IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 24 July 2023, accepted 16 August 2023, date of publication 21 August 2023, date of current version 31 August 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3307138*

SURVEY

# Advances in Skeleton-Based Fall Detection in RGB Videos: From Handcrafted to Deep Learning Approaches

**VAN-HA HOANG[1], JONG WEON LEE[1], (Member, IEEE), MD. JALIL PIRAN[2], (Senior Member, IEEE), AND CHUN-SU PARK[3]**

[1]Department of Software, Sejong University, Seoul 05006, South Korea
[2]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea
[3]Department of Computer Education, Sungkyunkwan University, Seoul 03063, South Korea

Corresponding author: Chun-Su Park (cspk@skku.edu)

This work was supported in part by the Technology Development Program through the Korean Ministry of Small and Medium Enterprises (SMEs) and Startups under Grant S3147433; and in part by the Ministry of Science and ICT (MSIT), South Korea, under the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP) under Grant IITP-2022-RS-2022-00156354.

**ABSTRACT** In the elderly population, falls are one of the leading causes of fatal and non-fatal injuries. Fall detection and early alarms play an important role in mitigating the negative effects of falls, especially given the growing proportion of the elderly population. Due to their non-intrusive nature, data availability, and low deployment costs, RGB videos have been used in many previous studies to detect falls. The RGB data, however, can be affected by background environment changes, resulting in non-recognition. To overcome these challenges, many researchers propose extracting skeleton data from RGB videos and using it for fall detection. Although there have been multiple surveys on fall detection, most of them focus on assessing fall detection systems using different kinds of sensors, and a comprehensive evaluation of skeleton-based fall detection in RGB videos is lacking. In this paper, we examine the most recent advances in skeleton-based fall detection in RGB videos, from handcrafted feature-based methods to advanced deep learning algorithms. Further, we present several skeleton-based fall detection techniques and their performance results on various benchmark datasets, along with challenges and future directions in this field.

**INDEX TERMS** Deep Learning, fall detection, pose estimation, RGB video, skeleton sequence, skeleton-based fall detection.

## I. INTRODUCTION

Globally, the population is aging rapidly. Approximately one-sixth of the world's population will be elderly (60 years or older) by 2030, and the population of this age group will reach 2.1 billion by that time [1]. As a result, it is becoming increasingly necessary to develop intelligent systems to support the elderly healthcare system. The Centers for Disease Control and Prevention of the United States report that falls are the predominant cause of both fatal and non-fatal injuries within the older population [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

Fall detection and early alarm play a crucial role in reducing the negative consequences of falls, particularly for the elderly. Thus, many fall detection systems have been developed [3], [4], [5]. According to [6], fall detection consists of three types of sensors as: 1) wearable devices, 2) ambient devices, and 3) cameras.

However, wearable devices may not be remembered to be worn (especially for the elderly) or may cause discomfort to users who must wear them all day, despite their effectiveness in detecting falls [7], [8], [9]. Ambient device-based approaches such as utilizing pressure sensors, infrared sensors, etc., can be relatively expensive [10], or may work only in a limited area (often indoor environment) [11], [12],

or may require complex set-up [13]. Due to their non-intrusive nature, data availability, and low deployment costs, cameras are also promising sources of fall detection information [14], [15], [16].

There are two types of cameras used for camera-based fall detection: 1) depth cameras (e.g., Microsoft Kinect [17], Intel RealSense [18]) and 2) RGB cameras. The RGB cameras are more popular than depth cameras due to their lower cost and easier deployment, resulting in a large amount of data. It is important to note that using only RGB video data can result in inaccurate action recognition due to the effects of context and background information [19], [20]. The fall recognition process can, therefore, be simplified by omitting the background information from RGB videos and extracting only the skeleton data (joints) of the human body.

The machine learning (ML) field has witnessed a shift from handcrafted features to deep learning (DL) approaches in recent years [21]. DL-based approaches can learn features directly from data instead of requiring domain knowledge, feature engineering skills, and time-consuming trial-and-error processes. Many vision tasks have been addressed by DL, including but not limited to image classification [22], face recognition [23], object detection [24], and action recognition [25]. To extract human skeleton data from RGB video for downstream tasks, including fall detection, several DL-based methods have been proposed [26], [27], [28].

Most fall detection studies examine methods that use various types of input sensors without focusing on a specific type of sensor data, as in [29], or focusing on a particular type of technique, as in [30]. Based on RGB videos as the input data, this survey provides a comprehensive review of both handcrafted-based and DL-based approaches for skeleton-based fall detection. This is the first comprehensive review of state-of-the-art skeleton-based fall detection methods in RGB videos of which we are aware.

Overall, the main contributions of this survey can be summarized as follows:

- Reviewing state-of-the-art handcrafted-based and DL-based approaches for skeleton-based fall detection in RGB videos.
- Presenting extensive performance comparisons of the reviewed methods across several popular benchmark datasets with brief summaries and insightful discussions.
- Highlighting the potential requirements, and challenges in skeleton-based fall detection in RGB videos.
- Outlining several future research directions.

The rest of this paper is organized as follows. Section II reviews the related surveys on fall detection. In Section III, we introduce the fundamental traditional ML algorithms and DL networks that are used in the works reviewed in this paper. In Section IV, we present the overall pipeline and conduct a thorough review and comparison of the state-of-the-art methods of skeleton-based fall detection using RGB videos. In Section V, we discuss requirements, challenges, and future research directions for this research area. Finally, Section VI draws the conclusions.

## II. RELATED SURVEY ARTICLES

Fall detection has been the subject of several publications. The authors in [31] classified fall detection into three distinct categories based on the sensor type, including wearable sensors, ambient sensors, and camera sensors. The authors presented typical components of a fall detection system and discussed the disadvantages of each type of sensor. Additionally, the work also discussed the design of personalized fall detection models by training classifiers with both user and non-user data. In this study, handcrafted features and traditional machine algorithms were only used to detect falls and DL approaches were not investigated.

In [32], the authors investigated the use of radio frequency and the combination of multiple sensors for fall detection. Keypoints were also discussed for fall detection, but they were extracted from depth videos rather than RGB videos. In [30], the authors reviewed various DL methods for fall detection. Only a few works related to skeleton-based fall detection were presented, since the work focused on various types of sensors. In [29], the authors examined the development of fall detection systems from both hardware and software perspectives. There were also some DL methods presented for fall detection, but they were not relevant to skeleton data.

By defining vision data into global, local, and depth descriptors, the authors in [33] provide an overview of vision-based fall detection systems. Despite the introduction of skeleton-based fall detection methods, this study did not investigate the latest state-of-the-art methods regarding modern DL architectures such as Graph Neural Networks (GNN) [35]. A literature review was conducted by the authors of [6] on vision-based fall detection systems using six types of techniques: inactivity/body shape change, posture, 3D head motion, spatial-temporal approaches, gait, and skeleton tracking. There were only a few skeleton-based fall detection methods investigated, and the period of the reviewed papers was only limited to 2019.

A recent study [34] reviewed the state-of-the-art DL methods for vision-based fall detection, including skeleton data. However, the system pipeline for skeleton-based fall detection in RGB videos using both hand-crafted features and DL approaches is not fully described. Additionally, only a limited number of works related to skeleton-based fall detection in RGB videos were reviewed.

For easy comparison, we provide a list of all the survey papers related to fall detection in Table 1. In this work, we aim to comprehensively cover all the aspects of skeleton-based fall detection in RGB videos, including the detailed system pipeline and the latest state-of-the-art methods that utilize advanced DL architectures. Furthermore, we also discuss challenges in building practical systems and outline some future research directions.

**TABLE 1. Summary of existing surveys on fall detection. Sensor Type: W = Wearable, A = Ambient, C = Camera, Dep. = Depth.**

| Survey | Sensor Type | | | | System pipeline | Handcrafted-based methods | DL-based methods | Remarks |
|---|---|---|---|---|---|---|---|---|
| | W | A | C | | | | | |
| | | | Dep. | RGB | | | | |
| Vallabh *et al.* 2018 [31] | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✘ | Works in range 2005-2017. |
| Ren *et al.* 2019 [32] | ✔ | ✔ | ✔ | ✔ | ✚ | ✔ | ✚ | Some works used keypoints extracted from depth cameras. |
| Islam *et al.* 2020 [30] | ✔ | ✔ | ✔ | ✔ | ✚ | ✘ | ✔ | Only few works for skeleton-based fall detection. |
| Wang *et al.* 2020 [29] | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✚ | Only one reviewed work related to skeleton data. |
| Gutiérrez *et al.* 2021 [33] | ✔ | ✔ | ✔ | ✔ | ✚ | ✔ | ✚ | Modern deep neural network architectures are not investigated. |
| Rastogi *et al.* 2022 [6] | ✔ | ✔ | ✔ | ✔ | ✔ | ✔ | ✚ | Only a few works related to skeleton-based fall detection. References for this topic only up to 2019. |
| Alam 2022 [34] | ✖ | ✖ | ✔ | ✔ | ✔ | ✘ | ✔ | System pipeline for skeleton-based fall detection is not fully investigated. The number of reviewed works on skeleton-based fall detection is limited. |
| Our survey 2023 | ✖ | ✖ | ✖ | ✔ | ✔ | ✔ | ✔ | System pipeline for skeleton-based fall detection in RGB videos is presented in more detail. This paper reviews more recent works covering both handcrafted features and DL approaches including ones utilizing advanced DL architectures. |

✔: The paper examined the factor, ✚: The paper investigated the factor to some extent, ✘: The paper did not take into account that factor.

## III. MACHINE LEARNING METHODS AND DEEP NETWORKS FOR SKELETON-BASED FALL DETECTION IN RGB VIDEOS

The skeleton-based fall detection pipelines typically employ traditional ML methods for classification, while DL methods are used for feature extraction and representation learning. This section provides a brief overview of both traditional ML methods and DL techniques applied to skeleton-based fall detection in the reviewed papers.

### A. DECISION TREE (DT)

There are several popular non-parametric supervised algorithms for classification called Decision Trees (DTs) [36]. They create a tree-like structure from the root to the leaf node. Every node represents a test applied to an attribute, each branch represents an answer to the test of the node with a particular attribute value, and the leaf nodes represent the class labels (classification output). In DTs, data samples are classified by following different paths from the root of the tree to the leaf node. Each traversal path represents a classification rule. There are various algorithms used for constructing decision trees such as ID3 [37], CART [38], etc. In the reviewed paper [39], the authors used DT in the classification phase.

### B. K-NEAREST NEIGHBORS (KNN)

K-Nearest Neighbors (KNN) [40] is a straightforward, non-parametric supervised learning technique that can be used for both classification and regression. A KNN classifier works by finding the K nearest neighbors of a new example and assigning it to the class with the highest number of votes. By calculating the distance between the new example and all the training examples, the K nearest neighbors is identified. Distance metrics include Euclidean distance, Manhattan distance, etc. As a classification technique, KNN has been employed in several reviewed studies [39], [41], [42], [43].

### C. RANDOM FOREST (RF)

Both classification and regression applications use Random Forest (RF) [44]. This ensemble learning method constructs an uncorrelated forest of decision trees, with each decision tree being built on a random subset of training data (using bagging technique [45]) and a random subset of features (using the random subspace method [44]), ensuring low correlation among the trees in the forest. RF's final class prediction is determined by the majority vote of all trees in the forest. Several reviewed studies have used RF for classification [39], [41], [42], [46], [47].

### D. SUPPORT VECTOR MACHINE (SVM)

Support Vector Machines [48] (SVMs) are well-known supervised learning models used for classification and regression. SVM's primary objective in classification is to identify a hyperplane within an N-dimensional space, where N represents the number of features, that can effectively

distinguish data points into distinct categories. The main objective in SVM training is to maximize the distances, commonly known as margins, between the hyperplane and the nearest data points on each side. The SVM can be extended to the kernelized SVM in the case of non-linear data, which uses the kernel function to shift the original data into a higher-dimensional space where the data can be linearly separated. There have been numerous reviewed studies using SVM for classification in skeleton-based fall detection systems [39], [41], [42], [43], [46].

### E. BOOSTING

Boosting is an ensemble learning algorithm that involves sequentially training a set of weak learners. In the training of the next weak learner, data points misclassified by the previous weak learner are given more weight. By combining the predictions of all the trained weak learners, the final prediction is made. Boosting algorithms include AdaBoost [49], XGBoost [50], etc. In the reviewed work [43], AdaBoost and XGBoost were applied during the classification phase.

### F. NEURAL NETWORK (NN)

Neural Network (NN) [51] simulates the structure of the human brain. There are several layers in a conventional NN: an input layer, an output layer, and one or more hidden layers. Each layer consists of artificial neurons with their own weights and bias. The neurons perform a linear transformation on the input from the previous layer, apply a non-linear activation function, and produce an output. Activation functions such as Tanh, Sigmoid, and ReLU (Rectified Linear Unit) [52] play an important role in introducing non-linear transformations to the model.

In an NN, input data enters the input layer and then passes through the hidden layers before reaching the output layer. By using the backpropagation algorithm [53], weights and biases are updated to minimize the loss function. In binary classification tasks like fall detection (Fall or No Fall), the loss function is typically the Binary Cross-Entropy (BCE). To compute the probability of each class in an NN, the Softmax activation function is commonly used [54].

A feedforward neural network (FNN) is a simple type of NN in which the data goes through one direction without any circle or loop in the network. Multilayer Perceptron (MLP) consists of at least three layers, specifically an input, an output layer, and one or more hidden layers. In classification and regression problems, MLP is widely used. A number of reviewed papers have used MLP for classification purposes [41], [46], [47], [55], [56], [57]. Several reviewed studies have employed the Softmax layer as a classification output layer [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68].

### G. AUTOENCODER (AE)

Autoencoder (AE) [69] are NNs specifically designed to reconstruct the input signal. There are two major components in this network: an encoder and a decoder [69]. In an encoder, input signals are converted into their corresponding representations in a lower-dimensional latent space. Conversely, a decoder reconstructs the latent values back into the original space to reproduce the original data. Autoencoder exhibits a wide range of applications, notably in dimensionality reduction [70], anomaly detection [71], and other related areas. An AE-based approach to skeleton-based fall detection was proposed by the authors of [72].

### H. CONVOLUTIONAL NEURAL NETWORK (CNN)

Conventional NNs use multiple fully connected (FC) layers with complete connectivity between neurons. However, this approach has drawbacks: a large number of parameters, high computational costs, and a failure to use local information in the input data. A convolutional neural network (CNN) [73] uses weight sharing and local connectivity mechanisms to take advantage of local structure in the input data.

CNNs have three major layer types: convolutional layer, pooling layer, and FC layer. A convolutional layer produces feature maps by moving a set of learnable filters or kernels across the input data. To introduce non-linearity into the model, these feature maps are typically applied with a non-linear activation function, such as ReLU [52]. The convolutional layer can be 1D, 2D, or 3D depending on the dimensionality of the input data. Pooling layers, also known as downsampling layers, are used to reduce dimensionality. In the network, they reduce the number of parameters and computational costs. By moving weightless filters across the input data, pooling layers perform aggregation functions. Maximum pooling and average pooling are the two main types of pooling layers. Pooling layers such as Global Average Pooling (GAP) and Global Max Pooling (GMP) can generate a feature vector for classification [74]. FC layers, which typically locate at the end of the network, take features from the previous layers, and produce the final output, typically for classification tasks.

There have been many CNN architectures proposed for different tasks, including Resnet [22] with residual connections for image classification, YOLO [75] for object detection, etc. CNN was incorporated into the NN architectures in several reviewed studies [62], [63], [64], [65], [67], [76].

### I. RECURRENT NEURAL NETWORK (RNN) AND ITS VARIANTS

Recurrent Neural Network (RNN) can be used to capture the temporal information in sequential data [77]. As opposed to FNN, RNN forms a cycle/loop that allows information to be passed from time step $t$ to time step $t+1$.

In general, RNNs are NN with a hidden state $\boldsymbol{h}$ processing the input sequence $\boldsymbol{x}(x_1, x_2, \ldots, x_T)$, where the sequence length $T$ can vary. At each time step $t$, the hidden state $\boldsymbol{h}_t$ and the output $y_t$ are updated as follows:

$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, x_t) \tag{1}$$

$$y_t = g(\boldsymbol{h}_t) \tag{2}$$

where $f$ and $g$ are non-linear activation functions. The back-propagation through time (BPTT) algorithm is commonly employed to train RNNs [78]. It is possible to form stacked RNNs by stacking multiple RNN layers, where the hidden state output of one layer is used as the input of the next layer.

To mitigate the gradient vanishing and exploding problems when training RNNs, Long Short-Term Memory (LSTM) [79] and Gated Recurrent Unit (GRU) [80] are introduced. LSTM and GRU use gating mechanisms to regulate the information flow. Unlike LSTM with three gates (input gate, forget gate, and output gate), GRU only has two gates (update gate and reset gate), making it less computationally expensive than LSTM. Empirical studies have not been able to determine which is better, LSTM or GRU [81].

RNN models, such as the one presented in [59], can be used to detect falls from sequences of frames in input videos. Several reviewed studies have also used LSTM networks for this purpose [56], [57], [58], [59], [60], [61], [62], [63], [82]. Similarly, GRU networks can also be used to capture and learn the temporal dependencies required for accurate fall prediction, as demonstrated in [59], [63], [65].

### J. ATTENTION MECHANISM AND TRANSFORMER

#### 1) ATTENTION MECHANISM

Attention mechanisms [83] were originally introduced to enhance Encoder-Decoder architectures [84] in machine translation tasks by overcoming the limitations of using fixed-length context vectors [85], [86]. A typical attention mechanism does not use a static context vector derived from the encoder as in the conventional approach. Decoding involves the decoder considering the encoder's hidden states and assigning weights to them for generating the context vector, which is then used to generate output at each time step. As a result, the decoder concentrates on the important elements of the input sequence. Multiple reviewed works have integrated attention mechanisms into their network architectures [57], [63], [87].

#### 2) TRANSFORMER

As a result of the sequential nature of Encoder-Decoder models with RNNs, LSTM, and GRU, the Transformer architecture [88] provides a solution to the parallelization limitation. By using self-attention mechanisms, it can process the whole input sequence at once while still incorporating the benefits of attention mechanisms. Transformer uses encoder-decoder architecture that has multiple layers of self-attention and feed-forward. Transformer's self-attention module is its most vital component.

The input sequence is first transformed into a set of vector embeddings, where each vector represents an element in the sequence. After being injected position information, the embeddings are used to generate three types of vectors: Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) vectors. $\mathbf{Q}$, $\mathbf{K}$ vectors are subsequently employed to compute the attention weights, which indicate the degree of relevance of each element with respect to the other elements in the sequence. Finally, the attention weights are combined with the $\mathbf{V}$ vectors to produce output vectors, which are rich representations that capture the dependencies between the elements in the sequence.

Transformer architectures have been applied to various domains, including natural language processing with models such as BERT (Bidirectional Encoder Representations from Transformers) [89] and GPT-3 [90], as well as in computer vision with models like ViT [91] and Swin Transformer [92]. BERT [89], as a specific example, represents a list of stacked encoders within the Transformer architecture [88]. To acquire bidirectional representations, the model is pre-trained on extensive unlabeled text data. BERT is offered in two versions: $BERT_{BASE}$ and $BERT_{LARGE}$, each differing in terms of layer count, hidden size, attention heads, and parameters.

A Transformer-based architecture was devised by researchers in [56] for skeleton-based fall detection. Additionally, in [68], the authors employed $BERT_{BASE}$ [89] to extract features from skeleton sequences, enabling activity recognition, including the detection of falls.

