Tuijin Jishu/Journal of Propulsion Technology
ISSN: 1001-4055
Vol. 46 No. 04 (2025)

# Real-Time Fall Detection Using Compressed Video Analytics

**Sudha Chaturvedi[1], Tapsi Nagpal[1], Praveenkumar S M[2], P S Hiremath[3], Vishnu S T[2]**
[1]Department of Computer Science and Information Technology Lingayas University, India
[2]Department of Computer Science and Engineering, PES University, India
[3]Department of Computer Applications, KLE Technological University, India.

***Abstract: -*** In video analytics, the detection of a human fall event is a critical application of video surveillance systems installed for the safety of children and senior citizens. In this paper, a novel method is proposed for real-time human fall event detection in a compressed video, which employs motion vectors (MV), and Transformer Prediction Heads -You only look once version five (TPH-YOLOv5) with fuzzy logic based multi object tracking. It is termed as FMVCNN. The video compression formats, namely, MPEG-4 and H.264 are examined for validating this method. The proposed method can be adapted to any format of video codecs and any type of camera settings without any prior setup.Numerous algorithms have been explored in the literature for detecting human fall events within compressed domain video, but they suffer from limitations on account of (i) keyframes set at a constant interval, (ii) utilization of only P frames, and (iii) setup specially for a given particular codec i.e. need to resetup every time codec changes. The proposed method addresses these limitations by using keyframe intervals of variable length, utilization of P/B frames, and setting up different codec variants. Further the crucial step of event box prediction in video frames is done using fuzzy logic, where in the motion vectors that constitute event box is a fuzzy set representing uncertainty in motion vectors related to an event. The experimental setup takes into account the benchmark datasets for fall events, which are Le2i, UR, and Multiple Cameras. The experimental outcomes of the proposed FMVCNN approach encouraging and compare well with those in recent literature for raw (uncompressed) video data. The proposed FMVCNN surpasses existing contemporary approaches executed in the compressed domain by markedly enhancing both the accuracy and speed of event detection. The ablation study considered various FMVCNN variants resulting from different video codecs, fuzzy representations, video datasets, and YOLO architectures.

***Keywords****: Convolutional Neural Network, Video Processing, Motion Vectors, Transformer, Attention Mechanism, Fuzzy Inference.*

## 1. Introduction

Incidents of fall during human activities frequently occur and are a matter of considerable concern. Injuries sustained during falls frequently correlate with mortality among older persons and certain patients [1].Detecting falls is crucial for the protection of children, the elderly, and patients in both solitary interior environments and busy outdoor settings. Video monitoring of such incidents is an efficient technical solution for addressing these security issues.

Human fall detection systems are generally divided into two main types: those that use wearable sensors and those that employ computer vision technology. Furthermore, [2] provides a comprehensive overview of various sensor-based and vision-based techniques for identifying human activities. Systems based on wearable sensors utilize various sensors, such as heart rate monitors, gyroscopes, and complete data collecting systems [3], attached to those at risk of falling. Wearable sensor-based devices employ computationally demanding measures to identify inappropriate human behavior, such as calculating frontal area and expecting skeletal joint positions. Additionally, many sensor-based systems need the user to physically wear the sensors. Individuals sometimes forget or may feel uneasy wearing such devices. In these situations, video surveillance systems can be employed for the passive, continual monitoring of individuals. Upon the occurrence of anomalous activity, such as a fall, computer vision-

based systems alert the designated caregiver for prompt assistance, therefore enhancing the individual's well-being. Surveillance cameras have emerged as a significant medium for monitoring the actions of those at risk of falling, circumventing the limitations of wearable devices. The diverse angles resulting from different camera positions necessitate an innovative training approach [4] for identifying potential areas of interest. When designing a system to detect human falls, the visual clues shown in these video sequences are crucial.

In the realm of literature, methods for detecting falls in surveillance videos aim at extracting the contour and configuration of individuals to identify their atypical movements. Nevertheless, the efficiency of these approaches is affected by the presence of a person's shadow and the perspective from which they are viewed. Automatic spatio-temporal feature extraction from massive datasets is achieved by deep learning approaches utilizing convolutional neural networks (CNNs) and long short-term memory (LSTM) architectures.

The issue of identifying fall events in compressed videos has been explored by numerous researchers in the existing literature. However, these methods suffer from limitation that arises due to the utilization of keyframes set at fixed intervals and only the P frames, which results in reduction of the speed of processing and visual quality of videos in the compression domain [5]. Further, some sources do not permit setting up a chosen codec, keyframe interval, or frame type.

