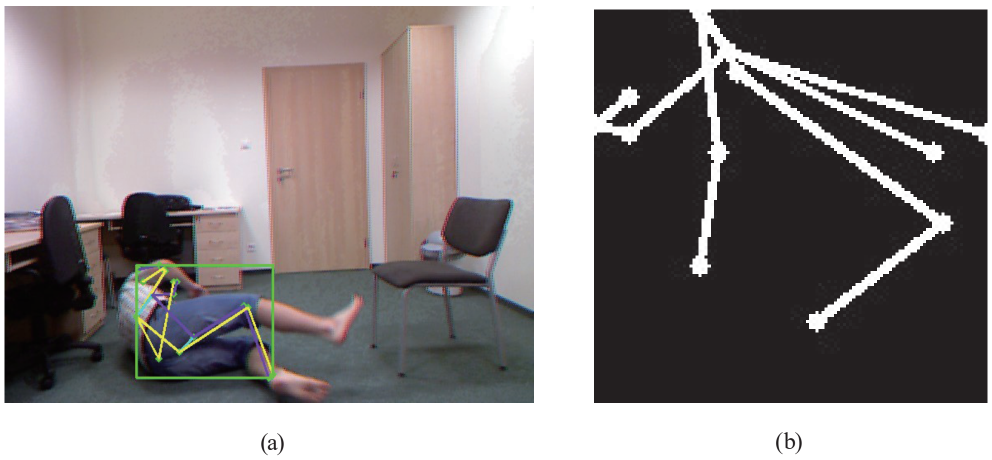


Fig. 13. (Color online) Incorrect human skeleton posture of limb detection results by the AI model: (a) MoveNet and (b) OpenPose.

1 to 2 s to go from losing their balance to fully falling, they face the danger of being unable to get up or losing consciousness due to head impact. Therefore, we have added the LINE Notify notification feature to quickly assist in rescue efforts in a timely manner. To effectively determine the duration of fall detection in order to trigger the LINE Notify alert, we conducted experiments to collect data and analyze the optimal threshold. These experiments primarily utilized the OpenPose AI model for fall detection.

In Sect. 4.4, regarding the prevention of fall incidents, we employed the GMDCSA24-A-Dataset-for-Human-Fall-Detection-in-Videos.[33] This dataset consists of photos depicting various possible activities of humans in daily life. It includes 74 videos showcasing different everyday behaviors (such as walking, bending, sitting, and falling), with each video lasting at least 5 s or more. This serves as the basis for our research integrating the OpenPose AI model with the LINE Notify notification feature for experimentation. In our data preprocessing tasks, we do not perform relabeling. Instead, we directly use the prelabeled videos, which already contain segments of various daily activities that should not be classified as falls by the model. The trained model then automatically extracts action features from these daily activities, with its output indicating whether a fall is detected.

Next, we establish a simulated detection system where the trained model analyzes each frame of daily activity videos to determine whether a fall behavior is present. The model's output will indicate a fall detection result for each frame, although it may occasionally misinterpret certain actions (such as rapid squatting or lying down) as falls. We then activate a notification mechanism, triggering the LINE Notify alert function whenever the model detects a suspected fall. The following are the experimental steps for this phase.

Step 1: Detect and record model outputs
The model will detect 74 video clips of activities of daily living and output whether there is any suspected fall behavior on each frame. When the model detects the suspected fall behavior, it will start the timer.

Step 2: Set the notification mode with different number of seconds for comparison
1-s notification mode: Once the model detects a fall, the notification will be triggered immediately.
2-s notification mode: The notification is triggered when a fall is detected and the behavior lasts for 2 s or more.
3-s notification mode: The notification is triggered when a fall is detected and the behavior persists for 3 s or more.

4.5.2 Data collection and analysis

In terms of data collection and experiments, we aim to obtain the following data:

(1) FPR Statistics: The number of times the model misclassifies daily activities as falls is recorded for each notification mode.

(2) Notification Effect Comparison: By analyzing notifications at 1, 2, and 3 s, we evaluate which setting most effectively reduces the FPR.

On the basis of this experiment, the data collection and analysis results of the OpenPose limb detection combined with the LINE Notify notification feature are shown in Table 6.

False Alarm Rate Statistics: Record the number of times the model misidentifies daily activities as falls in each notification mode.