### K. GRAPH NEURAL NETWORK (GNN)

The human skeleton can be naturally represented as a graph, where nodes represent joints and edges represent the bones that connect these joints. Graph Neural Networks (GNNs) [35] are a type of NN that can be used to process graph data. Formally, a graph $\mathbf{G}$ can be represented as follows:

$$\mathbf{G} = (\mathbf{A}, \mathbf{X}, \mathbf{E}), \tag{3}$$

where the adjacency matrix $\mathbf{A}$ captures the connections between nodes, while node embeddings $\mathbf{X}$ and edge embeddings $\mathbf{E}$ represent the feature vectors of nodes and edges, respectively.

A common type of GNN is Convolutional GNNs (ConvGNNs)). In ConvGNNs, each node is updated by aggregating information from its neighbors. A $k^{th}$ layer $\mathbf{H}$ of ConvGNNs can be represented as follows:

$$\mathbf{H}_k = \mathbf{f}(\mathbf{H}_{k-1}, \mathbf{A}, \theta_k), \tag{4}$$

where function $\mathbf{f}$ with parameters $\theta_k$ takes the node embeddings $\mathbf{H}_{k-1}$ of the previous layer, aggregate information from each node's neighbors using node relationship information from adjacent matrix $\mathbf{A}$, and outputs the node embeddings $\mathbf{H}_k$ of the current layer. ConvGNNs can capture the important structural information of the graph data by exchanging information between each node and its neighbors. There are many variants of ConvGNNs including Graph Convolutional Network (GCN) [93]. It is possible to learn both spatial and temporal patterns in graph data, such as changes over time in a human skeleton. The Spatial-Temporal Graph Convolution Network (ST-GCN) [94] is a typical method.

Among the reviewed studies, the authors in [66] implemented a GCN-based architecture, while the authors in [87] proposed a GCN-based architecture with self-attention. ST-GCN [94] was used in both systems [67] and [72]. In [67],

ST-GCN served as a key component of the multimodal system, while in [72], skeletal features were extracted using a pre-trained ST-GCN model for an AE-based fall detection system, employing reconstruction error analysis.

## IV. SKELETON-BASED FALL DETECTION IN RGB VIDEOS

In this section, we examine all the fundamental steps of skeleton-based fall detection system in RGB videos, including data collection, keypoints extraction, feature extraction, fall recognition, and performance evaluation. Fig. 1 shows the general pipeline for skeleton-based fall detection.

First, RGB videos related to fall/non-fall events are collected and processed using pose estimation frameworks to extract skeleton data, which undergo data preprocessing to obtain processed skeleton samples. Handcrafted features can be extracted from these samples for recognition using rule-based or ML-based methods. Alternatively, a deep NN can be trained on these samples to perform deep feature extraction, followed by a recognition process using FC layers with Softmax activation function at the output layer. Finally, performance evaluation is done by splitting the skeleton samples into sets (e.g., train or test) and using the designed metrics (see Section IV-F).

Table 4 summarizes all the skeleton-based fall detection methods reviewed in this work with details for each pipeline step. Unless otherwise stated, the methods in this table utilize 2D keypoints by default.

### A. DATA COLLECTION

The first step in building a fall detection system is data collection. Since the scope of this work is to review the skeleton-based fall detection methods for RGB videos, we discuss only this type of videos generated by conventional cameras, leaving out depth videos recorded by depth cameras such as Microsoft Kinect [17]. In Table 2, we summarize the most popular fall datasets.

- **Multiple cameras Fall Dataset (MultiCam)** was generated using eight Internet Protocol (IP) cameras mounted on the ceiling to record 24 scenarios, 22 of which contain a fall along with other distracting events, while the last two contain only distracting events [95]. This dataset consists of a range of activities of daily living (ADL), including walking in various directions, performing actions like falling (e.g., sitting down, standing up, crouching down), while simulated falls include those that occur forward, backward, while seated incorrectly, or when a person loses balance.
- **Le2i Fall Detection Dataset (Le2i)** aims to produce realistic video sequences with different environmental conditions including four types of simulated locations (Home, Office, Coffee Room, and Lecture Room) with varied illumination and occlusions [96]. There are 143 fall sequences and 48 sequences of ADL.
- **UR Fall Detection Dataset (URFD)** was constructed using two Microsoft Kinect cameras, one parallel to the door and one on the ceiling, to record 70 video sequences, consisting of 30 falls and 40 ADLs [97]. Additionally, the corresponding accelerometer data was also collected.
- **UP-Fall Detection Dataset (UP-Fall)** was collected using multiple types of sensors, including brain sensors, accelerometers, and two Microsoft LifeCam Cinema Cameras with two viewpoints (CAM1-lateral/side view, and CAM2-frontal view) [98]. It includes 17 individuals between the ages of 18 and 24 who attempt three times at each of 11 activities, including falls. There are five different types of falls: falling forward with hands, falling forward with knees, falling backwards, falling sideward, falling while sitting in an empty chair. Standing, jumping, sitting, laying, walking, and picking up objects are types of ADLs.

As falls are rare events, the number of fall videos is smaller than that of ADL videos. To overcome this challenge, researchers have generated synthetic fall videos and combined them with existing RGB videos to train fall detection models [56]. Furthermore, the current datasets face limitations in terms of both quantity and quality, as discussed in Section V-A.

### B. KEYPOINT EXTRACTION

#### 1) POSE ESTIMATION FRAMEWORKS

Human Pose Estimation (HPE) is the task of identifying the location of human body joints [106], [107], [108]. HPE can be categorized into two types: 2D and 3D pose estimation. 2D pose estimation involves finding the 2D coordinates $(x, y)$ of the body joints, while 3D pose estimation determines their 3D positions $(x, y, z)$ in the image. The output of pose estimation models typically contains a set of joints, also known as keypoints, represented by their 2D or 3D coordinates, along the confidence scores *conf _score* representing the likelihood that these keypoints are predicted correctly.

The classification of HPE can be extended based on the number of individuals appearing in the image, leading to single-person and multi-person pose estimation. In the case of multi-person pose estimation from a monocular camera, two major approaches are utilized including top-down and bottom-up. These approaches have been thoroughly reviewed in comprehensive studies [106], [107], [108]. In general, top-down methods begin by identifying the bounding boxes for each individual and then performing single-person pose estimation on each of them. Bottom-up approaches, in contrast, start with the prediction of the body joints of all individuals in the image, and subsequently group the joints that correspond to each person. In terms of performance, top-down approaches generally surpass bottom-up approaches, as indicated in [108]. However, they tend to be computationally expensive due to the linear increase in detection time with the number of individuals in the scene. On the other hand, bottom-up approaches are more computationally efficient because their runtime remains almost unaffected by the number of individuals present [26]. Moreover,

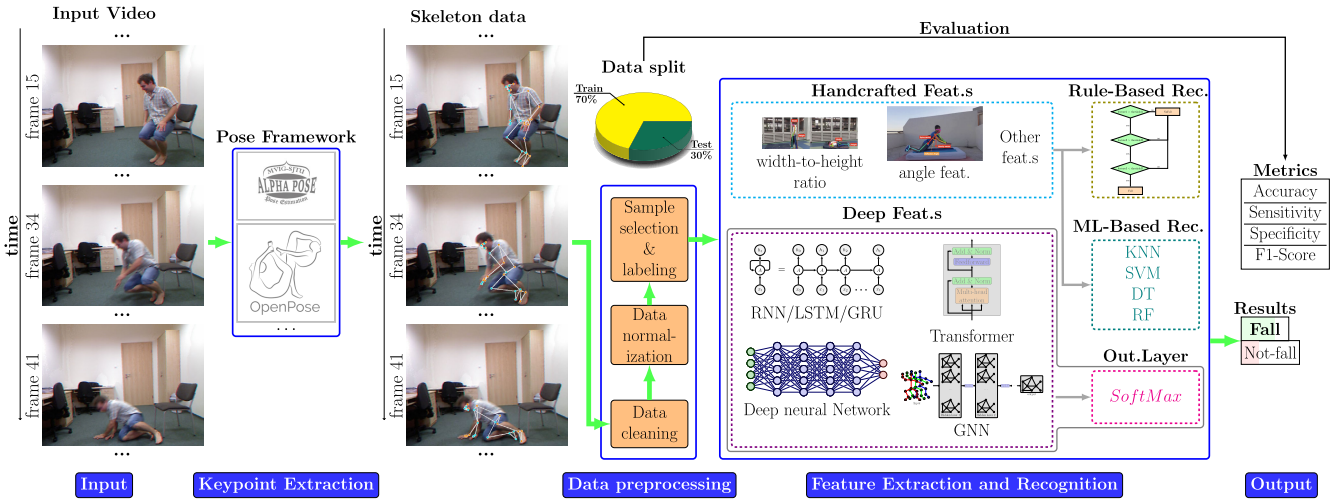



**FIGURE 1.** General pipeline of skeleton-based fall detection systems. Rec. = Recognition, Feat. = Feature, ML = Machine Learning, Pose Framework = Pose Estimation Framework.

**TABLE 2.** An overview of popular fall datasets with RGB video data. Cam. = Camera, IP = Internet Protocol, No. = Number, Viewp. = Viewpoint, Res. = Resolution, fps = Frame per second, Cond. = Data collection condition, Sim. = Simulated, Loc. = Location, Occ. = Occlusion, Act. = Activity, ADL = Activity of Daily Living.

| Dataset | Year | Camera Settings | | | Subject | | Scenario | | | | | No. of Data Sequence | | Access |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *RGB Cam. Type* | *No. Cam./ Viewp.* | *Video Res.* | *No. Sub-ject* | *Age Range* | *Cond.* | *Loc.* | *Occ.* | *No. of Act. Types* | | *Fall* | *ADL* | |
| | | | | | | | | | | *Fall* | *ADL* | | | |
| Multiple cameras Fall Dataset (MultiCam) [95] | 2010 | IP Cam. | 8 | 720×480, 30 fps | 1 | - | Sim. | Home | Yes | - | - | 24 × 8 | | http://www.iro.umontreal.ca/~labimage/Dataset/ |
| Le2i Fall Detection Dataset (Le2i) [96] | 2013 | - | 1 | 640×480, 25 fps | 9 | - | Sim. | Home, Office, Coffee Room, Lecture Room | Yes | 3 | 6 | 143 | 48 | https://imvia.u-bourgogne.fr/en/database/fall-detection-dataset-2.html |
| UR Fall Detection Dataset (URFD) [97] | 2014 | Microsoft Kinect | 2 | 640×640, 30 fps | 5 | > 26 | Sim. | Home, Office | No | - | - | 30 | 40 | http://fenix.univ.rzeszow.pl/mkepski/ds/uf.html |
| UP-Fall Detection Dataset (UP-Fall) [98] | 2019 | Microsoft LifeCam Cinema | 2 | 640×480, 18 fps | 17 | [18-24] | Sim. | Lab | No | 5 | 6 | 255 | 306 | https://sites.google.com/up.edu.mx/har-up/ |

bottom-up approaches exhibit better performance in crowded scenes with occlusion and complex poses, as highlighted in [109], [110].

For fall detection, most works use pretrained or open-source frameworks, as listed in Table 3, to extract 2D/3D keypoints in RGB videos. Fig. 2 shows the output of OpenPose [26] with 25 human keypoints.

- **OpenPose** is a real-time multi-person 2D HPE framework that follows the bottom-up approach [26]. OpenPose offers multiple pretrained models, each trained on different datasets, which generate either 15, 18, or 25 2D keypoints for each person in the image [112], and it can be run on various operating systems including Windows, macOS, and Linux. OpenPose can be executed using either CPU or GPU for optimal performance. Besides, OpenPose incorporates a supplementary module for 3D pose estimation. Nonetheless, this module has limitations as it is restricted to single-person scenarios, requires multiple stereo cameras, involves a complex calibration process, and lacks active support or maintenance [113]. Many works in this survey use OpenPose to extract 2D keypoints from the RGB videos due to its real-time performance. To make it even faster to be able to run on edge computing devices (e.g., the NVIDIA Jetson TX2), the MobileNetV2 architecture was used to replace the VGG model [114] to produce

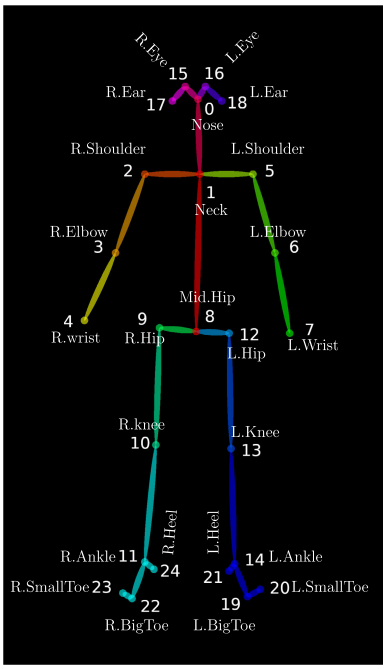



**FIGURE 2.** Detected keypoints/joints with their 25 corresponding body parts by OpenPose [26], [111].

**TABLE 3.** Pose estimation frameworks for keypoint extraction in RGB videos. Kp. Out. = Keypoint Output; Appr. = Approach, TD = Top-down, BU = Bottom-up.

| Framework | Kp. Out. | Appr. | Access |
|---|---|---|---|
| AlphaPose [99] | 2D/3D | TD | https://github.com/MVIG-SJTU/AlphaPose |
| DCPose [100] | 2D | TD | https://github.com/Pose-Group/DCPose |
| MediaPipe Pose [101] | 2D/3D | TD | https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md |
| MoveNet [102] | 2D | BU | https://github.com/tensorflow/tfjs-models/tree/master/pose-detection/src/movenet |
| MSPN [103] | 2D | TD | https://github.com/megvii-research/MSPN |
| OpenPifpaf [104] | 2D | BU | https://github.com/openpifpaf/openpifpaf |
| OpenPose [26] | 2D/3D | BU | https://github.com/CMU-Perceptual-Computing-Lab/openpose |
| PoseNet [105] | 2D | BU | https://github.com/tensorflow/tfjs-models/tree/master/pose-detection/src/posenet |

a lightweight version of OpenPose in some works [60], [61]. In addition, the authors in [60] attempted to enhance the performance of OpenPose by training it with a new augmented dataset, utilizing data augmentation methods such as flipping, clipping, and random scaling, to address the issue of missing fall motion samples.

- **AlphaPose** is another multi-person 2D pose estimation framework that utilizes the top-down approach based on the RMPE algorithm [99], [115]. AlphaPose can detect 17 keypoints in COCO format [116] or 16 keypoints in MPII format [117] for each person in the image. It can also be run on a CPU or GPU, but it only supports Windows and Linux. A recent update to AlphaPose in 2022 presented a fresh capability for estimating 3D poses. Nevertheless, the associated documentation and inference code necessary to utilize this feature remain inaccessible at present [118].
- **OpenPifpaf** is a novel bottom-up framework that first attempts to solve 2D pose estimation and pose tracking simultaneously [104].
- **MoveNet** is a highly precise pose estimation model that can detect 17 2D keypoints at above the real-time speed (30+ FPS) on the majority of modern laptops and PCs [102].
- **PoseNet** is a real-time human pose estimation model that can run in a web browser [105]. It can detect multiple poses, and each pose contains 17 2D keypoints.
- **MediaPipe Pose** is a lightweight CNN-based module of Google MediaPipe [119] that is optimized for single-person body pose detection and tracking [101]. It offers real-time performance on mobile phones, desktops, laptops, and web browsers, and can detect 33 keypoints [120] in 2D (using BlazePose [121]) and 3D (using BlazePose GHUM [122]).
- **DCPose** is a multi-person 2D human pose estimation framework that utilizes the previous frames and the next frames to improve the accuracy of the pose estimation in the current frame [100].
- **MSPN** is a top-down, multi-stage 2D pose estimation network consisting of two single-stage modules [103]. Through a cross-stage feature aggregation strategy, information loss during upsampling and downsampling is minimized. A coarse-to-fine supervision strategy is also implemented, which generates different ground-truth heat maps for different stages to improve localization accuracy.

### 2) DISCUSSION

Various frameworks support extracting 2D keypoints as shown in Table 3. According to Table 4, OpenPose is the most used framework for 2D keypoint extraction in the literature, likely due to its real-time performance. However, a recent study [56] evaluated the performance of various pose estimation frameworks, including OpenPose, AlphaPose, OpenPifpaf, MoveNet, and DCPose. The authors found that AlphaPose demonstrated the highest level of accuracy among human pose estimators in surveillance settings and yielded the best performance for skeleton-based fall detection.

For 3D keypoint extraction, MediaPipe Pose [101] is a widely employed framework in the literature,

**TABLE 4. Summary of representative methods for skeleton-based fall detection in RGB videos.**

| Appr. | Method | Kp. Extraction | | Data Preparation | | | | Feature Extraction | Recognition | | Evaluation | | Remark |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Frwk | No. Kp. | Clean. | Norm. | Sample Selection: Sample Type | Sample Selection: Label Scheme | | Type | Method | Dataset | Data Split | |
| Direct input | Serpa *et al.* 2020 [55] | AlphaPose, OpenPose, PoseNet | 15, 17, 18, 25 | IFRM | - | F | F | direct kp.s as input | ML | MLP (No. of hidden layers varies from 0 to 16, with 128 neurons in each layer) | URFD (Front) | train/test: 75/25 | Only consider fallen and no fall (No falling state) |
| | Ramirez *et al.* 2021 [41] | AlphaPose | 17 | IFRM | - | F | F | direct kp.s as input | ML | KNN, MLP, RF, SVM | UP-Fall (CAM1) | 10-fold cv with train/test: 70/30 | |
| | Ramirez *et al.* 2022 [46] | AlphaPose | 17 | IFRM | - | WD: $W = 36$ (2 sec), $S = 1$, use only 3 frames in $W$ | MOST | direct kp.s of 3 frames in a sample window (first, middle, and last frame) | ML | RF, SVM, MLP, AdaBoost | UP-Fall (CAM1) | 10-fold cv with train/test: 70/30 | |
| Hand-crafted | Alaoui *et al.* 2019 [39] | Custom | 15 | ConfRM | - | V | V | kp. angles and kp. distances, followed by PCA | ML | DT, KNN, RF, SVM | Le2i, URFD | - | |
| | Chen *et al.* 2020 [127] | OpenPose | 15 | - | - | - | - | speed of descent at center of hip joints (SDCH), angle between center line and ground (ACLG), with-to-height ratio (WHR) | R | A decisive process based on thresholds | Custom (10 subjects - 60 fall/40 non-fall videos) | - | |
| | Wang *et al.* 2020 [47] | OpenPose | 18 | - | - | WD: $W = 20$, $S = 1$ | - | Centroid Drop Rate (CDR), Upper Limb Velocity (ULV), 04 parameters of body enclosing external ellipse (PEE) | ML + R | Channel 1: MLP, Channel 2: RF, then Rules | Le2i,URFD | 10-fold cv | |
| | Dentamaro *et al.* 2021 [42] | OpenPose | 15 | MLI | - | V | V | spatio-temporal features , sigma-lognormal features, then statistical measures | ML | KNN, SVM with Gaussian Kernel (SVC), RF | URFD, Le2i | 10-fold cv | |
| | Liu *et al.* 2021 [43] | OpenPose | 25 | UKpRM | - | F | F | spine ratio, height-to-width ratio (HWR) , distance features, acceleration features, deflection features | ML | KNN, SVM, Boosting | Custom (240 fall/1267 ADL) | train/test: 80/20 | |
| | Zhang *et al.* 2022 [128] | OpenPose | 25 | - | - | - | - | HWR, angle of human spine inclination $\theta_1$, angle of right knee inclination $\theta_2$, angle of left knee inclination $\theta_3$ | Rule | threshold-based | URFD, Self-built | train/val /test: 50/25/15 | |
| DL | Hasan *et al.* 2019 [58] | OpenPose | 25 | UKpRM | MidHip Norm. | WD: $W = 24$, $S = 8$ | ALO | stacked 2-layer LSTM | ML | Softmax | MultiCam, URFD | train/test: 80/20 | Whole video splited into 24-frames clips, any fall clip detected then video fall |

Abbreviations/Notes:

- **Header rows:** Appr. = Approach, Frwk. = Framework, Kp. = Keypoint, No. = Number; No. Kp. = Number of Keypoints; Data Preparation: Clean. = Data Cleaning, Norm. = Data Normalization.
- **Kp. Extraction**: The extracted keypoints were in 2D format by default, unless otherwise specified.
- **Data Preparation**: ▸ **Clean.**: IFRM = Invalid frame removal, ConfRM = $conf\_score$ removal, UKpRM = Unnecessary Keypoint removal, MLI = fill missing keypoints by Linear Interpolation. ▸ **Norm.**: MidHip Norm. = Normalize by Mid.Hip keypoint, VRes Norm. = Normalize by video resolution, Min-Max Norm. = Min-Max Normalization, RP-Norm. = Relative Position Normalization, BP-Norm = Bone Parallel Normalization.
- **Sample selection**: ▸ **Type**: F = frame-based sample, V = whole-video sample, WD = window-based sample ($W$ = window size, $S$ = stride). ▸ **Label scheme**: MOST = Most repeated label, ALO = At least one, V = depend on video label, F = depend on frame label.
- **Recognition**: ▸ **Type**: ML = Machine-learning-based, R = Rule-based.
- **Evaluation**: ▸ **Data split**: cv = cross-validation.