The objective of the study in this paper is to develop a novel method for human fall event detection in a video under a compression format, which overcomes the limitations and yields improved results. The MPEG-4/H.264 videos are examined for the validation of this method. The main contributions in this study are:

1. Fall event detection algorithm, termed as FMVCNN, using motion vectors in the video codecs (MPEG-4/H.264), TPH-YOLOv5 followed by fuzzy based tracker, which supports (i) variable keyframes intervals, (ii) P/B frames, and (iii) scale variations of objects.
2. The validation of FMVCNN by experimentation on the videos of fall event datasets: Le2i, UR, and Multiple Cameras, in terms of performance metrics: precision, recall, F-score, speed, parameters and GFLOPS.
3. Ablation study comprises the performance analysis of FMVCNN implemented on two codecs: MPEG-4 (P frames) and H.264 (P/B frames), and that of FMVCNN variants, which are compared with the recent SOTA methods.

The organization paper comprises five sections. Section 2 provides an overview of the recent related work. Section 3 outlines the proposed FMVCNN algorithm for detecting fall events. Section 4 presents the experimental results and analysis. Finally, Section 5 presents the conclusions.

## 2. Related Work

Fall detection [6] involves recognizing when a person has fallen, with the goal of triggering an alert, such as notifying a caregiver or dispatching an ambulance. Approaches to fall detection are typically divided into sensor-based [7] and vision-based [6] methods. Although vision-based techniques provide extensive information, their effectiveness has been limited by computational constraints and algorithmic difficulties until recent advancements. With the advent of deep learning networks, the focus of fall detection research is shifted more towards vision-based solutions.

Fall detection differs from typical video classification tasks, as it requires rapid identification of potential falls in real-time video streams. To achieve this, the model must generate intermediate outputs. A widely-used technique involves applying a sliding window to analyze frame segments and detect falls. Yu et al. [8] were among the first to apply convolutional neural networks (CNNs) for fall detection, extracting binary silhouettes from each frame and performing pose classification to detect falls. Le et al. [9] opted to extract features from wearable devices rather than relying on poses. Adrián Núñez-Marcos et al. [10] proposed a fall detection solution using transformers, which analyze video clips to determine fall occurrences with a sliding-window approach for real-time alerts. Deep Feature Fusion with Computer Vision for Fall Detection and Classification (DFFCV-FDC) [11] technique utilizes Gaussian filtering for noise reduction and combines MobileNet, DenseNet, and ResNet for deep feature fusion. It demands considerable computational resources and not well-suited for real-time applications in dynamic environments. Yue Wang et al. [12] proposed fall detection algorithm using a single webcam, optimizing

precision and efficiency by combining background subtraction with the BlazePose human pose estimation model. Nicolas Thome et al. [13] devised a fall detection algorithm employing a multiview approach, in which motion is represented through a layered hidden Markov model (LHMM). Caroline Rougier and colleagues [14] introduced a technique centered on shape matching to effectively track an individual's silhouette throughout the video sequence using Gaussian mixture model. Lina Wu and colleagues [15] introduced an unsupervised fall detection method using GAN that leverages human pose images to reduce background noise, thereby safeguarding privacy. To identify human actions in compressed videos using BiLSTM, Praveenkumar et al. [16] proposed an attention-guided method.

### 2.1. Video Compression

Compression of digital video entails the elimination of both spatial and temporal redundancies within a video, thereby decreasing the number of bits required per frame. The compressed video is suitable for saving storage and transmission costs. A standard video codec consists of a series of intra (I) frames and inter frames, where an I frame is succeeded by a series of P or B frames as inter frames. An I frame provides a comprehensive color representation (e.g. RGB) in its entirety, while a P/B frame captures only the variations in pixel values relative to the frames that come before or after it. The P frames consist of the motion vectors and residuals that are predicted from prior I or P frames, whereas the B frames are made up of motion vectors and residuals predicted from both preceding and subsequent frames. B frames demonstrate a superior compression ratio relative to P frames, as they leverage information from both preceding and subsequent frames for predictive analysis. Motion vectors denote a specific pixel block within a frame that correspond to the related area in the subsequent frame. The residual indicates the discrepancy in motion prediction [17].

## 3. Proposed Methodology

### 3.1. FMVCNN

The proposed method FMVCNN facilitates real-time human action recognition related to human fall events in compressed video. Transformer Prediction Heads-You Only Look Once version five (TPH-YOLOv5) [18] is utilized for analyzing spatio-temporal features in videos, specifically the motion vectors, which detects objects of varying sizes. The fall event relates to the fall of a pedestrian. The implementation focuses on the MPEG-4 and H.264 compressed domains.

The methodology encompasses the detection of fall events, the fuzzy logic based prediction of event boxes, and the association of data, as depicted in Fig. 1. Tracking an object entails two fundamental steps: prediction and update. During prediction steps the event box detection is done, wherein the recognition of action $A^t$ at time t is achieved by leveraging the preceding action $A^{t-1}$ from time t-1, in conjunction with the fuzzy subset of motion vectors present in frame $F^t$ at time t. The update step on each frame is done by action recognition in I frames at regular k-spaced intervals, which resets the accumulated error in order to detect persons (objects) that appear/disappear from the scene captured in the frame. The TPH-YOLOv5 identifies the action $D^t$. The process of matching detected events that have confidence scores surpassing the threshold $\theta_c$ with predicted event boxes in $A^t$ occur during the data association phase. At this stage, the calculation of all pairwise intersection-over-union (IoU) cost measures is done, and the Hungarian method [19] is employed to match detected event actions with the predicted event actions. Two event boxes are considered matched when the Intersection over Union (IoU) distance between them exceeds the threshold value denoted as $\theta_{IoU}$