Comparison of Notification Effectiveness: Analyze which setting (1-second, 2-second, or 3-second notifications) is most effective in reducing the false alarm rate.

Table 6
Experimental data.

Notification mode	Total video duration (s)	Total fall detection time (s)	Number of notifications	False alarm rate (%)
1-s notification mode	681	48	48	7.05
2-s notification mode	681	34	17	4.99
3-s notification mode	681	15	5	2.20

The analysis of the results in Table 5 shows that the false alarm rate for the 1-second notification mode (7.05%) indicates that notifications sent too quickly may lead to an increase in misreporting during daily activities. The 2-second notification mode shows an improvement in false alarm rate (4.99%), providing a balance between reducing false alarms and not significantly delaying notification efficiency. The 3-second notification mode significantly reduces the false alarm rate (2.20%); while extending the fall detection time may help improve accuracy, it is also necessary to consider the timeliness for actual fall events. On the basis of the analysis of the above data, setting the LINE Notify notification threshold to 3 s may be the best choice, as it significantly lowers the false alarm rate while reducing misreporting. If more immediate responses are required in future applications, further adjustments to the notification time can be considered, even tailoring it for specific situations.

When our system detects that a person has fallen in the video and that the fall persists for more than 3 s, it will immediately send a message to notify family members or medical personnel using the LINE Notify service, as shown in Fig. 14. The program tracks the start time of the fall event through the fall_start_time variable and calculates the duration after the event occurs. If it exceeds 3 s, the LINE Notify notification will be triggered. If the system detects that a person has fallen for 2.5 s and then immediately gets back up, determining the current state to be normal, the start time of the fall event will be recalculated. This mechanism can be applied to the real-time monitoring of fall events, providing quick alerts and notifications to enhance the safety monitoring system. This real-time notification mechanism ensures that relevant personnel are

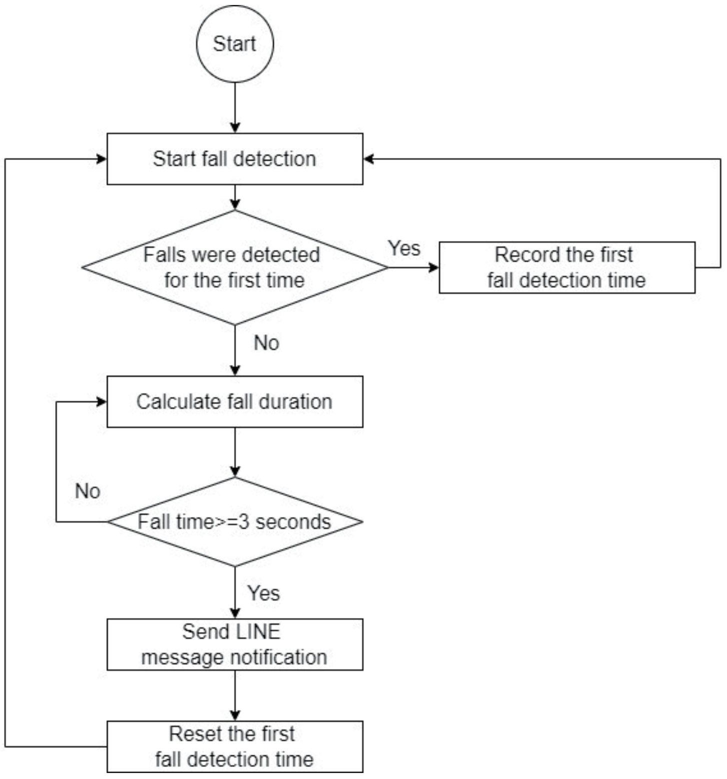

Start
Start fall detection
Falls were detected for the first time
Yes
Record the first fall detection time
No
Calculate fall duration
No
Fall time>=3 seconds
Yes
Send LINE message notification
Reset the first fall detection time

Fig. 14. LINE Notify notification flowchart.