**TABLE 4.** *(Continued.)* **Summary of representative methods for skeleton-based fall detection in RGB videos.**

| Appr. | Method | Kp. Extraction | | Data Preparation | | | | Feature Extraction | Recognition | | Evaluation | | Remark |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Frwk | No. Kp. | Clean. | Norm. | Sample Selection | | | Type | Method | Dataset | Data Split | |
| | | | | | | Sample Type | Label Scheme | | | | | | |
| DL | Jeong *et al.* 2019 [82] | OpenPose | 15 | MLI | VRes Norm. | WD: $W = 10$, $S = 1$ | MOST | Human Center Line Coordinate (HCLC) + Speed of HCLC (SHCLC) fed into a stacked 2-layer LSTM | Rule | Rule base on classification of LSTM | URFD | - | |
| | Lin *et al.* 2020 [59] | OpenPose | 25 | MLI, UKpRM, ConfRM | Min-Max Norm., RP-Norm. | WD: $W = 100$ | - | RNN, or LSTM, or GRU (1 layer) | ML | Softmax | Mixed dataset (URFD + Fall detection [150]) | train/test: 80/20 | Dataset contains a total of 1140 groups, each group 100 frames |
| | Chang *et al.* 2021 [60] | OpenPose, OpenPose-Light | 18 | - | - | WD: $W = 16$ | - | LSTM | ML | Softmax | MultiCam, Le2i, URFD | - | Pre-filtering reduces detection model runtime by filtering non-fall events. |
| | Galvao *et al.* 2021 [72] | OpenPose | 18 | - | - | - | - | ST-GCN features => Autoencoder => Reconstruction Error (RE) | Rule | Compare RE with a calculated threshold | URFD, UP-Fall | train/valid /test: 95/5/5. Train & valid sets consist only ADL data | Test on: single dataset, cross-dataset, and joint dataset |
| | Huu *et al.* 2021 [61] | OpenPose | 18 | IFRM | - | WD | - | LSTM | ML | Softmax | Mixed dataset (Self-built + MPII [117]) | varied depend on label | Labels: lying, crouching, sitting, standing, walking, punching, and falling |
| | Chen *et al.* 2022 [76] | LPN [123], Pose Lifting [124] | 25 | - | BP-Norm. | WD: $W = 300$ | - | 1D CNN (Dilated Convolution + Residual Connection) | ML | 1D Conv. (Output size = 2) | NTU RGB + D [151] | - | The dataset has nearly 57,000 videos (60 action classes), including falls |
| | Inturi *et al.* 2022 [62] | AlphaPose | 17 | IFRM | - | WD: $W = 64$, $S = 48$ | - | CNN + LSTM | ML | Softmax | UP-Fall | - | |
| | Juraev *et al.* 2022 [56] | OpenPose, Alpha-Pose, OpenPif-paf, MoveNet, DCPose | 17, 18 | - | Min-Max Norm. | WD: $W = 25$ | - | LSTM, or Transformer-based model | ML | MLP | Mixed Dataset (AIHub, Synthetic generated from KIST SynADL [152] | train/test: 75/25 | Labels: falling down, standing up, lying down, walking |

Abbreviations/Notes:

- **Header rows:** Appr. = Approach, Frwk. = Framework, Kp. = Keypoint, No. = Number; No. Kp. = Number of Keypoints; Data Preparation: Clean. = Data Cleaning, Norm. = Data Normalization.
- **Kp. Extraction**: The extracted keypoints were in 2D format by default, unless otherwise specified.
- **Data Preparation**: ▸ **Clean.**: IFRM = Invalid frame removal, ConfRM = $conf\_score$ removal, UKpRM = Unnecessary Keypoint removal, MLI = fill missing keypoints by Linear Interpolation. ▸ **Norm.**: MidHip Norm. = Normalize by Mid.Hip keypoint, VRes Norm. = Normalize by video resolution, Min-Max Norm. = Min-Max Normalization, RP-Norm. = Relative Position Normalization, BP-Norm = Bone Parallel Normalization.
- **Sample selection**: ▸ **Type**: F = frame-based sample, V = whole-video sample, WD = window-based sample ($W$ = window size, $S$ = stride). ▸ **Label scheme**: MOST = Most repeated label, ALO = At least one, V = depend on video label, F = depend on frame label.
- **Recognition**: ▸ **Type**: ML = Machine-learning-based, R = Rule-based.
- **Evaluation**: ▸ **Data split**: cv = cross-validation.

**TABLE 4.** *(Continued.)* **Summary of representative methods for skeleton-based fall detection in RGB videos.**

| Appr. | Method | Kp. Extraction | | Data Preparation | | | | Feature Extraction | Recognition | | Evaluation | | Remark |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Frwk | No. Kp. | Clean. | Norm. | Sample Selection: Sample Type | Sample Selection: Label Scheme | | Type | Method | Dataset | Data Split | |
| DL | Lau *et al.* 2022 [63] | MediaPipe Pose | 33 (2D & 3D) | IFRM | UKpRM (2D kp), R.Norm. (3D kp) | WD: $W = 35$ | - | 2D: gait speed, with-to-height ratio (WHR) + 3D: trunk angle => 1D CNN, (Attention-based) LSTM/GRU | ML | Softmax | Mixed dataset (MultiCam, URFD, and Fall detection dataset [150]) | train/val /test: 70/15/15 | Labels: non-fall, pre-impact fall (before hitting the floor), and fall (after hitting the floor) |
| | Li *et al.* 2022 [57] | YoloV3 [153], MSPN [103] | 17 | ConfRM | - | WD: $W = 30$ | label of last frame | Adaptive keypoint attention module and a stacked 3-layer RLSTM (A modifed version of LSTM using residual connection [22]) | ML | MLP | URFD, Le2i, Self-built | train/test: 70/30 | |
| | Suarez *et al.* 2022 [64] | MediaPipe Pose | 33 (3D) | - | MidHip Norm. + Value scaling | WD: $W = 16$ | - | keypoints, distance features, velocity features => 1D CNN | ML | Softmax | URFD, UP-Fall | URFD: train/test = 80/20. UP-Fall (CAM1): 2 first trials to train, the last for test. | UP-Fall (CAM1) has 3 action trials/subject |
| | Yadav *et al.* 2022 [65] | OpenPose | 25 | ConfRM | Min-Max Norm. | WD: $W = 45$, $S = 9$ | - | 1D CNN + GRU | ML | Softmax | UP-Fall, Self-built | train/valid /test: 70/15/15 | |
| | Yuan *et al.* 2022 [87] | AlphaPose | 17 | - | VRes Norm. | WD: $W = 20$ | | kp. values + kp. velocities => GCN + self-attention (Spatial + Temporal) | ML | Softmax | Le2i | - | |
| | Zahan *et al.* 2022 [66] | OpenPose | 25 | IFRM | Zero-mean and unit variance | WD: $W = 300$ ($W = 1145$ for UP-Fall dataset) | - | Spatial Graph Convolutional Network (SGCN) + Separable Temporal Convolutional Network (Sep-TCN) | ML | Softmax | URFD, UP-Fall | URFD: train/test = 70/30. Up-Fall: 5-fold cv (Cross-Fall and Cross-Trial) | Random masking during training: spatial joints masking and temporal frames masking |
| | Amsaprabhaa *et al.* 2023 [67] | OpenPose | 25 | UKpRM, MLI | Min-Max Norm., RP-Norm. | WD | - | ST-GCN [94] + 1D CNN | ML | Softmax | URFD, Self-built (190 fall/210 non-fall videos) | train/test: 70/30 | |
| | Ramirez *et al.* 2023 [68] | AlphaPose | 17 | IFRM | - | WD: $W = 36$ (2 sec), $S = 1$, use only 3 frames in $W$ | MOST | 12-layer BERT (Base version) [89] | ML | Softmax | UP-Fall (CAM1) | train/val /test: 70/15/15 | Multi-class: 11 labels (5 falls, 6 ADLs) from Up-Fall dataset and an unknown label. |

Abbreviations/Notes:

- **Header rows:** Appr. = Approach, Frwk. = Framework, Kp. = Keypoint, No. = Number; No. Kp. = Number of Keypoints; Data Preparation: Clean. = Data Cleaning, Norm. = Data Normalization.
- **Kp. Extraction**: The extracted keypoints were in 2D format by default, unless otherwise specified.
- **Data Preparation**: ▸ **Clean.**: IFRM = Invalid frame removal, ConfRM = $conf\_score$ removal, UKpRM = Unnecessary Keypoint removal, MLI = fill missing keypoints by Linear Interpolation. ▸ **Norm.**: MidHip Norm. = Normalize by Mid.Hip keypoint, VRes Norm. = Normalize by video resolution, Min-Max Norm. = Min-Max Normalization, RP-Norm. = Relative Position Normalization, BP-Norm = Bone Parallel Normalization.
- **Sample selection**: ▸ **Type**: F = frame-based sample, V = whole-video sample, WD = window-based sample ($W$ = window size, $S$ = stride). ▸ **Label scheme**: MOST = Most repeated label, ALO = At least one, V = depend on video label, F = depend on frame label.
- **Recognition**: ▸ **Type**: ML = Machine-learning-based, R = Rule-based.
- **Evaluation**: ▸ **Data split**: cv = cross-validation.

as demonstrated in the works [63], [64]. This preference arises from the difficulty in using other 3D-pose-supported frameworks like OpenPose [26] and AlphaPose [99], as mentioned earlier. Notably, the authors in [63] utilized MediaPipe Pose to extract not only 3D keypoints but also 2D keypoints. Alternatively, another approach to obtain 3D keypoints involves converting 2D keypoints to 3D. In an example study [76], the authors employed the Lightweight Pose Network (LPN) [123] for 2D keypoints extraction and subsequently followed the conventional pose lifting pipeline [124] to acquire 3D keypoints.

Choosing an appropriate pose estimation framework in a skeleton-based fall detection system is important because it affects the performance and usefulness of the system. When selecting a framework, two key factors to consider are pose quality and inference speed, along with additional factors such as source code availability, ease of use and integration, and hardware specifications. In general, better pose quality leads to better action detection performance [125]. Consequently, the framework offering the highest pose quality should be chosen among the candidate options. Inference speed is also important because the system needs to process input data in real-time, and the pose estimation step is typically the most time-consuming step in the entire pipeline, as discussed in Section IV-G2b.

Therefore, the choice of the pose estimation framework should be based on the specific application requirements, with the goal of striking a balance between pose quality and inference speed. For example, in home environments with few people and a GPU-equipped system, AlphaPose [99] can be utilized due to its high pose quality and ability to run at approximately 20 FPS on a GPU [118]. In contrast, in public environments with numerous individuals and a resource-constrained device, MediaPipe Pose [101] can be employed since it offers acceptable pose quality while maintaining real-time inference speed, achieving a processing speed of approximately 18 FPS with the full model on the CPU of a Pixel 3 phone [126].

### C. DATA PREPROCESSING

#### 1) DATA CLEANING

After the keypoint extraction step, the data are refined to improve the efficiency of subsequent processing. Data cleaning involves two main steps: removing invalid or unnecessary data and handling missing data. Skeleton data is considered invalid if it belongs to a person who is not involved in the fall event [41], [62], [46], or if it lacks specific keypoints such as the neck, hip keypoints [61], [63], or if it contains wrong keypoint data such as the detection of other objects, such as a chair, as a human body [66]. Some keypoints that do not contribute to fall detection can be removed from the whole detected keypoint sets, as argued by the authors in [43], [58], [59], [67]. For example, the authors in [58] used only 8 keypoints (R.Knee, L.Knee, R.Hip, Mid.Hip, L.Hip, R.Shoulder, L.Shoulder, Neck - see Fig. 2) and removed other keypoints. Some works only use the location information of the keypoints and discard all *conf_score* values [39], [57], [59], [65]. To handle missing data, linear interpolation is typically employed [42], [82].

#### 2) DATA NORMALIZATION

Subsequently, the data are normalized to ensure a uniform scale for improving the convergence speed when training a model. Formally, the normalization process for a keypoint can be defined as follows:

$$P_{part\_norm} = \mathcal{N}(P_{part\_det}), \tag{5}$$

where $P_{part\_det}$ is the detected keypoint of a particular body part (see Fig. 2) by a pose estimation framework, $P_{part\_norm}$ is the corresponding normalized keypoint, and $\mathcal{N}$ is the normalization process.

There are several methods to normalize keypoints. Detected keypoints can be normalized by the location of a reference joint or by the video resolution. In [58], each keypoint is normalized by the hip joint as follows:

$$P_{part\_norm} = P_{part\_det} - P_{midhip\_det}, \tag{6}$$

where $P_{midhip\_det}$ is the detected keypoint of the Mid.Hip. In contrast, the authors in [82], [87] proposed to normalize each keypoint by the video resolution as follows:

$$P_{part\_norm} = (P^x_{part\_det}/W, P^y_{part\_det}/H), \tag{7}$$

where $P^x_{part\_det}$, $P^y_{part\_det}$ are $x$, $y$ coordinate values of detected keypoint; $W, H$ are the width, and height of the video frame, respectively.

Rescaling the skeleton data to the range of $[0, 1]$ was achieved in some studies [56], [59] through by using Min-Max Normalization (Min-Max Norm.). This method can be defined as follows:

$$v_{norm} = \frac{v - v_{min}}{v_{max} - v_{min}}, \tag{8}$$

where $v$ is the original value, $v_{min}$ and $v_{max}$ are the minimum and maximum values of the original data, respectively.

Additionally, the authors in [59] also proposed a new normalization method named Relative Position Normalization (RP-Norm) to normalize the keypoints data. In this RP-Norm method, each original keypoint $P_{part\_det}(x^f, y^f)$ in a particular frame $f$ is set to a new relative coordinate position $P_{part\_norm}(x^f_r, y^f_r)$ by using the center point of the frame $C(x^f_c, y^f_c)$, the keypoint of Mid.Hip $P_{midhip\_det}(x^f_{midhip}, y^f_{midhip})$ to calculate the displacement distance $D(x^f_{dis}, y^f_{dis})$ as described in the following equations:

$$P_{part\_det} = (x^f, y^f) \tag{9}$$

$$C(x^f_c, y^f_c) = (W/2, H/2) \tag{10}$$

$$D(x^f_{dis}, y^f_{dis}) = (x^f_{midhip} - x^f_c, y^f_{midhip} - y^f_c) \tag{11}$$

$$P_{part\_norm}(x^f_r, y^f_r) = (x^f - x^f_{dis}, y^f - y^f_{dis}) \tag{12}$$

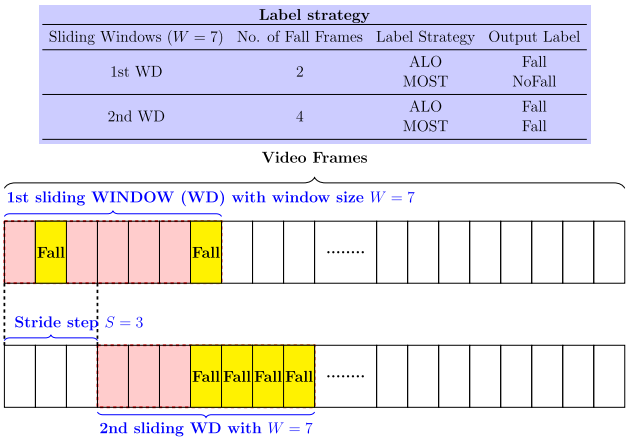



**FIGURE 3.** Sliding windows with different labeling strategies.

where $W$ and $H$ are the width and height of the video frame $f$, respectively. The authors in [66] proposed a method for processing skeleton data in which the spine joints are first moved to the origin to obtain consistent skeleton orientations, followed by a normalization process to ensure zero mean and unit variance.

3D keypoints can be normalized in several ways. The authors in [76] normalized 3D keypoints by applying centering, scaling, and rotation operations. Specifically, the 3D keypoints were centered, scaled, and rotated to achieve Bone Parallel Normalization (BP-Norm.) by aligning the hip-spine bone with the $z$-axis and the left-to-right shoulder bone with the $x$-axis. In [63], the normalization of 3D keypoints involved scaling them to fit within the range of $[-1, 1]$ for each dimension. Additionally, the hip joint served as the origin $(0, 0, 0)$. In [64], 3D keypoints were normalized by first calculating the mid hip keypoint using (13), then performing mid hip normalization as in (6). Finally, the normalized keypoints were scaled by dividing them by the pose size, which was calculated by multiplying the shoulder-hip distance by a certain factor.

### 3) SAMPLE SELECTION

After cleaning and normalizing the data, the next step is to select the samples for feature extraction. There are two types of sample formats: 1) **frame-based samples** and 2) **sequence-based samples**. A frame-based sample, as used in [41], [43], [55], consists of keypoints from a single frame. However, this format can lead to ambiguity in distinguishing between falls and ADLs when the subject is in certain positions, such as lying on a bed. The label for a frame-based sample is determined by manual frame-by-frame annotation.

Most works prefer sequence-based samples to extract features. Sequence-based samples can be divided into two categories:

- A **whole-video sample** is a sequence that contains the keypoints of all frames in a video [39], [42]. The label of a whole-video sample is marked as "fall" if there is a fall at any time in the video.
- A **window-based sample**: is a sequence that contains the keypoints of a particular number of frames or duration in a video [47], [56], [57], [58], [59], [60], [61], [62], [65], [66], [82], [46], [63], [64], [76]. This type of sample is generated using a sliding window technique with a fixed window size ($W$) and a fixed window stride ($S$). The unit of $W$ and $S$ is either number of frames or in seconds. The values of $W$ and $S$ vary among works. For example, [82] uses $W = 10$ frames and $S = 1$ frame, while [62] uses $W = 64$ frames and $S = 48$ frames. Noticeably, the selection of values for variables $W$ and $S$ lacks clear criteria or rules in the majority of reviewed works. In their study [46], the authors employed an exhaustive search method to determine optimal values for $W$. Their objective was to identify the specific value of $W$ that maximizes system performance, particularly in terms of recall rate. It is important to note that this approach is limited to a specific dataset and may not be generalizable. Different strategies are used to label the window-based sample:
  - **Most repeated label (MOST)**: the label of a window-based sample is determined by the majority label of the frames in the window [82]. For example, if the number of frames with fall label is greater than the number of frames with ADL label, then the label of the window-based sample is "fall".
  - **At least one (ALO)**: the label of a window-based sample is determined as "fall" by the presence of at least one frame with fall label in the window [58].
  - **Others**: Other labeling strategies may be employed, such as last frame labeling in [57] where the label of a window-based sample is determined by the label of the last frame in the window.