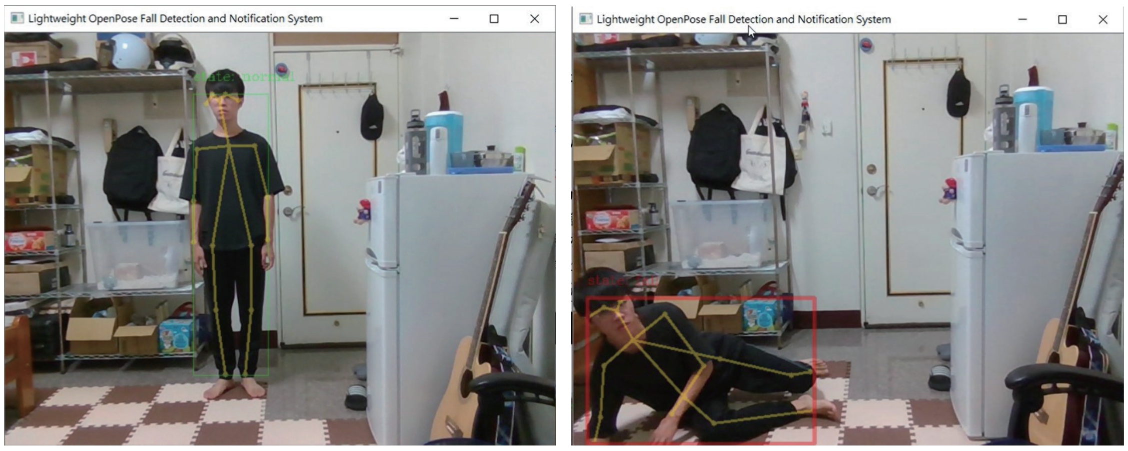
promptly informed when a fall occurs, maximizing the reduction of potential damage risks and demonstrating the system's efficiency in practical applications.

4.6 Experimental results

During the system execution phase, we captured images of individuals appearing in the video using a camera and employed the Lightweight OpenPose and MoveNet models for joint detection. Subsequently, we utilized the MLP model for human activity recognition based on the detected joint points, displaying the current status above the detection box. When the system determines that the individual in the frame is "normal," both the detection box and the status text are shown in green, as illustrated in Figs. 15(a) and 16(a). Conversely, if the system identifies the individual as "falling," the detection box and status text appear in red, as shown in Figs. 15(b) and 16(b). If the fall state persists for more than 3 s, the system will promptly send a notification to family members or healthcare personnel via LINE Notify, as depicted in Fig. 17. This design not only intuitively presents the status of individuals in the frame but also facilitates rapid notification in the event of a fall, enhancing response speed and further reducing potential harm risks.

4.7 Discussion

The novelty of this study lies in analyzing the differences between OpenPose and MoveNet models in dealing with human pose estimation. Compared with previous studies, such as that by Kepski and Kwolek, they used a k-NN classifier to analyze action sequences and perform fall detection. To reduce processing overload in complex scenes, an accelerometer was employed to indicate potential impacts and initiate depth image analysis.(34) Cao *et al.* demonstrated the effectiveness of a two-branch CNN for posture estimation in fall detection, achieving high


Lightweight OpenPose Fall Detection and Notification System
State: normal
Lightweight OpenPose Fall Detection and Notification System

(a) (b)

Fig. 15. (Color online) Lightweight OpenPose model detection results: (a) Normal state and (b) Fall state.