  The labeling strategies may not be described in the reviewed works, such as [47], [65]. Fig. 3 shows this window-based sampling process with two labeling strategies: MOST and ALO.

### 4) DISCUSSIONS

#### *a: ON DATA CLEANING*

The existing literature commonly employed heuristic techniques to eliminate invalid skeleton data for training. These techniques operated on the assumption that the main actor possesses the highest quality skeleton, allowing for the selection of valid skeleton data by identifying the skeleton with the highest confidence score [41] or the largest standard deviation over time [66] and discarding the rest. This assumption generally holds true in controlled environments or simulated datasets, where the camera focuses predominantly on the main actor, resulting in better skeleton data compared to other individuals in the scene. However, this approach may not be appropriate for real-world datasets, particularly in scenes with multiple people, because there is no guarantee that the main actor will possess the highest quality skeleton.

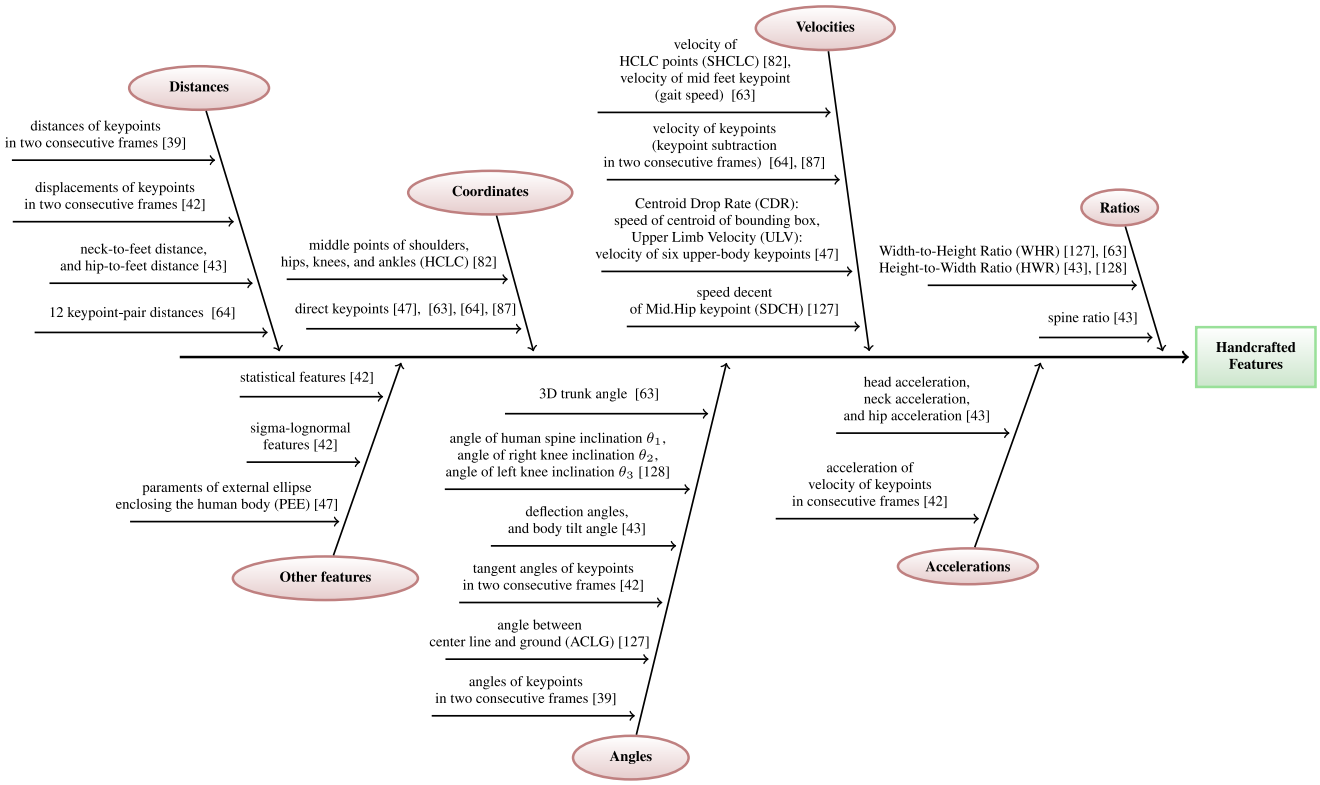



**FIGURE 4.** Categorization of handcrafted features in the reviewed works.

Consequently, utilizing these approaches in such cases may result in the removal of valid skeleton data, thereby reducing the available training data. Thus, alternative strategies for obtaining valid skeleton data should be developed.

Furthermore, the inclusion or exclusion of keypoint *conf_score* values during data cleaning lacks sufficient justification. For instance, the works [41], [62] chose to preserve these scores, while the studies [39], [57], [59], [65] removed them. However, none of these studies provided explicit reasons for their respective decisions. Consequently, the effect of including or excluding keypoint confidence scores on system performance was not investigated.

### b: ON NORMALIZATION

Data normalization generally improves classification performance [129], [130], [131]. However, in skeleton-based fall detection in RGB videos, there is a lack of description regarding skeleton data normalization in many studies (see *Norm* column in Table 4). While some studies provided details on normalization techniques [64], [65], [82], [87], they failed to analyze the impact of these techniques on system performance. The issue of whether normalization improves classification performance or establishes the superiority of one method over others remains unresolved in the current literature. For instance, in [59], RP-Norm improved classification, whereas Min-Max Norm surprisingly led to lower performance compared to unnormalized skeleton data. However, this conclusion is dataset-specific and may vary across different datasets.

Therefore, researchers are encouraged to explore various normalization methods, including those discussed in this survey, alternative normalization techniques (e.g., described in [131]), and those utilized in skeleton-based action recognition (e.g., mentioned in [132], [133]), to determine the approach that yields the best performance on their specific application data.

### c: ON SAMPLE SELECTION

The majority of the methods in Table 4 used a consistent approach for sample selection during both training and testing. However, an exception can be found in the work [58]. The authors employed a window-based sampling technique for training but adopted a different approach for testing. During testing, they divided the video into window-based samples, and if any of these clips were identified as a fall, the entire video was considered to contain a fall event. The testing strategy of using the entire video as test data can improve system performance, but it is not suitable for real-time applications.

To ensure real-time feasibility, an appropriate sample selection strategy must be chosen. Whole-video samples are not suitable because they require the entire video to be recorded before a decision can be made. Frame-based samples work in real-time but struggle to differentiate between falls and ADLs in specific positions like lying on a bed,

leading to false alarms. Window-based samples are highly suitable for real-time applications. They require a limited number of frames to make a decision and are less affected by the ambiguity found in frame-based sampling. In real systems, selecting an optimal window size $W$ is crucial. Large window sizes can cause detection delays due to the short duration of falls (e.g., reported as 2 seconds in [134]) and the requirement for the system to wait for the window to fill (e.g., $W = 300$ in [76]). Conversely, small window sizes may result in false alarms due to insufficient information for decision-making. Guidelines for window size selection can be found in [135], although they are not specific to fall detection. Testing the system's performance and speed with different window sizes is necessary to determine the optimal window size that achieves the desired performance while maintaining real-time speed requirements, as in [47]. Furthermore, in the inference step, the system can flexibly determine when to trigger an alarm, rather than relying solely on the detection of a single input sample. An example of this approach can be found in [82]. The authors in this work assigned weights to the predictions obtained from multiple window-based samples. Based on the weighted sum of these results, a final decision was made.

### D. SKELETAL FEATURE EXTRACTION

Once samples have been obtained, the subsequent task is to extract features from them. There are two primary methods for feature extraction: 1) handcrafted features and 2) DL features. It is worth noting that some studies have used skeleton data directly as input features for training a classifier, without any feature extraction process. For example, two studies [41], [55] both used this approach. The authors in [46] also used skeleton data directly, but they chose to use the data from three particular frames (the first, middle, and last frame) in a sample window, after performing an exhaustive search.

#### 1) HAND-CRAFTED FEATURES

The handcrafted features typically utilize spatial and temporal information of keypoints to compute various types of features including coordinate features, distance features, angle features, velocity features, acceleration features, ratio features, and some other features. Fig. 4 presents all the handcrafted features along with their categories used in the reviewed studies. For the sake of brevity, the handcrafted features described in this section were computed based on 2D keypoints, unless otherwise stated.

In [39], the authors computed feature vectors for each whole-video sample. Distance and angle features for each keypoint are computed based on its coordinate at frame $t$ and frame $t + 1$. This process was applied to all 15 keypoints over all frames of each video to obtain the feature vectors. The authors then employed Principal Component Analysis (PCA) [136] to ensure all feature vectors of varying length video to have a uniform length, as depicted in Fig. 5.

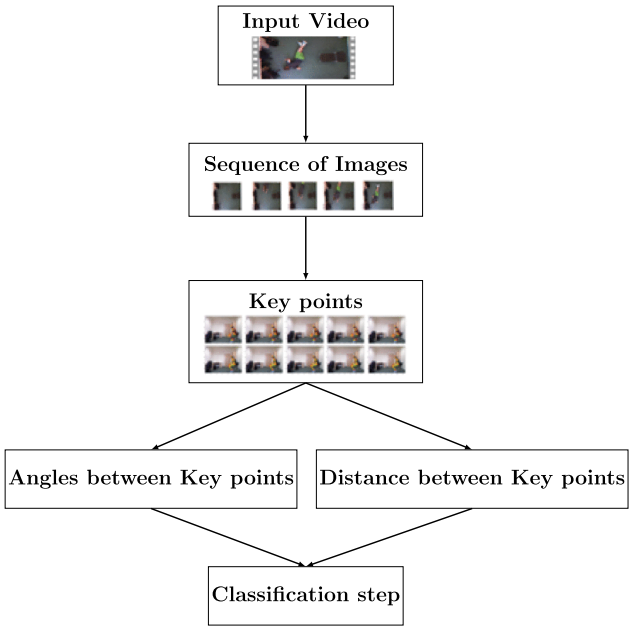



**FIGURE 5.** Computing angle and distance features in [39]. Adapted from [39] under the CC BY 4.0 license.

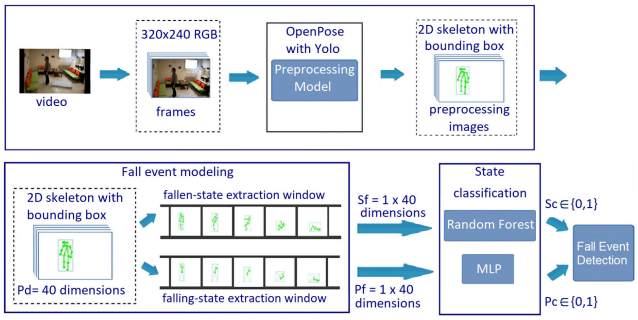


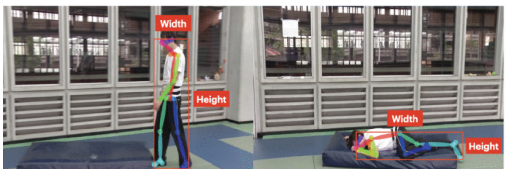

**FIGURE 6.** Dual channel integration in [47]. Adapted from [47] under the CC BY 4.0 license.

Some studies, such as [82], [127] computed the middle keypoint of a body part $P_{mid\_part}$ (see Fig. 2) by utilizing the left keypoint $P_{left\_part}$ and the right keypoint $P_{right\_part}$ of that body part for further processing with the following formula:

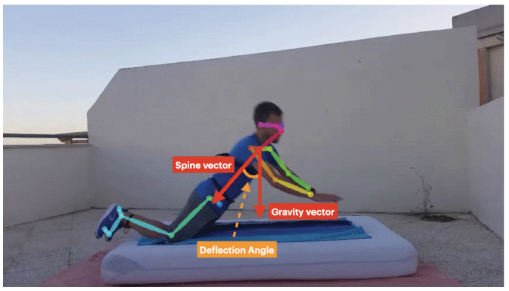
$$P_{mid\_part} = (P_{left\_part} + P_{right\_part})/2, \tag{13}$$

For instance, if the detected keypoints do not include the Mid.Hip keypoint due to the output format of the pose estimation framework used, this keypoint can be computed as $P_{mid\_hip} = (P_{left\_hip} + P_{right\_hip})/2$.

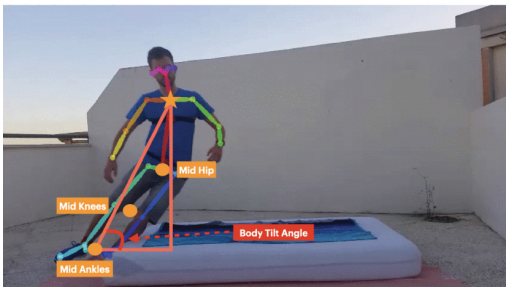
In [82], the authors introduced the Human Center Line Coordinate (HCLC) feature, consisting of four coordinates representing the middle points of the shoulders, hips, knees, and ankles, which can be calculated using (13). Additionally, the authors calculated the Speed of HCLC (SHCLC) as an additional feature by dividing the distance between the HCLC coordinates at two timestamps by the time interval between them.




(a) Width-to-Height Ratio




(b) Spine Deflection Angle




(c) Body Tilt Angle

**FIGURE 7. Some handcrafted features used in [43]. Adapted from [43] under the CC BY 4.0 license.**

For use in a rule-based fall detection system, the authors developed three types of handcrafted features in [127]: speed of descent at center of hip keypoints (SDCH), calculated in the same way as SHCLC, angle between center line and the ground (ACLG) calculated using the head keypoint and the middle point of the ankle, and the width-to-height ratio (WHR) for the bounding box of the human body.

In [47], the authors proposed two types of handcrafted features including falling-state features and fallen-state features, as shown in Fig. 6. The falling-state features consist of the Centroid Drop Rate (CDR), which are calculated by determining the average moving speed of the human body centroid (the diagonal intersection of the bounding box of a human body obtained using Yolo [75] and DeepSort [137]), and Upper Limb Velocity (ULV) by averaging the value of the movement velocity of six joints (L.Eye, R.Eye, Nose, Neck, L.Shoulder, and R.Shoulder) in the x-axis direction between two sampling frames. The fallen-state features consist of direct keypoints and four parameters of the external ellipse enclosing the human body (PEE) derived from keypoints, providing a more accurate description of the human body shape.

The authors in [42] applied the Kinematic Theory of Rapid Human movement and its sigma-lognormal model [138] to extract many features for fall detection. They computed 1120 features by calculating spatio-temporal features and sigma-lognormal features for all 14 keypoints in each video. The spatio-temporal features contain information of displacement, velocity, acceleration, and tangent angle of each keypoint in consecutive frames. On the other hand, the sigma-lognormal features consist of Lognormal stroke number and five different parameters for all lognormal strokes. After computing these features, the authors applied statistical measures, including mean, median, standard deviation, 1 percentile, and 99 percentiles, to obtain a final set of handcrafted features for fall detection.

In [43], the authors constructed several types of features (see Fig. 7 for illustration) including ratio features (spine ratio, which is the Neck-to-Mid.Hip distance divided by the L.Hip-to-R.Hip distance, and the height-to-width ratio (HWR) of the bounding box), distance features (neck-to-feet distance and hip-to-feet distance), acceleration features (acceleration of head, spine, and hip), and deflection features. The deflection features consist of:

- Deflection angles: the angles between gravity vector and six vectors formed by two keypoints including Neck-Mid.Hip, L.Hip-R.Hip, R.Hip-R.Knee, R.Knee-R.Ankle, L.Hip-L.Knee, and L.Knee-L.Ankle.
- Body tilt angle: the smallest angle among the angles between the horizontal vector $v$ parallel to the ground and the vectors formed by two keypoints including Neck-Mid.Hip, Neck-Mid.Knee, and Neck-Mid.Ankle.

Similarly, the authors in [128] calculated four feature values from human skeleton data: the HWR of the human bounding box, and three angle features including the angle of human spine inclination $\theta_1$, the angle of right knee inclination $\theta_2$, and the angle of left knee inclination $\theta_3$. These angles $\theta_1$, $\theta_2$, $\theta_3$ are the angles between the vector $v$ parallel to the ground and the three vectors formed by two specific keypoints (Neck-Mid.Hip, L.Knee-L.Ankle, and R.Knee-R.Ankle), respectively.

In [87], the authors used skeleton data to derive handcrafted features, which were then utilized as two input streams for a deep neural network to extract deep features. They computed two distinct types of features: 1) 2D coordinates over frames, and 2) the velocity of these coordinates. The velocity values were obtained by subtracting the current frame's coordinates from the corresponding coordinates in the previous frame.

3D keypoints were also used to extract handcrafted features. In [63], 2D skeleton data were used to compute the WHR feature, like [127]. Additionally, 26 values of $(x, y)$ representing 13 crucial 2D keypoints and gait speed were also extracted. The gait speed was calculated by dividing the walking distance, determined by the positional difference of the mid feet keypoints, by the time per frame. The 3D keypoints were used to compute the 3D trunk angle feature, which is the angle between the body trunk and the vertical plane that passes through the hip. The authors in [64] extracted three types of features from 3D skeleton keypoints, which had been

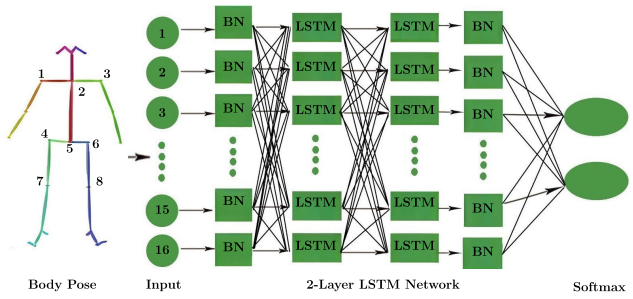



**FIGURE 8.** Deep features extracted using LSTM [58]. Adapted from [58] - Copyright 2019 IEEE.

normalized as described in Section IV-C2: 1) the normalized keypoints themselves, 2) 12 types of distance features, each calculated by subtracting the coordinates of two particular keypoints (e.g., shoulder-elbow), and 3) velocity features calculated by subtracting 3D keypoints in two consecutive frames.

### 2) DL FEATURES

The normalized skeleton data are typically fed into a NN to extract deep features. Noticeably, some works [63], [82] used handcrafted features as input to a deep neural network to extract deep features.

Various types of NNs have been employed for DL feature extraction, including popular ones such as CNNs and RNNs (including LSTM [79] and (GRU) [80] variants), as well as more modern architectures such as Transformer [88] and GNN [35].

In [58], 2D skeleton data extracted using OpenPose [26] were fed into a stacked 2-layer LSTM [79] with 40 LSTM cells to generate deep temporal features before connecting to the Softmax classifier as shown in Fig. 8. Additionally, to reduce the internal covariate shift problem, the authors used batch normalization (BatchNorm) [139] after the input layer and the RNN layers. Similarly, the authors in [60], [61] feed the processed skeleton data into LSTM to extract DL features for later recognition step.

In the same way, the authors in [59] began by normalizing keypoints and then filled in any missing data by linear interpolation. They then used RNN, LSTM [79], or GRU [80] (with one hidden layer) to learn the temporal representation of the skeleton data, obtaining the feature vector for classification using the Softmax layer.

A stacked LSTM network with 256 neurons in each layer was used by the authors in [82] to extract high-level features.

Unlike previous works, the authors in [62] utilized the CNN+LSTM architecture to extract deep features for classification. Specifically, they extracted skeleton data consisting of 51 values (17 keypoints $\times$ 3 values of $x$, $y$, and $conf\_score$) using AlphaPose [99]. These data were then fed into a CNN module to obtain spatial features, which were subsequently fed into an LSTM network to learn the temporal relationships and generate high-level features. These high-level features were transformed using a fully connected layer to capture nonlinear relationships before being connected to a Softmax layer for classification. Similarly, the author in [65] proposed to use CNN and GRU [80] to learn the deep features. The input data consists of keypoints from a sequence of 45 frames. The keypoints in each frame are processed through two blocks of 1D CNN, BatchNorm [139] and Max Pooling, followed by one block of 1D CNN, BatchNorm and GAP [74] before being fed into GRU [80]. The output of each GRU cell is then connected to two FC layers to produce the final features for classification.

1D-CNN networks have been employed for deep feature extraction. The authors in [76] employed a 1-CNN-based network to extract deep features from 3D skeleton data. They began by concatenating 3D keypoints from each 300-frame sample as input. These inputs were then passed through a 1D-CNN layer with a kernel size of 3, followed by four residual blocks. Each residual block consisted of two 1D-CNN layers: one dilated 1D-CNN [140] with a dilation factor of $3^k$ (where $k \in \{1, 2, 3, 4\}$, corresponding to the order of the residual block) and one 1D-CNN with a kernel size of 1. The dilated 1D-CNN was used to capture long-range temporal relationships while keeping the computational cost low. Finally, the output of the last residual block underwent average pooling to generate the deep features before classification process. In [64], the researchers utilized a 1D-CNN-based network to extract deep features from three types of handcrafted features: 3D keypoints, distance features, and velocity features. The network architecture consisted of two 1D-CNN layers, one global average pooling (GAP) layer, and two fully connected (FC) layers. The classification was performed using a Softmax layer. A dropout layer [141] and a ReLU activation layer [52] were applied after each 1D-CNN layer and the first FC layer to improve performance and prevent overfitting.

Attention mechanisms can be employed for fall detection. The authors in [57] proposed an adaptive keypoint attention module to assign varying importance weights to different keypoints for fall detection. Each input sample was constructed by extracting skeleton data from 30 frames. Each frame contained 17 keypoints with the $conf\_score$ removed, resulting in a sample with dimensions of $(17 \times 2 \times 30)$. After applying BatchNorm [139], the input sample was transposed to $(17 \times 30 \times 2)$ before entering the adaptive keypoint attention module. This module utilized GAP and GMP layers to obtain $T_{avg}$ and $T_{max}$, respectively, which were then concatenated and passed through another GAP to produce fused features. Two FC layers were applied to these fused features to produce attention weights for the keypoints. Next, these attention weights were multiplied with the input data to obtain weighted input data, which encodes the importance of each keypoint in the fall events. These weighted input data were then passed to a stacked 3-layer RLSTM to extract deep features before passing through a MLP module with four FC layers for classification. Note that RLSTM is a variant of LSTM that utilizes residual connections [22] to

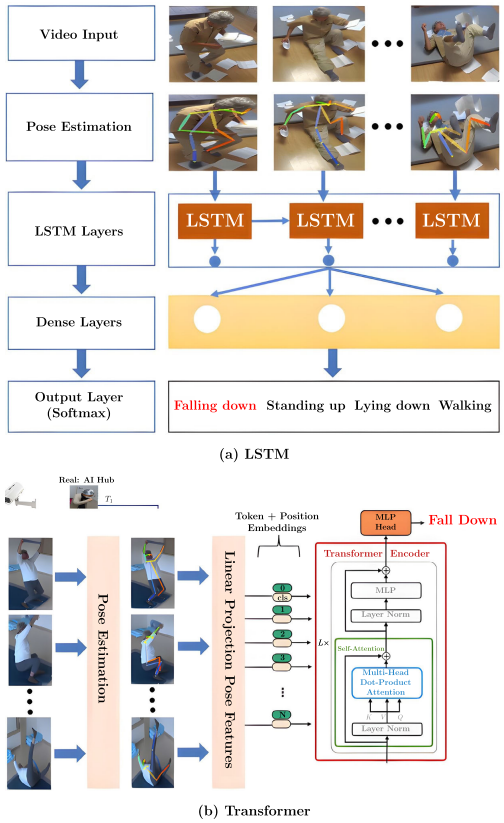



**FIGURE 9. Fall detection using (a) LSTM and (b) Transformer with skeleton data obtained from various pose estimators in [56]. Adapted from [56] under the CC BY 4.0 license.**

address the issue of information loss, which may arise due to the use of three hidden layers of LSTM in the proposed architecture. In another study [63], the authors employed pre-extracted handcrafted features, including WHR, 3D trunk angle, skeleton keypoints, and gait speed, as inputs for deep neural network architectures to extract deep features. The authors explored various network types, including 1D-CNN, LSTM [79], GRU [80], and LSTM/GRU with attention mechanism, which was achieved by incorporating a customized attention layer. Various fusion strategies were employed in this study, including feature fusion, model stacking, and score fusion. Feature fusion aimed to explore various combinations of input features, while model stacking involved the stacking of different sub-networks to create a larger network. Score fusion utilized prediction results from different models for majority voting. The results indicated that the best performance was achieved by model stacking using GRU and GRU with attention mechanism.

Transformer, a modern DL architecture [88], has also been utilized for fall detection using skeleton data. In [56], a Transformer-based model was developed to extract deep features from skeleton data. The model involved linearly projecting preprocessed keypoints onto a predefined Transformer encoder and combining them with class tokens and positional encoding to create input embeddings. These embeddings were then processed by the multi-head attention layers, followed by FC layers to extract deep features. Finally, the deep features were connected to the MLP head for action classification, including fall events. Noticeably, this study also compared the performance of the Transformer and LSTM models (see Fig. 9) and found that the Transformer-based model outperformed the LSTM-based model in terms of accuracy. The authors in [68], proposed another transformer-based model for activity recognition, including falls. They employed the $BERT_{BASE}$ [89] model, which has 12 layers, a hidden size of 768, 12 attention heads, and 110M parameters, to extract deep features from the skeleton input. Each skeleton input vector was constructed by concatenating the values of 17 keypoints from three particular frames: the first, middle, and last frames in the corresponding sample window. This concatenation process resulted in a vector of 153 values. The core idea behind their approach was to treat each skeleton input vector as if it were textual data. To achieve this, each value within an input vector was mapped to a corresponding token represented by an integer value ranging from 0 to 30,000, aligning with the vocabulary size of the $BERT_{BASE}$ model. These tokens were then fed into the $BERT_{BASE}$ model to extract the deep features. Finally, the extracted features were connected to a fully connected (FC) layer for the classification task. In addition, the authors also utilized TABGAN [142], a model derived from GAN (Generative Adversarial Networks) [143], to generate synthetic data. This strategic approach aimed to address the challenge of data imbalance within the UP-Fall dataset [98], ultimately resulting in improved performance of the detection system.

Human skeleton data can be viewed as a graph, where nodes represent joints and edges represent the bones connecting these joints. Therefore, GNN algorithms can be utilized for processing this data type for fall detection.

In [66], the authors proposed a GCN-based architecture (as shown in Fig. 10) to extract deep features from skeleton data. In more detail, joint and motion data in skeleton sequences were first projected into a higher dimensional space using an encoding layer, which included a BatchNorm [139] layer and $1 \times 1$ convolution, and then combined into a compact feature vector which can represent both joint and motion information. Next, the compact feature vector was then processed through two learning blocks, each consisting of a Spatial Graph Convolutional Network (SGCN) and a Separable Temporal Convolutional Network (Sep-TCN), to learn the deep features for classification. SGCN learned the spatial relationships in the compact feature vector by aggregating information from neighboring nodes with the help of the adjacency matrix, while Sep-TCN learned global temporal relationships using depthwise separable convolution [144]. Both SGCN and Sep-TCN used residual connections [22] in their design. Additionally, the authors proposed to use randomized masking by random removing keypoints in a frame (spatial masking) or random removing some frames in the sequence (temporal masking) as a regularization technique to improve the generalization of the model.

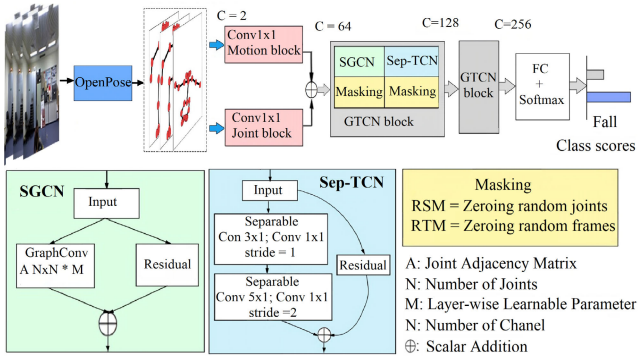



**FIGURE 10.** SDFA architecture [66]. Adapted from [66] - Copyright 2023 IEEE.

The authors of [67] used a feature fusion strategy to extract deep features by combining ST-GCN [94] and 1D CNN. They computed two sets of spatio-temporal kinematic gait features. The first feature set (F1) was generated using ST-GCN [94] to capture both spatial and temporal relationships between joints. To generate the second set ($F2$), the gait time series were first calculated by extracting seven gait features on each frame including:

- Gait speed: the distance between the L.Ankle and R.Ankle joints divided by the time interval.
- Six angles: the angles of the L.Hip, L.Knee, L.Ankle and the angles of R.Hip, R.Knee, R.Ankle.

Next, this gait time series was applied noise reduction before being fed into a multichannel 1D-CNN module, which consisted of series of convolutional layers and max pooling layers. The purpose of this was to model the temporal relationships between the gait features and generate the second feature set ($F2$). After being flattened to 1D, these two feature sets ($F1$ and $F2$) were combined as a single input for the second 1D-CNN module, which contained one convolutional layer and one average pooling instead of max pooling. The output of the second 1D-CNN was used as deep features for classification.

In the context of fall detection systems, it is noteworthy that a pretrained ST-GCN model [94] can be effectively utilized as a feature extractor. In [72], the authors proposed a fall detection framework based on anomaly detection using reconstruction error analysis [71]. First, a pretrained ST-GCN model [94] was employed to derive 256-dimensional vectors from 2D skeleton data extracted from input videos through OpenPose [26]. Then, these vectors were utilized as inputs to train three different types of models: AE models based on a neural network, models based on PCA [136], or models based on SVD (singular value decomposition) [145]. These models were exclusively trained on ADL data, which constituted 90% of the total dataset. Mean Squared Error (MSE) served as the loss function. The resulting output of the AE models corresponded to the reconstruction error, which was then compared to a predefined threshold to determine whether a fall event had occurred.

The integration of attention mechanisms in GCN-based models presents a feasible approach. In [87], hand-crafted features (keypoints and velocities in sample frames) were used as input streams. Each stream underwent a $1 \times 1$ convolutional layer to obtain the corresponding embeddings. These embeddings were combined and augmented with spatial embeddings, representing the keypoint types (e.g., hip or knee). Notably, the spatial relationships between keypoints were learned from data to produce a learnable adjacency matrix, rather than being predefined as in [94]. The embeddings and the learnable adjacency matrix were then processed by a GCN layer to produce immediate features. Subsequently, the immediate features underwent a Spatial Self-Attention module with multi-head attention, resembling the traditional self-attention [88], but operating on keypoint embeddings rather than words. This module captured interdependencies among keypoints within a frame. The output of this module was then combined with temporal embeddings, indicating frame orders, and passed through a Temporal Self-Attention module. This module utilized the self-attention mechanism [88] to capture the dependencies between each keypoint and its corresponding keypoint in other frames, enabling the model to capture temporal changes in the skeleton. Finally, the output of this module was fed into a Max pooling layer for extracting deep features used for classification.

### 3) DISCUSSIONS

Several handcrafted features have been developed based on information extracted from keypoints and their temporal changes, including keypoint coordinates, distances, angles, velocities, accelerations, ratios, and others. Surprisingly, most of the reviewed studies [43], [47], [127], [128] did not investigate the importance of each feature type in fall detection. According to some studies [42], [82], velocity and acceleration are considered more important than other features for fall detection. Although this conclusion may be influenced by the dataset and classifier, it is reasonable because sudden changes in velocity and acceleration of keypoints occur during falls. It is recommended to develop a list of handcrafted features, including velocity, acceleration, angles, and WHR. Then perform feature selection [146] to determine the optimal set of features for fall detection.

Handcrafted features, while generally explainable, require significant skills and time to design effectively. In contrast, DL methods provide the ability to automatically learn features from data. Deep features for skeleton-based fall detection can be acquired by utilizing various type of NN such as 1D CNN [64], [76], RNN/LSTM/GRU [56], [57], [58], [59], [60], [61], GCN [66], [87], [147], some modern networks (Transformer [56], BERT [68]) or combination of these architectures [57], [62], [63], [67]. Due to the variability of evaluation settings (discussed in Section IV-G1), it is challenging to determine the superior methods. However, some insights can be derived from several of the reviewed studies. The work [59] found that both LSTM and GRU outperformed

RNN, but it remains unclear which of the two is better due to mixed results. The study [63] demonstrated that GRU performed better than 1D-CNN. Interestingly, the authors in [57] discovered that increasing the number of stacked LSTM layers does not guarantee better performance, as their 4-layer model performed worse than the 3-layer model. Two studies [56], [68] showed that transformer-based methods outperformed LSTM under similar evaluation settings, while the work [66] presented that GCN-based architecture outperformed 1D-CNN architecture. Incorporating modern network design mechanisms, such as residual connections [22], attention [83], or self-attention [88], has been demonstrated to enhance model performance in various studies [57], [63], [87]. Therefore, leveraging modern network architectures and network design mechanisms such as attention is recommended to improve model performance.

Furthermore, deep features suffer from limited interpretability since they are derived from black-box deep neural network architectures [148]. There has been no discussion of the explanation of features learned by DL models or their similarity to handcrafted features in the reviewed works. The exploration of explainable AI is a promising direction for future research, as the topic has emerged as a prominent topic in AI research [149].

### E. RECOGNITION

By using a set of rules or training a classifier, features extracted from previous steps can be used to determine whether a fall occurred.

#### 1) RULE-BASED METHODS

Rule-based methods can use extracted features as input values to determine if a fall has occurred. In [127], the authors employed three extracted features as three decision conditions in a decisive flowchart to recognize falls, as shown in Fig. 11. Similarly, the authors in [128] performed the experiments on URFD [97] and two self-collected datasets to find out the best threshold for their handcrafted features, which included three inclination angles ($\theta_1$, $\theta_2$, and $\theta_3$) as well as the HWR of the bounding box, to recognize falls.

In their work [72], the authors utilized reconstruction error analysis to detect fall events. They compared the reconstruction error, measured by MSE, obtained from their models with predetermined thresholds. If the reconstruction error exceeded the threshold, the input was classified as a fall event, and vice versa. Threshold determination varied depending on the type of employed model. For SVD/PCA-based models, the maximum MSE served as the threshold, whereas for neural network-based AE models, the threshold was determined by multiplying the mean MSE by a constant. Each threshold value was determined by experimental analysis using validation data consisting solely of ADL data, which accounted for 5% of the total experimental dataset.

A rule-based approach can be used in conjunction with an ML-based approach in an overall recognition process.

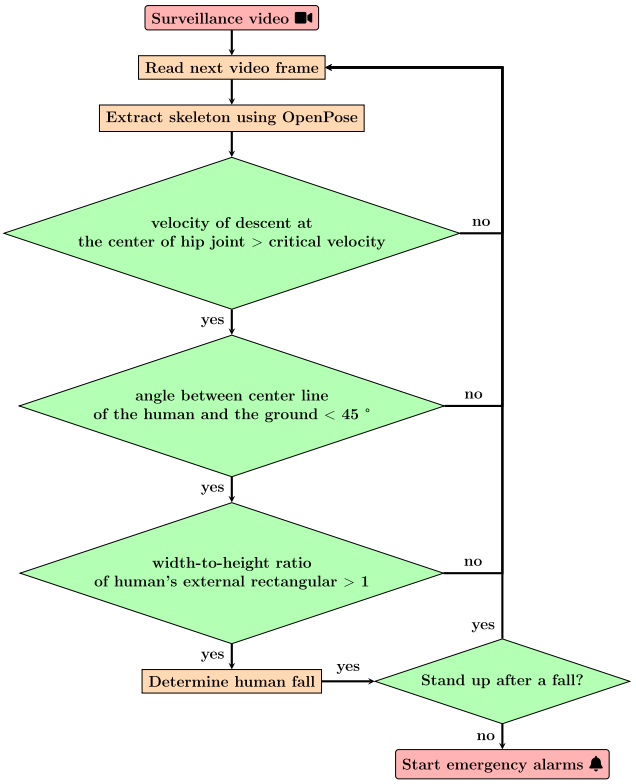



**FIGURE 11.** Ruse-based flowchart for skeleton-based fall detection in [127]. Adapted from [127] under the CC BY 4.0 license.

For example, in [47], the authors used ML-based methods to perform the state classification (falling state using MLP and fallen state using RF), and then created a set of rules to decide whether a person was falling or not based on the state classification results as shown in Fig. 6. In [82], the final fall detection result was computed based on inference results of LSTM models.

#### 2) ML-BASED METHODS

After extracting features, the next step is to use them to train a fall recognition classifier. There are several types of classifiers that can be used for this purpose. If the input features are handcrafted, traditional classifiers such as SVM [48], DT [36], RF [44], and KNN [40] can be utilized, as in [39], [41], [43], [46]. In contrast, if the input features are DL features, most methods use an MLP to map the inputs to binary classification output (fall/not fall) by using the Softmax activation function at the output layer, as in [56], [59], [66].

It is worth noting that trained models can continuously detect falls, but this is computationally expensive. To alleviate this, the authors in [60] proposed a pre-filtering process based on judgements of spine line (line between Mid.Shoulder and Mid.Hip) deviation angle, distance, acceleration, and the HWR of bounding box to filter out all non-fall events, reducing the computational cost of running the detection model. The ML techniques used in each reviewed work can be found in Table 4.

#### 3) DISCUSSIONS

In general, rule-based methods are simple, easy to implement, explainable, and computationally efficient. There is, however, a challenge in determining appropriate rules/thresholds due to the diversity of falls and individual characteristics, which can result in high false alarm rates. In contrast, ML-based methods can automatically learn complex fall patterns from data, but they are more difficult to explain and require a large amount of data.

### F. PERFORMANCE METRICS

Skeletal data of a frame or a sequence of frames are fed into the skeleton-based fall detection system to predict whether the fall has occurred or not, treating it as a binary classification problem. The performance of a skeleton-based fall detection system is evaluated using the metrics proposed in [154], like the evaluation process for typical fall detection systems:

- Accuracy:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{14}$$

- Sensitivity, also known as Recall:

$$Sensitivity/Recall = \frac{TP}{TP + FN}, \tag{15}$$

- Specificity:

$$Specificity = \frac{TN}{TN + FP}, \tag{16}$$

- F1 score:

$$F1 = \frac{2}{Recall^{-1} + Precision^{-1}} = \frac{2TP}{2TP + FP + FN}, \tag{17}$$

where $TP$ stands for the number of true positives, i.e., where a fall occurred and system recognized it, $TN$ represents the number of times the system correctly identified that no fall has occurred (true negative), $FP$ denotes the number of times the system wrongly recognized a fall event when there was none (false positive), $FN$ is the number of times the system failed to detect a fall event that actually occurred (false negative), and $Precision$, which is calculated as $Precision = TP/(TP + FP)$, shows the percentage of correctly identified falls among all events predicted as falls.

#### 1) ASSESSING THE UTILITY OF ACCURACY, SENSITIVITY, AND SPECIFICITY

Fall detection systems are often evaluated using accuracy, sensitivity, and specificity metrics. Accuracy is the proportion of correct predictions out of all predictions made on all events, including falls and non-falls. However, evaluating a fall detection system based solely on accuracy is insufficient due to the dataset's imbalance caused by the rarity of fall events. A system that consistently predicts no falls can achieve high accuracy but fails to recognize any fall event. Therefore, sensitivity/recall and specificity metrics are employed in addition to accuracy to ensure a comprehensive evaluation.