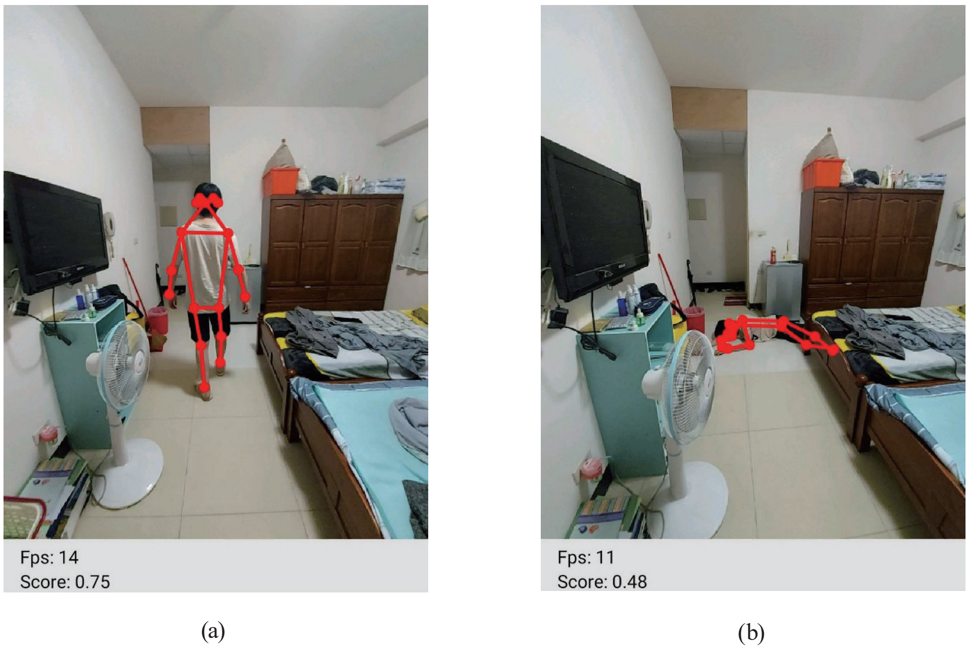
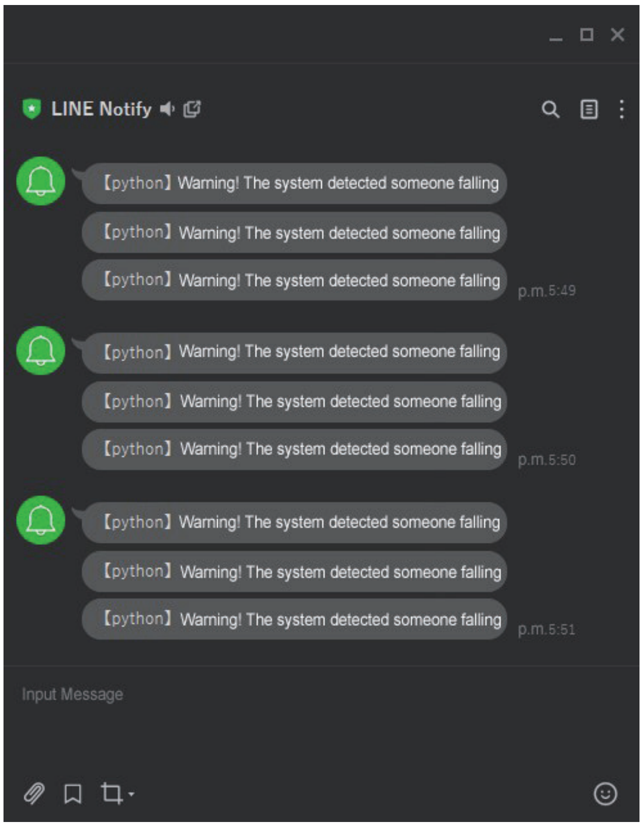


Fig. 16. (Color online) MoveNet model detection results: (a) Normal state and (b) Fall state.



【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling p.m.5:49
【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling p.m.5:50
【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling
【python】 Warning! The system detected someone falling p.m.5:51

Fig. 17. (Color online) The fall detection and alert system sends real-time fall notifications via LINE Notify.

accuracy and real-time performance, regardless of the number of individuals in the image.(20) Osokin proposed Lightweight OpenPose(4), an adaptation of the original OpenPose architecture for use on edge devices. This model significantly improved computational efficiency while maintaining multiperson pose estimation capabilities. In our study, we employed OpenPose to demonstrate its effectiveness in pose estimation for fall detection but highlighted high computational costs as a limitation. By integrating Lightweight OpenPose, as implemented in this work, we provide a lower computational alternative. The TensorFlow Team introduced the architecture, performance, and advantages of MoveNet in low-latency applications. They demonstrated benchmark results on the COCO dataset and analyzed MoveNet's latency performance on various devices, such as mobile phones and embedded systems.(5)

In this study, we conducted a comparative analysis of lightweight AI models (Lightweight OpenPose and MoveNet) for real-time fall detection. Our innovation lies in the following:
(1) Comparing the performances of the two models in low-computation environments,
(2) Integrating the models into an instant alert system via LINE Notify, specifically tailored for elderly care applications, and
(3) Addressing class imbalance and occlusion issues in fall detection datasets, enhancing the practical applicability of the system.

5. Conclusions

In this study, we successfully established an efficient and real-time fall detection and alert system. By combining a web camera as the sensor with the Lightweight OpenPose and MoveNet models for sensing human pose points, we not only achieved real-time detection but also effectively reduced the computational burden on the device during this process. We further used an MLP model to accurately identify movements of the skeletal joints, ensuring that the notification procedure could be quickly activated after a fall event, thereby effectively reducing bodily harm. Additionally, we integrated LINE Notify services to enable real-time notification functionality. When an individual experiences a fall, the system can immediately send notifications to their family members or healthcare personnel. This helps ensure a swift response to fall events, minimizing damage while improving the individual's quality of life. This not only enhances the responsiveness to fall incidents but also shortens rescue times, further reducing the potential for injury. The application of this technology is beneficial not only for improving the quality of life for the elderly and individuals with special needs but also provides valuable practical applications for related medical fields.

In the performance evaluation of AI models, Lightweight OpenPose demonstrated better PR AUC and F1 scores, especially when handling imbalanced datasets, as its model architecture effectively captures pose points. On the other hand, MoveNet excelled in ROC AUC performance, making it suitable for more balanced datasets and large-scale predictions, showcasing its advantages in global model performance. In terms of challenges, both Lightweight OpenPose and MoveNet exhibited a reduction in performance in the "fall" category in the preprocessed data, with Lightweight OpenPose showing a more significant decrease. This is due to occlusions of the limbs during falls, leading to incomplete skeleton generation and causing class imbalance issues. Furthermore, even when a skeleton is generated, there may still be cases of partial deformation or misidentification of objects as human bodies, which poses challenges to the system's accuracy.

For future research, we suggest further extending the system's capabilities by classifying a wider range of actions to meet different usage scenarios and requirements. This includes enhanced support for identifying a variety of daily activities, thereby providing more comprehensive health monitoring services. Through continuous technological progress and system upgrades, this research result will have a more profound and positive impact on medical applications. It is suggested that future research may include the following:

- Pose estimation analogy: The image recognition and motion classification techniques used in this study for human fall detection can serve as a blueprint for developing similar systems that can identify defects in moving parts through visual data analysis.
- Lightweight models for real-time applications: The focus is on lightweight AI models optimized for low-computing environments, especially for resource-constrained industrial applications where efficiency is critical.
- Integration and automation: Similarly, just like looking into integrating notification systems for instant alerts, machine learning can be combined with IoT technology to monitor production processes in real time. This integration helps streamline manufacturing operations by helping detect production anomalies early and ensuring consistent material quality.

References

1 CDC: https://wonder.cdc.gov/ucd-icd10.html (accessed February 2024).
2 CDC: https://wonder.cdc.gov/mcd.html (accessed February 2024).
3 Falls: https://www.who.int/news-room/fact-sheets/detail/falls (accessed January 2024)
4 D. Osokin: Proc. 8th Int. Conf. Pattern Recognition Applications and Methods (SciTePress, 2019) 744–748. https://doi.org/10.5220/0007555407440748
5 TensorFlow: https://www.tensorflow.org/hub/tutorials/movenet (accessed October 2024).
6 J. Brieva, H. Ponce, E. Moya-Albor, and L. Martínez-Villaseñor: Proc. 2019 IEEE 14th Inter. Symp. Autonomous Decentralized System (ISADS). (IEEE, 2012) 1–5. https://doi.org/10.1109/ISADS45777.2019.9155767
7 S. Rastogi and J. Singh: Soft Comput. **26** (2022) 3679. https://doi.org/10.1007/s00500-021-06717-x
8 A. Biswas and B. Dey: Advances in Communication, Devices and Networking, R. Bera, P. C. Pradhan, C. M. Liu, S. Dhar, and S. N. Sur Eds. (Springer, Singapore, 2020). https://doi.org/10.1007/978-981-15-4932-8_46
9 V. Harsh, B. Hetvi, and M. Raj: Proc. 2023 7th Int. Conf. Intelligent Computing and Control Systems (IEEE, Madurai, India, 2023) 541–546. https://doi.org/10.1109/ICICCS56967.2023.10142753
10 D. G. Lowe: Int. J. Comput. Vision **60** (2004) 91. https://doi.org/10.1023/B:VISI.0000029664.99615.94
11 P. Viola and M. J. Jones: Int. J. Comput. Vision **57** (2004) 137. https://doi.org/10.1023/B:VISI.0000013087.49260.fb
12 G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and C. I. Sánchez: Med. Image Anal. **42** (2017) 60. https://doi.org/10.1016/j.media.2017.07.005
13 A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun: Nature **542** (2017) 115. https://doi.org/10.1038/nature21056
14 H. P. Chan, R. K. Samala, L. M. Hadjiiski, and C. Zhou: Advances in Experimental Medicine and Biology, G. Lee and H. Fujita, Eds. (Springer, Cham, 2020) 1213. https://doi.org/10.1007/978-3-030-33128-3_1
15 C. Chen, A. Seff, A. Kornhauser, and J. Xiao: Proc. 2015 IEEE Int. Conf. Computer Vision (IEEE 2015) 2722–2730. https://doi.org/10.1109/ICCV.2015.312
16 C. Kim, M. Lee, K. H. Hwang, and Y. Ha: J. Supercomput. **78** (2021) 1961. https://link.springer.com/article/10.1007/s11227-021-03929-8