Sensitivity (recall) indicates the percentage of correctly identified falls among all actual falls, while specificity indicates the percentage of correctly detected non-falls among all actual non-falls. As sensitivity increases, the system becomes better at detecting true falls and reducing the number of undetected falls (false negatives). Conversely, increasing specificity can improve and enhance the system's ability to accurately classify non-fall events, thereby reducing the occurrence of false positives (false alarms).

Typically, there is a trade-off between sensitivity and specificity. In general, increasing sensitivity will decrease specificity, and vice versa. Therefore, it is critical to find a balance between the two metrics which is appropriate for the application. When considering fall detection, assigning greater importance to sensitivity over specificity is preferable because the cost of not detecting true falls is generally much higher than that of false alarms.

$$MathF_\beta = \frac{(1 + \beta^2) \times Precision \times Recall}{(\beta^2 \times Precision) + Recall}, \tag{18}$$

#### 2) ASSESSING THE SUFFICIENCY OF THE $F1$ SCORE

Many reviewed works utilized the $F1$ score for performance evaluation. It combines precision and recall into a single metric using the harmonic mean. However, one drawback of the $F1$ score is its inability to adjust the proportional importance of precision and recall. In certain contexts, such as healthcare systems like fall detection, it is crucial to prioritize recall over precision because the failure to detect real falls can have severe consequences.

To overcome this limitation, we propose replacing the use of the $F1$ score with the adoption of the $F_\beta$ score, represented by (18), as a more appropriate metric for fall detection systems. The weighting of precision and recall is adjustable using the *beta* parameter in the $F_\beta$ score. When $\beta$ is set to a value less than 1, precision is given more importance. On the other hand, when $\beta$ is set to a value greater than 1, recall/sensitivity is treated as more significant, resulting in a penalty for false negatives, which is suitable for fall detection systems, as discussed previously. The choice of $\beta$ can be adjusted based on the specific requirements of the system. For instance, in critical environments like hospitals, $\beta$ can be set to 2 or 3. Conversely, in non-critical environments such as homes, a slightly lower $\beta$ value, such as 1.5, may be sufficient.

In conclusion, we recommend employing the $F2$ score ($F_\beta$ score with $\beta = 2$) for evaluating fall detection systems that prioritize minimizing false negative predictions. This approach aligns with its usage in certain disease detection systems [155], [156].

### G. PERFORMANCE RESULTS

Table 5 displays the performance of the reviewed methods on the standard datasets. To ensure clarity, we present the best performance achieved by each method for each metric, as they were evaluated across various settings.

#### 1) ISSUE OF DIFFERENT EVALUATION SETTINGS

To the best of our knowledge, despite the publication of several studies and datasets in this research field, there is a lack of standard evaluation protocols. The relevant works may conduct experiments on different data and different evaluation settings (data selection method, data splitting method), which makes it difficult to compare the works.

Evaluation settings also varied among the different works since the public datasets do not officially provide the data splits for training/validation/testing. For example, both works [58] and [66] used the same URFD dataset [97] for evaluation, but they used window-based samples with different window sizes and different data splits.

Some studies did not conduct testing on widely used public datasets but instead utilized self-built, non-public datasets, which poses challenges in terms of reproducibility. For example, the authors in [56] merged AI Hub dataset collected by Korean government and the Kist SynADL [152] datasets to train/test their systems. Similarly, [43], [127] chose to use their own dataset to evaluate their methods.

Furthermore, as mentioned in Section IV-G2.b, the reporting of speed performance is often overlooked and lacks standardization.

#### 2) PERFORMANCE COMPARISON

##### *a: RECOGNITION PERFORMANCE ANALYSIS*

Section IV-D3 provides separate discussions on the performance of handcrafted and DL methods, without direct comparison between these two approaches, which is the focus of this section.

Our analysis shows that there is no clear answer to the question of whether handcrafted or DL approaches are better for skeleton-based fall detection. This is because the different evaluation settings (Section IV-G1) make it difficult to compare different methods. Additionally, all the reviewed works used either handcrafted or DL methods, but not both. To the best of our knowledge, there is a lack of comprehensive research that directly compares handcrafted and DL methods for fall detection using skeletal data, especially within the same evaluation settings. This contrasts with other works that have conducted similar comparisons for action recognition tasks using skeletal data [157]. Therefore, we strongly recommend that future research conduct this comparison to determine the more appropriate approach for skeleton-based fall detection.

Interestingly, the study [157] revealed that handcrafted features exhibited better performance on smaller datasets compared to DL features, whereas DL methods yielded better results on larger datasets in action recognition using skeleton data. This finding suggests that DL approaches have the potential to become prominent in the research field of skeleton-based fall detection as the availability of data increases.

##### *b: SYSTEM SPEED ANALYSIS*

There is a lack of speed consideration in many reviewed works. They reported the recognition performance but did not report system speed performance, which is an important factor for real-time fall detection systems. In Table 5, the inconsistency in reporting speed across studies, including different speed measurement approaches (i.e., overall system pipeline speed or just classification model inference speed) and different hardware configurations, makes it challenging to compare speed performance directly. The hardware used for evaluation significantly influences the system's speed. For instance, the study [60] achieved 10 fps on an edge computing device, while [65] reported 40 fps on a more powerful GPU. To address this issue, we recommend that future studies report system speed using standardized hardware for fair comparisons.

Some specific works provide insight into the speed of different stages within the system pipeline. For example, the work [46] found that pose estimation accounted for about 90% of the system's processing time, with the remaining 10% being allocated to other operations. The keypoint extraction stage took 68 times longer than the feature extraction stage and 12 times longer than the classification stage in [47]. Another paper [57] reported that pose estimation took 46.35 ms, while the other steps took only 2.24 ms. The speed of classification models has been investigated and found to be highly efficient. Specifically, two studies [56], [66] showed that their DL models, utilizing Transformer and GCN architectures, respectively, achieved inference times of only about 1 millisecond.

These results indicate that the pose estimation stage is the most time-consuming component in the system pipeline, compared to other stages such as data preprocessing, feature extraction, and classification. The other stages generally exhibit fast processing times mainly due to the small size of the skeleton input sample. This characteristic of small size results from the standard composition of the sample, determined by multiplying the number of keypoints (usually ranging from 15 to 33 keypoints) by the dimension of each keypoint (2 for 2D keypoints or 3 for 3D keypoints), and the size of the sample window. Data preprocessing and handcrafted feature extraction require only a small amount of computation. Furthermore, the classification models, including DL models, are generally designed to be moderately complex, with a limited number of parameters, to mitigate the risk of overfitting problems. Consequently, the classification stage is also fast. These observations emphasize the importance of selecting an appropriate pose estimation method to achieve real-time performance in skeleton-based fall detection systems.

**TABLE 5.** Performance comparison of the existing skeleton-based fall detection methods in RGB videos. Recognition metrics: Acc = Accuracy, Sens = Sensitivity, Spec = Specificity, F1 = F1 score, Other = Other Dataset (see *Evaluation* ▶ *Dataset* column in Table 4). Speed metrics: whole system pipeline speed (fps = frames per second), classification model inference speed (ms/infer = milliseconds per inference). Hardware types provided in Speed column: E = Edge computing devices, C = CPU, G = GPU, U = Unknown. Further hardware details are in the original papers.

| Method | Multicam [95] | | | | Le2i [96] | | | | URFD [97] | | | | UP-Fall [98] | | | | Other | | | | Speed |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *Acc* | *Sens* | *Spec* | *F1* | *Acc* | *Sens* | *Spec* | *F1* | *Acc* | *Sens* | *Spec* | *F1* | *Acc* | *Sens* | *Spec* | *F1* | *Acc* | *Sens* | *Spec* | *F1* | |
| Alaoui *et al.* 2019 [39] | - | - | - | - | 98.5 | 97 | 100 | - | 97.5 | 100 | 95 | - | - | - | - | - | - | - | - | - | - |
| Hasan *et al.* 2019 [58] | - | 98.9 | 96 | - | - | 99 | 97 | - | - | 99 | 96 | - | - | - | - | - | - | - | - | - | - |
| Jeong *et al.* 2019 [82] | - | - | - | - | - | - | - | - | - | 97.38 | - | - | - | - | - | - | - | - | - | - | - |
| Lin *et al.* 2020 [59] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 98.2 | 100 | 96.4 | - | - |
| Chen *et al.* 2020 [127] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 97 | 98.3 | 95 | - | - |
| Serpa *et al.* 2020 [55] | - | - | - | - | - | - | - | - | - | 94.45 | 99.9 | - | - | - | - | - | - | - | - | - | - |
| Wang *et al.* 2020 [47] | - | - | - | - | 96.91 | 96.51 | 97.37 | 97.08 | 97.33 | 97.78 | 96.67 | 97.78 | - | - | - | - | - | - | - | - | - |
| Chang *et al.* 2021 [60] | 98.3 | - | - | - | 98.1 | - | - | - | 97.6 | - | - | - | - | - | - | - | - | - | - | - | E:10 fps |
| Dentamaro *et al.* 2021 [42] | - | - | - | - | 98 | 97.2 | 98.3 | 98 | 99.6 | 97 | 98 | 99.6 | - | - | - | - | - | - | - | - | - |
| Galvao *et al.* 2021 [72] | - | - | - | - | - | - | - | - | 100 | 100 | 100 | 100 | 98.62 | 92 | 99 | 93 | - | - | - | - | - |
| Huu *et al.* 2021 [61] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 99 | - | - | - | G:13 fps |
| Liu *et al.* 2021 [43] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 94.2 | - | - | - | - |
| Ramirez *et al.* 2021 [41] | - | - | - | - | - | - | - | - | - | - | - | - | 99.34 | 98.23 | 99.48 | 98.52 | - | - | - | - | - |
| Chen *et al.* 2022 [76] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 99.83 | 94.25 | - | - | C:18/G:63 fps |
| Inturi *et al.* 2022 [62] | - | - | - | - | - | - | - | - | - | - | - | - | 98.59 | 94.37 | 98.96 | 92.47 | - | - | - | - | - |
| Juraev *et al.* 2022 [56] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 94.35 | | | | G: 1 ms/ infer |
| Lau *et al.* 2022 [63] | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | 96.18 | - | - | - | - |
| Li *et al.* 2022 [57] | - | - | - | - | - | - | - | - | 99.73 | 98.75 | 99.74 | - | 99.62 | 99.26 | 99.84 | - | - | - | - | - | G:21 fps |
| Ramirez *et al.* 2022 [46] | - | - | - | - | - | - | - | - | - | - | - | - | 99.81 | 99.81 | 99.81 | 99.56 | - | - | - | - | - |
| Suarez *et al.* 2022 [64] | - | - | - | - | - | - | - | - | 99.2 | 98.65 | - | 97.73 | 89.3 | 83.41 | - | 83.67 | - | - | - | - | - |
| Yadav *et al.* 2022 [65] | - | - | - | - | - | - | - | - | - | - | - | - | 96.7 | 96.7 | - | 96.6 | - | - | - | - | G:40 fps |
| Yuan *et al.* 2022 [87] | - | - | - | - | 98.43 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Zahan *et al.* 2022 [66] | - | - | - | - | - | - | - | - | 100 | 100 | 100 | 100 | 88.71 | 92.94 | 85.15 | 88.27 | - | - | - | - | U:1.05 ms/infer |
| Zhang *et al.* 2022 [128] | - | - | - | - | - | - | - | - | 99 | 98 | 99 | - | - | - | - | - | 98.5 | 98.2 | 98.4 | - | - |
| Amsaprabhaa *et al.* 2023 [67] | - | - | - | - | - | - | - | - | 95.8 | 94.31 | - | 96.19 | - | - | - | - | - | - | - | - | - |
| Ramirez *et al.* 2023 [68] | - | - | - | - | - | - | - | - | - | - | - | - | 99.5 | 85.79 | 99.71 | 87.2 | - | - | - | - | - |

### 3) SKELETON-BASED FALL DETECTION: RESOLVED OR ONGOING?

The metrics recorded in Table 5 demonstrate consistently high values, with numerous studies achieving accuracy, sensitivity, and specificity exceeding 98%. The remarkable performances of these works raise a crucial question: *"Has the problem of skeleton-based fall detection been truly solved?"*. However, our evaluation reveals a negative answer to this question. We argue that the high performance observed in these studies is due to two main factors:

- *The simplicity of the test dataset*: The available datasets are small and simple, typically assuming the presence of only one person in the scene. Falls are simulated, and the environment is controlled, which fails to capture the complexity of real-world scenarios.
- *The lack of a standardized and challenging evaluation protocol*: Most studies have the flexibility to determine sample construction and data partitioning for training, validation, and testing, resulting in varying levels of difficulty during evaluation. For example, in [66], the authors utilized the URFD dataset [97] and the UP-Fall dataset [98] to evaluate their system, employing two evaluation methods. The URFD dataset was evaluated using a standard random split of 70% for training and

30% for testing, whereas the UP-Fall dataset adopted a cross-trial assessment approach, using the first two trials for training and the remaining trial for testing. Notably, there was a significant contrast in the results obtained from these two datasets. Results on URFD achieved perfect accuracy, sensitivity, and specificity of 100%, while performance on UP-Fall achieved only 88.7%, 92.94%, and 85.15%, respectively. We argue that this disparity can be attributed to variances in the dataset complexity and the challenging nature of the evaluation protocol. Obviously, the evaluation protocol on the UP-Fall dataset is more challenging than the URFD dataset.

Based on these observations, we conclude that the problem of fall detection is still not fully solved, and there is much room for improvement.

## V. CHALLENGES AND FUTURE RESEARCH DIRECTIONS

### A. SCARCITY OF DATASET

High-quality datasets are scarce despite extensive research in the field of fall detection.

In terms of quality, the existing datasets are either simulated or collected in laboratory environments. The simulated dataset is not realistic enough to be used in real-life applications, e.g., the dataset collected in a lab environment does not cover the fall events that happen in a crowded environment. There have been limited efforts to enhance the authenticity of fall datasets, such as the creation of a high-quality simulation fall dataset [158] or the gathering of naturally occurring fall videos from YouTube [159].

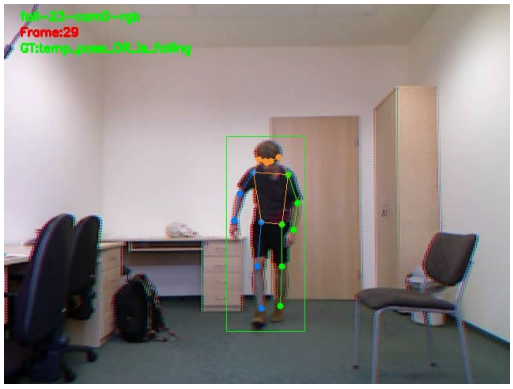
In terms of quantity, falls are relatively rare events, making it challenging to collect a large amount of data. The existing fall datasets are much smaller compared to other action recognition datasets, such as Kinetics [160] with 0.5 million videos. The lack of large-scale datasets presents a major challenge for researchers in developing robust fall detection systems. The shortage of fall video data could be addressed by:

- *Synthetic data generation*: fall video data can be synthesized by simulating the fall events in a virtual environment [56], [161] or employing physical-based stimulation [162].
- *Data sharing promotion*: Data sharing of fall video data should be actively encouraged among research communities and other stakeholders, including the elderly, governments, and the healthcare industry, by providing appropriate incentives [163], [164]. By implementing effective strategies, data sharing can be the key in constructing large-scale, high-quality fall video datasets, similar to the success of the COUGHVID medical dataset [165].

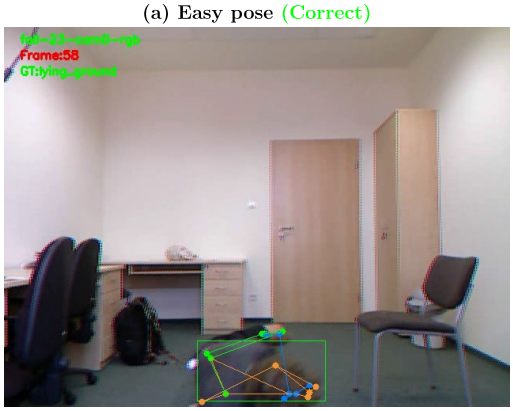
### B. 3D KEYPOINT UTILIZATION AND POSE QUALITY ENHANCEMENT

#### 1) STRENGTHENING THE USE OF 3D KEYPOINTS

Most studies listed in Table 4 primarily used 2D skeleton data, while only a subset of studies employed 3D skeleton data. However, it has been observed that 2D skeleton data yield higher error rates in action recognition, particularly for some action categories such as falls, during action recognition [132]. These higher error rates can be attributed to limitations such as the absence of depth information and sensitivity to size and orientation [132]. Therefore, it is crucial to explore the potential of utilizing 3D skeleton data in skeleton-based fall detection. This exploration can be facilitated by taking advantage of advances in more efficient 3D pose estimation methods [107] and a variety of approaches for 3D skeleton-based action recognition [107].




(a) Easy pose (Correct)




(b) Falling/fallen pose (Incorrect)

**FIGURE 12.** Poor pose estimation when the person is falling/fallen. Example from URFD dataset [97].

#### 2) ENHANCING POSE ESTIMATION QUALITY

Skeleton-based action recognition is highly dependent on the accuracy of pose estimation. Poor pose estimation can negatively affect the recognition performance [125]. In the case of a fall, the action typically occurs in a short period of time, and the pose estimation may be inaccurate when the person is falling/fallen. Fig. 12 shows an example of bad pose estimation when the person is falling/fallen.

Therefore, there is a need to enhance the keypoint quality through the adoption of advanced pose estimation methods. These newer methods, such as [166], [167] for 2D keypoints, and [168], [169] for 3D keypoints, offer significant advancements in terms of keypoint quality. A list of frequently

updated 2D and 3D pose estimation methods can be found online at [170] and [171], respectively. In addition to the pose estimation frameworks mentioned in Section IV-B1, exploring other frameworks that implement more advanced pose estimation methods is a promising direction. The MMPose framework [172] serves as a noteworthy example in this regard.

Alternatively, the pose estimation quality can be improved by implementing pose refinement techniques such as [173] for 2D poses and [174] for 3D poses.

### C. IMPROVEMENTS IN LEARNING METHODS

Several approaches can be employed to improve learning methodologies:

- Employing advanced architectures can enhance skeleton-based fall detection performance, as discussed in Section IV-D3. As shown in Table 4, most methods relied on conventional architectures such as CNN, LSTM, and GRU, with few studies using modern architectures such as Transformer [56], [68], GCN [66], [87], and ST-GCN [67], [72].The evolution of architectures, such as Transformers and GNN/GCN, has led to notable improvements. This includes the development of more powerful Transformers [175], enhanced GNNs inspired by ST-GCN [176], and innovative hybrid approaches that combine Transformers and GNNs, as demonstrated in [177]. However, existing work in skeleton-based fall detection has yet to fully exploit the potential of these advanced architectures. Incorporating these advanced architectures holds great promise for enhancing skeleton-based fall detection methods.
- Few-shot learning is a sub-area of ML that focuses on learning from a few training samples. There are several approaches [178], [179] to leverage few-shot learning in action recognition. Obviously, the few-shot learning is a promising direction to improve the performance of fall detection, given the shortage of fall data.
- Semi-supervised [180], unsupervised learning [181] are also promising directions as they can harness unlabeled data to enhance the performance of the detection system.
- Multi-modality learning has shown potential in enhancing action recognition performance by incorporating information from various modalities, such as RGB, depth, skeleton, or infrared data [25]. Therefore, it is promising to develop frameworks for fall detection that leverage multi-modality learning. Indirect implementation of multi-modality learning involves utilizing skeleton data estimated from multiple modalities. For example, the authors in [182] employed 3D skeleton data estimated from millimeter wave (mmWave), RGB images, and Inertial Measurement Unit (IMU) sensors for action detection. Alternatively, multi-modality learning can be directly achieved by utilizing different types of data from multiple modalities. For instance, in [183], the authors employed both skeleton data and RGB images for action recognition. The availability of multi-modality fall datasets like UP-Fall [98] makes the development of such frameworks feasible.