17 Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh: Proc. IEEE Conf. Comput. Vision and Pattern Recognition (IEEE, 2017) 7291–7299. https://arxiv.org/pdf/1612.00137 (accessed April 2025).

18 A. Newell, K. Yang, and J. Deng: Proc. European Conf. Computer Vision 2016, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. (Springer, Cham., 2016) 9912. 483–499. https://doi.org/10.1007/978-3-319-46466-4_29

19 S. Kreiss, L. Bertoni, and A. Alahi: Proc. 2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (IEEE, Long Beach CA, 2019) 11969–11978. https://doi.org/10.1109/CVPR.2019.01225

20 Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh: Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition (IEEE, 2017) 1302–1310. https://doi.org/10.1109/CVPR.2017.143

21 P. Xu , A. Kiilerich, R. Blattmann, Y. Yu, S.-L. Zhu, and K. Mølmer: Phys. Rev. A: At. Mol. Opt. Phys. **96** (2017) 010101. https://doi.org/10.1103/PhysRevA.96.010101.

22 CenterNet: Keypoint Triplets for Object Detection: https://arxiv.org/pdf/1904.08189.pdf (accessed October 2024).

23 G. Papandreou, T. Zhu, L. C. Chen, S. Gidaris, J. Tompson, and K. Murphy: PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model, V. Ferrari, M. Hebert, C. Sminchisescu and Y. Weiss, Eds. Computer Vision – ECCV 2018. Lecture Notes in Computer Science **11218** (Springer, Cham, 2018) 282. https://doi.org/10.1007/978-3-030-01264-9_17

24 Next-generation pose detection with MoveNet and TensorFlow.js. TensorFlow Blog: https://blog.tensorflow.org/2021/05/next-generation-pose-detection-with-MoveNet-and-tensorflowjs.html (accessed October 2024).

25 Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner: Proc. the IEEE 86 (IEEE, 1998). 2278–2324. https://doi.org/10.1109/5.726791.

26 W. Zaremba, I. Sutskever, and O. Vinyals: Proc. 2015 Int. Conf. on Learning Representations (ICLR) (ICLR, 2015) arXiv:1409.2329. https://doi.org/10.48550/arXiv.1409.2329

27 I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy: https://arxiv.org/abs/2105.01601 (accessed October 2024).

28 S. H. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee: Neurocomputing **378** (2020) 112. https://doi.org/10.1016/j.neucom.2019.10.008.

29 X. Glorot, A. Bordes, and Y. Bengio: Proc. the 14th Int. Conf. Artificial Intelligence and Statistics (Fort Lauderdale, FL, USA., 2011) 315–323. https://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

30 T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur: Proc. 2011 IEEE Int. Conf. Acoustics, Speech and Signal Processing (Prague, Czech Republic, 2011) 5528–5531.

31 D. Rumelhart, G. Hinton, and R. Williams: Nature **323** (1986) 533. https://www.nature.com/articles/323533a0

32 UR Fall Detection Dataset: http://fenix.ur.edu.pl/~mkepski/ds/uf.html (accessed October 2024).

33 E. Alam, A. Sufian, P. Dutta, M. Leo, and I. A. Hameed: Data Brief. **57** (2024) 110892. https://doi.org/10.1016/j.dib.2024.110892

34 M. Kepski and B. Kwolek: Proc. 2014 Int. Conf. Computer Vision Theory and Applications (IEEE, 2014) 640–647.