### D. STANDARDIZATION OF EVALUATION PROTOCOL

The variability in results presented in Table 5 can be attributed to the lack of uniform evaluation protocols. This inconsistency gives rise to multiple issues concerning the comparability of recognition performance and speed performance across different studies, as well as the reproducibility of the results, as discussed in Section IV-G1. Therefore, it is crucial to establish a standardized evaluation protocol, which includes both recognition performance and speed performance, for fair comparison and reproducibility.

As discussed in Section IV-G3, existing evaluation methods do not include challenging aspects in their assessment. The generalizability of the proposed methods should be assessed using tougher evaluation protocols to address this limitation. The robustness of the results would be enhanced if cross-view evaluation (training and testing on different camera views) or cross-subject evaluation (training and testing on different subjects) were incorporated, as in [133].

### E. TOWARDS REAL-LIFE SKELETON-BASED FALL DETECTION SYSTEMS

#### 1) ADDRESSING THE NEEDS OF THE ELDERLY

Despite being a primary user group, the needs and perspectives of the elderly are often overlooked in the development and usage of fall detection systems [184]. This oversight extends to skeleton-based fall detection systems as well. The current skeleton-based fall detection systems heavily rely on simulated datasets that fail to accurately represent the age range of the elderly (refer *Age Range* column to Table 2) or neglect the actual demands of the elderly in building such systems. To bridge this gap, it is essential to involve key stakeholders, including AI researchers, system developers, healthcare/biomechanics professionals, and most importantly, the elderly themselves, in the development process. By actively engaging all these stakeholders, we can ensure that the resulting systems effectively address the real demands of the elderly and cater to their unique needs.

#### 2) PRIVACY-PRESERVING, EXPLAINABLE SYSTEMS

Current skeleton-based fall detection systems depend on extracting skeleton data from RGB videos. However, these RGB videos may raise privacy concerns due to the presence of sensitive personal information, such as identifiable features like faces and genders. This contrasts with to other fall detection systems that provide privacy protection [11]. Therefore, it is crucial to develop a skeleton-based fall detection system that effectively protects individual privacy. One solution is to develop privacy-preserving pose estimation methods [185]. In addition, explainability is another factor that should be considered when developing skeleton-based fall detection systems that employ DL methods, as discussed in IV-D3.

This is particularly important to improve user confidence and acceptance of the developed systems.

#### 3) MULTI-PERSON, MULTI-CAMERA, EXPLAINABLE FALL DETECTION SYSTEM IN REAL-TIME

The extraction of skeleton data is an essential step in skeleton-based fall detection methods, but it also increases the time cost of the system. Therefore, it is crucial to design the system efficiently to ensure its real-time performance. In addition, most of the existing methods can detect a fall event only for a single person from a single camera. However, in real-life scenarios, especially in surveillance, there may be multiple people and multiple cameras involved. To address this, a skeleton-based fall detection system should be able to detect the fall events for multiple people simultaneously and make use of the information from multiple cameras to improve its performance.

### F. FALL PREDICTION

The terms "*action recognition*" and "*action prediction*" can be easily confused and have diverse definitions across various studies. While action recognition typically refers to the task of recognizing a human action from a trimmed video input where the entire action execution is captured from its beginning to its end, action prediction aims to use incomplete video input to identify the action, i.e., making inferences about an action before fully observing its complete execution [186]. Moreover, action prediction can be categorized into two subtasks, as outlined in [186]:

- *Early action prediction*: aims to predict the action in a trimmed video by observing only a limited number of initial frames (e.g., 30% of total frames) of the video.
- *Long-term action prediction*: focus on prediction the future action based on the observed action. In other words, given a sequence of action $\boldsymbol{A}$ (which can be either completed or ongoing), the objective is to predict the subsequent action $\boldsymbol{B}$. *Risk-based action prediction*, as discussed in [187], [188], is a specific type of long-term action prediction. Its aim is to forecast an action in advance, with a defined anticipation time $\tau_a$ before the action takes place. Essentially, the task involves predicting the action label at time $t - \tau_a$, given that the action begins at time stamp $t$.

Similar to action recognition, fall detection systems typically take a reactive approach, detecting a fall event only after it has occurred. Despite accurate fall detection, the consequences of the fall can be severe, especially for the elderly. To mitigate this, a promising approach is to predict falls before they are fully completed (similar to early action prediction) or, ideally, even before they occur (like risk-based action prediction). This proactive approach enables timely intervention to mitigate the negative consequences associated with falls. For instance, activating safety airbags to protect critical body parts of the elderly can be implemented when a fall is predicted to happen [189]. Moreover, risk-based fall detection becomes even more valuable if falls can be predicted well in advance, such as over extended periods of time (e.g., days, weeks). This enables caregivers to prioritize attention for individuals who are at a higher risk of falling. An example of this is presented in [190], where the authors analyze gait parameters to predict the likelihood of falls occurring within the next three weeks.

Skeleton-based fall prediction remains relatively unexplored in the literature, with only a limited number of studies conducted thus far (e.g., early fall detection as shown in [191], [192]). As a result, there is significant potential for further investigation and research in this area.

## VI. CONCLUSION

Fall detection is a crucial function in intelligent healthcare systems, especially for the elderly. Researchers have developed various methods that use skeleton data extracted from RGB videos to detect falls.

In this paper, we conducted a comprehensive examination of skeleton-based fall detection methods. Our analysis covered the entire system pipeline, including data collection, data preprocessing, feature extraction, and recognition techniques. Additionally, we provided a performance comparison of the reviewed methods on the popular fall detection benchmark datasets. Moreover, we identified the key challenges faced in this field and highlighted potential research directions for future studies.

Through extensive discussions and analysis, this paper aims to inspire researchers in this field to emphasize the practical application aspects, such as privacy concerns and detection speed, of skeleton-based fall detection systems, rather than focusing solely on the recognition accuracy. We hope that this paper will serve as a useful reference for researchers and practitioners in this research domain.

## REFERENCES


[1] *Ageing and Health*. Accessed: Oct. 21, 2022. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/ageing-and-health

[2] (Dec. 2020). *Keep on Your Feet-Preventing Older Adult Falls | CDC*. Accessed: Sep. 29, 2022. [Online]. Available: https://www.cdc.gov/injury/features/older-adult-falls/index.html

[3] D. Yacchirema, J. S. de Puga, C. Palau, and M. Esteve, "Fall detection system for elderly people using IoT and ensemble machine learning algorithm," *Pers. Ubiquitous Comput.*, vol. 23, nos. 5–6, pp. 801–817, Nov. 2019.

[4] F. Shu and J. Shu, "An eight-camera fall detection system using human fall pattern recognition via machine learning by a low-cost Android box," *Sci. Rep.*, vol. 11, no. 1, p. 2471, Jan. 2021.

[5] G. Şengül, M. Karakaya, S. Misra, O. O. Abayomi-Alli, and R. Damaševičius, "Deep learning based fall detection using smartwatches for healthcare applications," *Biomed. Signal Process. Control*, vol. 71, Jan. 2022, Art. no. 103242.

[6] S. Rastogi and J. Singh, "Human fall detection and activity monitoring: A comparative analysis of vision-based methods for classification and detection techniques," *Soft Comput.*, vol. 26, no. 8, pp. 3679–3701, Apr. 2022.

[7] J. Howcroft, J. Kofman, and E. D. Lemaire, "Prospective fall-risk prediction models for older adults based on wearable sensors," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 10, pp. 1812–1820, Oct. 2017.

[8] F. Hussain, F. Hussain, M. Ehatisham-Ul-Haq, and M. A. Azam, "Activity-aware fall detection and recognition based on wearable sensors," *IEEE Sensors J.*, vol. 19, no. 12, pp. 4528–4536, Jun. 2019.

[9] A. Chelli and M. Pätzold, "A machine learning approach for fall detection and daily living activity recognition," *IEEE Access*, vol. 7, pp. 38670–38687, 2019.
[10] M. Daher, A. Diab, M. E. B. El Najjar, M. A. Khalil, and F. Charpillet, "Elder tracking and fall detection system using smart tiles," *IEEE Sensors J.*, vol. 17, no. 2, pp. 469–479, Jan. 2017.
[11] S. Mashiyama, J. Hong, and T. Ohtsuki, "Activity recognition using low resolution infrared array sensor," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 495–500.
[12] Z. Liu, M. Yang, Y. Yuan, and K. Y. Chan, "Fall detection and personnel tracking system using infrared array sensors," *IEEE Sensors J.*, vol. 20, no. 16, pp. 9558–9566, Aug. 2020.
[13] D. Droghini, E. Principi, S. Squartini, P. Olivetti, and F. Piazza, "Human fall detection by using an innovative floor acoustic sensor," in *Multidisciplinary Approaches to Neural Computing*. Cham, Switzerland: Springer, 2018, pp. 97–107.
[14] Y. Kong, J. Huang, S. Huang, Z. Wei, and S. Wang, "Learning spatiotemporal representations for human fall detection in surveillance video," *J. Vis. Commun. Image Represent.*, vol. 59, pp. 215–230, Feb. 2019.
[15] J. Nogas, S. S. Khan, and A. Mihailidis, "DeepFall: Non-invasive fall detection with deep spatio-temporal convolutional autoencoders," *J. Healthcare Informat. Res.*, vol. 4, no. 1, pp. 50–70, Mar. 2020.
[16] S. J. Berlin and M. John, "Vision based human fall detection with Siamese convolutional neural networks," *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 5751–5762, Apr. 2021.
[17] Z. Zhang, "Microsoft Kinect sensor and its effect," *IEEE Multimedia-Mag.*, vol. 19, no. 2, pp. 4–10, Feb. 2012.
[18] *Intel Realsenset Computer Vision—Depth and Tracking Cameras*. Accessed: Oct. 21, 2022. [Online]. Available: https://www.intelrealsense.com/#Products
[19] P. Weinzaepfel and G. Rogez, "Mimetics: Towards understanding human actions out of context," *Int. J. Comput. Vis.*, vol. 129, no. 5, pp. 1675–1690, May 2021.
[20] L. Song, G. Yu, J. Yuan, and Z. Liu, "Human pose estimation and its application to action recognition: A survey," *J. Vis. Commun. Image Represent.*, vol. 76, Apr. 2021, Art. no. 103055.
[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[23] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "SphereFace: Deep hypersphere embedding for face recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6738–6746.
[24] A. Bochkovskiy, C.-Y. Wang, and H.-Y. Mark Liao, "YOLOv4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[25] Z. Sun, Q. Ke, H. Rahmani, M. Bennamoun, G. Wang, and J. Liu, "Human action recognition from various data modalities: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3200–3225, Mar. 2023.
[26] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1302–1310.
[27] J. Li, C. Wang, H. Zhu, Y. Mao, H.-S. Fang, and C. Lu, "CrowdPose: Efficient crowded scenes pose estimation and a new benchmark," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10855–10864.
[28] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5686–5696.
[29] X. Wang, J. Ellul, and G. Azzopardi, "Elderly fall detection systems: A literature survey," *Frontiers Robot. AI*, vol. 7, p. 71, Jun. 2020.
[30] Md. M. Islam, O. Tayan, Md. R. Islam, Md. S. Islam, S. Nooruddin, M. N. Kabir, and M. R. Islam, "Deep learning based systems developed for fall detection: A review," *IEEE Access*, vol. 8, pp. 166117–166137, 2020.
[31] P. Vallabh and R. Malekian, "Fall detection monitoring systems: A comprehensive review," *J. Ambient Intell. Humanized Comput.*, vol. 9, no. 6, pp. 1809–1833, Nov. 2018.
[32] L. Ren and Y. Peng, "Research of fall detection and fall prevention technologies: A systematic review," *IEEE Access*, vol. 7, pp. 77702–77722, 2019.
[33] J. Gutiérrez, V. Rodríguez, and S. Martin, "Comprehensive review of vision-based fall detection systems," *Sensors*, vol. 21, no. 3, p. 947, Feb. 2021.
[34] E. Alam, A. Sufian, P. Dutta, and M. Leo, "Vision-based human fall detection systems using deep learning: A review," *Comput. Biol. Med.*, vol. 146, Jul. 2022, Art. no. 105626.
[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[36] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.
[37] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
[38] L. Breiman, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.
[39] A. Y. Alaoui, S. El Fkihi, and R. O. H. Thami, "Fall detection for elderly people using the variation of key points of human skeleton," *IEEE Access*, vol. 7, pp. 154786–154795, 2019.
[40] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 1, pp. 21–27, Jan. 1967.
[41] H. Ramirez, S. A. Velastin, I. Meza, E. Fabregas, D. Makris, and G. Farias, "Fall detection and activity recognition using human skeleton features," *IEEE Access*, vol. 9, pp. 33532–33542, 2021.
[42] V. Dentamaro, D. Impedovo, and G. Pirlo, "Fall detection by human pose estimation and kinematic theory," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 2328–2335.
[43] Y.-H. Liu, P. C. K. Hung, F. Iqbal, and B. C. M. Fung, "Automatic fall risk detection based on imbalanced data," *IEEE Access*, vol. 9, pp. 163594–163611, 2021.
[44] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[45] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[46] H. Ramirez, S. A. Velastin, P. Aguayo, E. Fabregas, and G. Farias, "Human activity recognition by sequences of skeleton features," *Sensors*, vol. 22, no. 11, p. 3991, May 2022.
[47] B.-H. Wang, J. Yu, K. Wang, X.-Y. Bao, and K.-M. Mao, "Fall detection based on dual-channel feature integration," *IEEE Access*, vol. 8, pp. 103443–103453, 2020.
[48] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst.*, vol. 13, no. 4, pp. 18–28, Jul./Aug. 1998.
[49] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *J.-Jpn. Soc. Artif. Intell.*, vol. 14, nos. 771–780, p. 1612, 1999.
[50] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[51] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, Nov. 2018, Art. no. e00938.
[52] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[53] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural Networks for Perception*. Amsterdam, The Netherlands: Elsevier, 1992, pp. 65–93.
[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, pp. 183–186.
[55] Y. R. Serpa, M. B. Nogueira, P. P. M. Neto, and M. A. F. Rodrigues, "Evaluating pose estimation as a solution to the fall detection problem," in *Proc. IEEE 8th Int. Conf. Serious Games Appl. Health (SeGAH)*, Aug. 2020, pp. 1–7.
[56] S. Juraev, A. Ghimire, J. Alikhanov, V. Kakani, and H. Kim, "Exploring human pose estimation and the usage of synthetic data for elderly fall detection in real-world surveillance," *IEEE Access*, vol. 10, pp. 94249–94261, 2022.
[57] J. Li, M. Gao, B. Li, D. Zhou, Y. Zhi, and Y. Zhang, "KAMTFENet: A fall detection algorithm based on keypoint attention module and temporal feature extraction," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 5, pp. 1831–1844, 2022.

[58] M. M. Hasan, M. S. Islam, and S. Abdullah, "Robust pose-based human fall detection using recurrent neural network," in *Proc. IEEE Int. Conf. Robot., Autom., Artificial-Intelligence Internet-Things (RAAICON)*, Nov. 2019, pp. 48–51.

[59] C.-B. Lin, Z. Dong, W.-K. Kuan, and Y.-F. Huang, "A framework for fall detection based on OpenPose skeleton and LSTM/GRU models," *Appl. Sci.*, vol. 11, no. 1, p. 329, Dec. 2020.

[60] W.-J. Chang, C.-H. Hsu, and L.-B. Chen, "A pose estimation-based fall detection methodology using artificial intelligence edge computing," *IEEE Access*, vol. 9, pp. 129965–129976, 2021.

[61] P. Nguyen Huu, N. Nguyen Thi, and T. P. Ngoc, "Proposing posture recognition system combining MobilenetV2 and LSTM for medical surveillance," *IEEE Access*, vol. 10, pp. 1839–1849, 2022.

[62] A. R. Inturi, V. M. Manikandan, and V. Garrapally, "A novel vision-based fall detection scheme using keypoints of human skeleton with long short-term memory network," *Arabian J. Sci. Eng.*, vol. 48, no. 2, pp. 1143–1155, 2022.

[63] X. L. Lau, T. Connie, M. K. O. Goh, and S. H. Lau, "Fall detection and motion analysis using visual approaches," *Int. J. Technol.*, vol. 13, no. 6, p. 1173, Nov. 2022.

[64] J. J. P. Suarez, N. Orillaza, and P. Naval, "AFAR: A real-time vision-based activity monitoring and fall detection framework using 1D convolutional neural networks," in *Proc. 14th Int. Conf. Mach. Learn. Comput. (ICMLC)*, Feb. 2022, pp. 555–559.

[65] S. K. Yadav, A. Luthra, K. Tiwari, H. M. Pandey, and S. A. Akbar, "ARFDNet: An efficient activity recognition & fall detection system using latent feature pooling," *Knowl.-Based Syst.*, vol. 239, Mar. 2022, Art. no. 107948.

[66] S. Zahan, G. M. Hassan, and A. Mian, "SDFA: Structure aware discriminative feature aggregation for efficient human fall detection in video," *IEEE Trans. Ind. Informat.*, vol. 19, no. 8, pp. 8713–8721, Aug. 2023.

[67] M. Amsaprabhaa, "Multimodal spatiotemporal skeletal kinematic gait feature fusion for vision-based fall detection," *Exp. Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118681.

[68] H. Ramirez, S. A. Velastin, S. Cuellar, E. Fabregas, and G. Farias, "BERT for activity recognition using sequences of skeleton features and data augmentation with GAN," *Sensors*, vol. 23, no. 3, p. 1400, Jan. 2023.

[69] P. Li, Y. Pei, and J. Li, "A comprehensive survey on design and application of autoencoder in deep learning," *Appl. Soft Comput.*, vol. 138, May 2023, Art. no. 110176.

[70] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

[71] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lect. IE*, vol. 2, pp. 1–18, Dec. 2015.

[72] Y. M. Galvão, L. Portela, J. Ferreira, P. Barros, O. A. D. A. Fagundes, and B. J. T. Fernandes, "A framework for anomaly identification applied on fall detection," *IEEE Access*, vol. 9, pp. 77264–77274, 2021.

[73] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.

[74] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*.

[75] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[76] Z. Chen, Y. Wang, and W. Yang, "Video based fall detection using human poses," in *Proc. CCF Conf. Big Data*. Guangzhou, China: Springer, 2022, pp. 283–296.

[77] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, Jul. 2019.

[78] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[79] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[80] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*.

[81] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[82] S. Jeong, S. Kang, and I. Chun, "Human-skeleton based fall-detection method using LSTM for manufacturing industries," in *Proc. 34th Int. Tech. Conf. Circuits/Syst., Comput. Commun. (ITC-CSCC)*, Jun. 2019, pp. 1–4.

[83] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4291–4308, Oct. 2021.

[84] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–9.

[85] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*.

[86] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.

[87] J. Yuan, C. Liu, C. Liu, L. Wang, and Q. Chen, "Real-time human falling recognition via spatial and temporal self-attention augmented graph convolutional network," in *Proc. IEEE Int. Conf. Real-Time Comput. Robot. (RCAR)*, Jul. 2022, pp. 438–443.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[89] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[90] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[91] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," 2020, *arXiv:2010.11929*.

[92] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002.

[93] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[94] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 1–9.

[95] (2010). *Multicam—Multiple Cameras Fall Dataset*. [Online]. Available: http://www.iro.umontreal.ca/~labimage/Dataset/

[96] I. Charfi, J. Miteran, J. Dubois, M. Atri, and R. Tourki, "Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and AdaBoost-based classification," *J. Electron. Imag.*, vol. 22, no. 4, Jul. 2013, Art. no. 041106.

[97] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," *Comput. Methods Programs Biomed.*, vol. 117, no. 3, pp. 489–501, Dec. 2014.

[98] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, "UP-fall detection dataset: A multimodal approach," *Sensors*, vol. 19, no. 9, p. 1988, Apr. 2019.

[99] H.-S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y.-L. Li, and C. Lu, "AlphaPose: Whole-body regional multi-person pose estimation and tracking in real-time," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 6, pp. 7157–7173, Jun. 2023.

[100] Z. Liu, H. Chen, R. Feng, S. Wu, S. Ji, B. Yang, and X. Wang, "Deep dual consecutive network for human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 525–534.

[101] *Mediapipe/Docs/Solutions/Pose.Md at Master · Google/Mediapipe · Github*. Accessed: Jun. 23, 2023. [Online]. Available: https://github.com/google/mediapipe/blob/master/docs/solutions/pose.md

[102] *TFJS-Models/Pose-Detection/SRC/Movenet at Master · Tensorflow/Tfjs-Models*. Accessed: Oct. 3, 2023. [Online]. Available: https://github.com/tensorflow/tfjs-models/tree/master/pose-detection/src/movenet

[103] W. Li, Z. Wang, B. Yin, Q. Peng, Y. Du, T. Xiao, G. Yu, H. Lu, Y. Wei, and J. Sun, "Rethinking on multi-stage networks for human pose estimation," 2019, *arXiv:1901.00148*.

[104] S. Kreiss, L. Bertoni, and A. Alahi, "OpenPifPaf: Composite fields for semantic keypoint detection and spatio-temporal association," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 13498–13511, Aug. 2022.

[105] *TFJS-Models/Pose-Detection/Src/Posenet at Master · Tensorflow/Tfjs-Models*. Accessed: Oct. 3, 2023. [Online]. Available: https://github.com/tensorflow/tfjs-models/tree/master/pose-detection/src/posenet
[106] Y. Chen, Y. Tian, and M. He, "Monocular human pose estimation: A survey of deep learning-based methods," *Comput. Vis. Image Understand.*, vol. 192, Mar. 2020, Art. no. 102897.
[107] W. Liu, Q. Bao, Y. Sun, and T. Mei, "Recent advances of monocular 2D and 3D human pose estimation: A deep learning perspective," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–41, Apr. 2023.
[108] C. Zheng, W. Wu, C. Chen, T. Yang, S. Zhu, J. Shen, N. Kehtarnavaz, and M. Shah, "Deep learning-based human pose estimation: A survey," *ACM Comput. Surv.*, Jun. 2023.
[109] S. Jin, X. Ma, Z. Han, Y. Wu, W. Yang, W. Liu, C. Qian, and W. Ouyang, "Towards multi-person pose tracking: Bottom-up and top-down methods," in *Proc. ICCV PoseTrack Workshop*, vol. 2, 2017, p. 7.
[110] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "HigherHRNet: Scale-aware representation learning for bottom-up human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5385–5394.
[111] *Openpose: Openpose Doc-Output*. Accessed: Mar. 2, 2023. [Online]. Available: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_02_output.html
[112] *Openpose: Openpose Advanced Doc—Demo—Advanced*. Accessed: May 26, 2023. [Online]. Available: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_advanced_demo_advanced.html#body-25-vs-coco-vs-mpi-models
[113] *Openpose/Doc/Advanced/3D_Reconstruction_Module.Md At Master · CMU-Perceptual-Computing-Lab/Openpose · Github*. Accessed: Jun. 20, 2023. [Online]. Available: https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/advanced/3d_reconstruction_module.md#expected-visual-results
[114] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[115] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2353–2362.
[116] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision—ECCV*. Zurich, Switzerland: Springer, 2014, pp. 740–755.
[117] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2D human pose estimation: New benchmark and state of the art analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3686–3693.
[118] *Issues · MVIG-SJTU/Alphapose*. Accessed: Jun. 20, 2023. [Online]. Available: https://github.com/MVIG-SJTU/AlphaPose/issues?q=3D
[119] *Google/Mediapipe: Cross-Platform, Customizable ML Solutions for Live and Streaming Media*. Accessed: Jun. 23, 2023. [Online]. Available: https://github.com/google/mediapipe
[120] Pose Landmark Detection Guide | Mediapipe | Google for Developers. Accessed: Jun. 23, 2023. [Online]. Available: https://developers.google.com/mediapipe/solutions/vision/pose_landmarker#pose_landmarker_model
[121] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, "BlazePose: On-device real-time body pose tracking," 2020, *arXiv:2006.10204*.
[122] I. Grishchenko, V. Bazarevsky, A. Zanfir, E. G. Bazavan, M. Zanfir, R. Yee, K. Raveendran, M. Zhdanovich, M. Grundmann, and C. Sminchisescu, "BlazePose GHUM holistic: Real-time 3D human landmarks and pose estimation," 2022, *arXiv:2206.11678*.
[123] Z. Zhang, J. Tang, and G. Wu, "Simple and lightweight human pose estimation," 2019, *arXiv:1911.10346*.
[124] J. Martinez, R. Hossain, J. Romero, and J. J. Little, "A simple yet effective baseline for 3D human pose estimation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2659–2668.
[125] H. Duan, Y. Zhao, K. Chen, D. Lin, and B. Dai, "Revisiting skeleton-based action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 2959–2968.
[126] *Model Card Blazepose GHUM 3D*. Accessed: Jul. 6, 2023. [Online]. Available: https://storage.googleapis.com/mediapipe-assets/Model%20Card%20BlazePose%20GHUM%203D.pdf
[127] W. Chen, Z. Jiang, H. Guo, and X. Ni, "Fall detection based on key points of human-skeleton using OpenPose," *Symmetry*, vol. 12, no. 5, p. 744, May 2020.
[128] Y. Zhang, X. Zheng, W. Liang, S. Zhang, and X. Yuan, "Visual surveillance for human fall detection in healthcare IoT," *IEEE MultimediaMag.*, vol. 29, no. 1, pp. 36–46, Jan. 2022.
[129] J. Sola and J. Sevilla, "Importance of input data normalization for the application of neural networks to complex industrial problems," *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3, pp. 1464–1468, Jun. 1997.
[130] C. W. Hsu, C. C. Chang, and C. J. Lin, "A practical guide to support vector classification," Dept. Comput. Sci. Inf. Eng., Univ. Nat. Taiwan, Taipei City, Taiwan, Tech. Rep., 2003, pp. 1–12.
[131] S. Bhanja and A. Das, "Impact of data normalization on deep neural network for time series forecasting," 2018, *arXiv:1812.05519*.
[132] P. Elias, J. Sedmidubsky, and P. Zezula, "Understanding the limits of 2D skeletons for action recognition," *Multimedia Syst.*, vol. 27, no. 3, pp. 547–561, Jun. 2021.
[133] J. Liu, A. Shahroudy, M. Perez, G. Wang, L.-Y. Duan, and A. C. Kot, "NTU RGB+D 120: A large-scale benchmark for 3D human activity understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2684–2701, Oct. 2020.
[134] G. Ding, J. Tian, J. Wu, Q. Zhao, and L. Xie, "Energy efficient human activity recognition using wearable sensors," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, Apr. 2018, pp. 379–383.
[135] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, Apr. 2014.
[136] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
[137] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2017, pp. 3645–3649.
[138] R. Plamondon, "A kinematic theory of rapid human movements: Part I. Movement representation and generation," *Biol. Cybern.*, vol. 72, no. 4, pp. 295–307, Mar. 1995.
[139] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[140] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, *arXiv:1511.07122*.
[141] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jun. 2014.
[142] I. Ashrapov, "Tabular GANs for uneven distribution," 2020, *arXiv:2010.00638*.
[143] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2014.
[144] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[145] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Trans. Autom. Control*, vol. AC-25, no. 2, pp. 164–176, Apr. 1980.
[146] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, Jul. 2018.
[147] M. Gao, J. Li, D. Zhou, Y. Zhi, M. Zhang, and B. Li, "Fall detection based on OpenPose and MobileNetV2 network," *IET Image Process.*, vol. 17, no. 3, pp. 722–732, Feb. 2023.
[148] D. Castelvecchi, "Can we open the black box of AI?" *Nature*, vol. 538, no. 7623, pp. 20–23, Oct. 2016.
[149] G. Ras, N. Xie, M. Van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *J. Artif. Intell. Res.*, vol. 73, pp. 329–396, Jan. 2022.
[150] K. Adhikari, H. Bouchachia, and H. Nait-Charif, "Activity recognition for indoor fall detection using convolutional neural network," in *Proc. 15th IAPR Int. Conf. Mach. Vis. Appl. (MVA)*, May 2017, pp. 81–84.
[151] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, "NTU RGB+D: A large scale dataset for 3D human activity analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1010–1019.

[152] H. Hwang, C. Jang, G. Park, J. Cho, and I.-J. Kim, "ElderSim: A synthetic data generation platform for human action recognition in eldercare applications," 2020, *arXiv:2010.14742*.
[153] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.
[154] N. Noury, A. Fleury, P. Rumeau, A. K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, "Fall detection–Principles and methods," in *Proc. 29th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, Aug. 2007, pp. 1663–1666.
[155] A. Dascalu and E. O. David, "Skin cancer detection by deep learning and sound analysis algorithms: A prospective clinical study of an elementary dermoscope," *EBioMedicine*, vol. 43, pp. 107–113, May 2019.
[156] D. Devarriya, C. Gulati, V. Mansharamani, A. Sakalle, and A. Bhardwaj, "Unbalanced breast cancer data classification using novel fitness functions in genetic programming," *Exp. Syst. Appl.*, vol. 140, Feb. 2020, Art. no. 112866.
[157] L. Wang, D. Q. Huynh, and P. Koniusz, "A comparative review of recent Kinect-based action recognition algorithms," *IEEE Trans. Image Process.*, vol. 29, pp. 15–28, 2020.
[158] G. Baldewijns, G. Debard, G. Mertes, B. Vanrumste, and T. Croonenborghs, "Bridging the gap between real-life data and simulated data by providing a highly realistic fall dataset for evaluating camera-based fall detection algorithms," *Healthcare Technol. Lett.*, vol. 3, no. 1, pp. 6–11, 2016.
[159] Y. Fan, M. D. Levine, G. Wen, and S. Qiu, "A deep neural network for real-time detection of falling humans in naturally occurring scenes," *Neurocomputing*, vol. 260, pp. 43–58, Oct. 2017.
[160] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, M. Suleyman, and A. Zisserman, "The kinetics human action video dataset," 2017, *arXiv:1705.06950*.
[161] U. Asif, B. Mashford, S. Von Cavallar, S. Yohanandan, S. Roy, J. Tang, and S. Harrer, "Privacy preserving human fall detection using video data," in *Proc. Mach. Learn. Health Workshop*. PMLR, 2020, pp. 39–51.
[162] G. Mastorakis, T. Ellis, and D. Makris, "Fall detection without people: A simulation approach tackling video data scarcity," *Exp. Syst. Appl.*, vol. 112, pp. 125–137, Dec. 2018.
[163] A. Rowhani-Farid, M. Allen, and A. G. Barnett, "What incentives increase data sharing in health and medical research? A systematic review," *Res. Integrity Peer Rev.*, vol. 2, no. 1, pp. 1–10, Dec. 2017.
[164] W. D. Chawinga and S. Zinn, "Global perspectives of research data sharing: A systematic literature review," *Library Inf. Sci. Res.*, vol. 41, no. 2, pp. 109–122, Apr. 2019.
[165] L. Orlandic, T. Teijeiro, and D. Atienza, "The COUGHVID crowdsourcing dataset, a corpus for the study of large-scale cough analysis algorithms," *Sci. Data*, vol. 8, no. 1, p. 156, Jun. 2021.
[166] Y. Xu, J. Zhang, Q. Zhang, and D. Tao, "ViTPose: Simple vision transformer baselines for human pose estimation," 2022, *arXiv:2204.12484*.
[167] H. Liu, F. Liu, X. Fan, and D. Huang, "Polarized self-attention: Towards high-quality pixel-wise mapping," *Neurocomputing*, vol. 506, pp. 158–167, Sep. 2022.
[168] Y. Cheng, B. Wang, and R. T. Tan, "Dual networks based 3D multi-person pose estimation from monocular video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 1636–1651, Feb. 2023.
[169] Y. Xue, J. Chen, X. Gu, H. Ma, and H. Ma, "Boosting monocular 3D human pose estimation with part aware attention," *IEEE Trans. Image Process.*, vol. 31, pp. 4278–4291, 2022.
[170] *2D Human Pose Estimation | Papers With Code*. Accessed: Jul. 14, 2023. [Online]. Available: https://paperswithcode.com/task/2d-human-pose-estimation/latest
[171] *3D Human Pose Estimation | Papers With Code*. Accessed: Jul. 14, 2023. [Online]. Available: https://paperswithcode.com/task/3d-human-pose-estimation/latest
[172] MMPose Contributors. (2020). *OpenmmLab Pose Estimation Toolbox and Benchmark*. [Online]. Available: https://github.com/open-mmlab/mmpose
[173] G. Moon, J. Y. Chang, and K. M. Lee, "PoseFix: Model-agnostic general human pose refinement network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 7773–7781.
[174] A. Zeng, L. Yang, X. Ju, J. Li, J. Wang, and Q. Xu, "SmoothNet: A plug-and-play network for refining human poses in videos," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2021, pp. 625–642.
[175] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *AI Open*, vol. 3, pp. 111–132, Oct. 2022.
[176] L. Feng, Y. Zhao, W. Zhao, and J. Tang, "A comparative review of graph convolutional networks for human skeleton-based action recognition," *Artif. Intell. Rev.*, vol. 55, pp. 4275–4305, Nov. 2022.
[177] Y. Liu, H. Zhang, D. Xu, and K. He, "Graph transformer network with temporal kernel attention for skeleton-based action recognition," *Knowl.-Based Syst.*, vol. 240, Mar. 2022, Art. no. 108146.
[178] N. Ma, H. Zhang, X. Li, S. Zhou, Z. Zhang, J. Wen, H. Li, J. Gu, and J. Bu, "Learning spatial-preserved skeleton representations for few-shot action recognition," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 174–191.
[179] L. Wang and P. Koniusz, "Temporal-viewpoint transportation plan for skeletal few-shot action recognition," in *Proc. Asian Conf. Comput. Vis. (ACCV)*, Dec. 2022, pp. 4176–4193.
[180] Z. Tu, J. Zhang, H. Li, Y. Chen, and J. Yuan, "Joint-bone fusion graph convolutional network for semi-supervised skeleton action recognition," *IEEE Trans. Multimedia*, vol. 25, pp. 1819–1831, 2022.
[181] K. Su, X. Liu, and E. Shlizerman, "PREDICT & CLUSTER: Unsupervised skeleton based action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 9628–9637.
[182] S. An, Y. Li, and U. Ogras, "MRI: Multi-modal 3D human pose estimation dataset using mmWave, RGB-D, and inertial sensors," 2022, *arXiv:2210.08394*.
[183] A. Franco, A. Magnani, and D. Maio, "A multimodal approach for human activity recognition based on skeleton and RGB data," *Pattern Recognit. Lett.*, vol. 131, pp. 293–299, Mar. 2020.
[184] F. J. Thilo, B. Hürlimann, S. Hahn, S. Bilger, J. M. Schols, and R. J. Halfens, "Involvement of older people in the development of fall detection systems: A scoping review," *BMC Geriatrics*, vol. 16, no. 1, pp. 1–9, Dec. 2016.
[185] C. Hinojosa, J. C. Niebles, and H. Arguello, "Learning privacy-preserving optics for human pose estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2553–2562.
[186] Y. Kong and Y. Fu, "Human action recognition and prediction: A survey," *Int. J. Comput. Vis.*, vol. 130, no. 5, pp. 1366–1401, May 2022.
[187] I. Morawski, W.-N. Lie, L. Aing, J.-C. Chiang, and K.-T. Chen, "Deep-learning technique for risk-based action prediction using extremely low-resolution thermopile sensor array," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 6, pp. 2852–2863, Jun. 2023.
[188] R. Girdhar and K. Grauman, "Anticipative video transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 13485–13495.
[189] S.-H. Yang, W. Zhang, Y. Wang, and M. Tomizuka, "Fall-prediction algorithm using a neural network for safety enhancement of elderly," in *Proc. CACS Int. Autom. Control Conf. (CACS)*, Dec. 2013, pp. 245–249.
[190] L. J. Phillips, C. B. DeRoche, M. Rantz, G. L. Alexander, M. Skubic, L. Despins, C. Abbott, B. H. Harris, C. Galambos, and R. J. Koopman, "Using embedded sensors in independent living to predict gait changes and falls," *Western J. Nursing Res.*, vol. 39, no. 1, pp. 78–94, Jan. 2017.
[191] X. Tao and Z. Yun, "Fall prediction based on biomechanics equilibrium using Kinect," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 4, Apr. 2017, Art. no. 155014771770325.
[192] M. Hua, Y. Nan, and S. Lian, "Falls prediction based on body keypoints and Seq2Seq architecture," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1251–1259.

**VAN-HA HOANG** received the engineering degree in software and the M.S. degree in computer science from the University of Information Technology (UIT), VNUHCM, in 2014 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Department of Software, Sejong University. In 2018, he was an Internship Student with the National Institute of Informatics (NII), Tokyo, Japan. His research interests include computer vision and deep learning.

**JONG WEON LEE** (Member, IEEE) received the M.S. degree in electrical and computer engineering from the University of Wisconsin at Madisonin, in 1991, and the Ph.D. degree from the University of Southern California, in 2002. He is currently a Professor with the Department of Software, Sejong University. His research interests include virtual reality, augmented reality, and human–computer interaction.

**MD. JALIL PIRAN** (Senior Member, IEEE) received the Ph.D. degree in electronics and information engineering from Kyung Hee University, South Korea, in 2016.

Afterward, he was with the Kyung Hee University's Networking Laboratory as a Postdoctoral Fellow. He is currently an Associate Professor with the Department of Computer Science and Engineering, Sejong University, Seoul, South Korea. He has published a substantial number of technical papers in well-known international journals and conferences in the field intelligent information and communication technology (IICT), specifically in the fields of machine learning, data science, wireless communications and networking, 5G/6G, the Internet of Things (IoT), and cyber security.

Prof. Piran works with several journals as an Editor, including IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Journal of Engineering Applications of Artificial Intelligence* (Elsevier), *Journal of Physical Communication* (Elsevier), and *Journal of Computer Communication* (Elsevier). He is also the Secretary of the IEEE Consumer Technology Society on Machine learning, Deep learning, and AI. Moreover, he was the Chair of the "5G and Beyond Communications" Session at the 2022 IEEE International Conference on Communications (ICC). Additionally, he serves as a reviewer for almost all top journals and a member of several conferences as well. In the worldwide community, he is a member of the Association for Computing Machinery (ACM) and an Active Delegate from South Korea to the Moving Picture Experts Group (MPEG). IAAM "Scientist Medal of the Year 2017" was awarded to him for notable and outstanding research in new age technology and innovation in Stockholm, Sweden. In 2017, he was recognized by the Iranian Ministry of Science, Technology, and Research as an "Outstanding Emerging Researcher." Additionally, his Ph.D. dissertation has been selected as the "Dissertation of the Year 2016" by the Iranian Academic Center for Education, Culture, and Research in the Engineering Group.

**CHUN-SU PARK** received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, in 2003 and 2009, respectively. From 2009 to 2010, he was a Visiting Scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles, CA, USA. From 2010 to 2012, he was a Senior Research Engineer with Samsung Electronics. From 2012 to 2014, he was an Assistant Professor with the Department of Information and Telecommunication Engineering, Sangmyung University. From 2014 to 2016, he was an Associate Professor with the Department of Digital Contents, Sejong University. He joined the Department of Computer Education, Sungkyunkwan University, in 2017, where he is currently an Associate Professor. His research interests include video signal processing, parallel computing, and multimedia communications.