### 3.2. Motion Vectors

In a compressed domain, the video encoding constitutes some key frames that are in full form and a sequence of several non-key frames intervening every pair of consecutive key frames. Every non-key frame is divided into macroblocks. For each macroblock, a motion vector is calculated that directs to a macroblock with analogous characteristics in the subsequent frame (denoted by red arrows in Fig. 2). Thus, information content in the form of motion vectors is an efficient video encoding. MPEG-4 features macroblocks that are 16 × 16 pixels in size, whereas H.264 utilizes macroblocks that can be either 4 × 4 or 16 × 16 pixels. A sequence of video frames In the interval between two successive key frames, one might encounter frames of P or B type. A non-key frame is

classified as P type if motion vectors pertain only to preceding frames, while it is classified as B type if motion vectors reference non-key frames from both preceding and succeeding frames. In MPEG-4, a non-key frame is solely of type P, while in H.264, it can be of P or B type. The motion vectors serve as representations of moving objects within a frame and are utilized in the recognition of human actions visually.

In an encoded video, motion vectors are calculated and accessible within its codec, such as MPEG-4 or H.264[5]. A motion vector in the frame $F^t$ at time t is represented by the ten-tuple

$$m_j^t = (s_j^t,\ v_{j,w}^t,\ v_{j,h}^t,\ x_{j,src}^t,\ y_{j,src}^t,\ x_{j,dst}^t,\ y_{j,dst}^t, v_{j,x}^t,\ v_{j,y}^t,\ v_{j,s}^t) \tag{1}$$

which possesses the reference frame $F^{t+s_j^t}$ characterized by the offset $s_j^t$ in relation to $F^t$. In a P frame, $s_j^t$ takes on a negative value, while for a B frame, it is positive. The variables $v_{j,w}^t$ and $v_{j,h}^t$ represent the width and height of the macroblock, respectively, in relation to the motion vector. Figure 3 illustrates the computation of motion vectors. A motion vector in $F^t$ originates from the center $(x_{j,dst}^t, y_{j,dst}^t)$ of its associated macroblock and directs towards the center $(x_{j,src}^t, y_{j,src}^t)$ of the macroblock in the reference frame that exhibits a comparable appearance. Additionally, $v_{j,x}^t$ and $v_{j,y}^t$ represent the x- and y-components of $m_j^t$, which are scaled by $v_{j,s}^t$ in accordance with the following relations:

$$\begin{aligned} x_{j,src}^t &= \lfloor x_{j,dst}^t + v_{j,x}^t / v_{j,s}^t \rfloor, \\ y_{j,src}^t &= \lfloor y_{j,dst}^t + v_{j,y}^t / v_{j,s}^t \rfloor \end{aligned} \tag{2}$$

Generally, the quantity $|M^t|$ of motion vectors within the set $M^t \subset Z^{10}$ of motion vectors in $F^t$ at time t may fluctuate from one frame to another. For key frames, it holds that $M^t = \emptyset$, i.e. $M^t = 0$.

### 3.3. Handling of Key Frames

The absence of motion vectors in key frames presents challenges for action recognition that relies on motion vectors when these frames appear in a compressed video frame sequence. In the proposed method, a novel technique is utilized for managing keyframes during action recognition and update steps, occurring at intervals of length k, regardless of the frame type (key or non-key frame) encountered. In the action recognition process, for a key frame, set $M^t = M^{t-1}$; that is, the motion vectors from the preceding frame are utilized for action recognition computation. This holds true under the assumption that the change between two consecutive frames due to object motion is minimal, leading to the conclusion that $M^{t-1} \cong M^t$. Consequently, $M^{t-1}$ is employed for action recognition when $M^t$ for a key frame is unavailable

### 3.4. Handling of Key Frames

In the procedure for event box prediction, the bounding boxes $B^t$ are generated using information from both the past predicted boxes $B^{t-1}$ and present motion vectors $M^t$. The core concept involves a fuzzification process, wherein the adjustment of the centre of each box in $B^{t-1}$ takes into account the fuzzy set representation of x and y components of motion vectors originating within the box. Despite accounting for scale changes in the boxes, a notable enhancement in tracking performance persists, showcasing the efficacy of the fuzzy logic based tracking methodologies in achieving improved tracking speed and robustness. The algorithm is structured to address scenarios involving frames of either P type exclusively or P/B type. The process includes the following steps:

1. Define the subset $\widetilde{M}^t \subset M^t$ consisting of motion vectors that possess a non-zero magnitude.

$$\widetilde{M^t} = \{m_j^t \in M^t | ((v_{j,x}^t)^2 + (v_{j,y}^t)^2)^{1/2} > 0\}. \tag{3}$$

2. The temporal distance $s_j^t$ is used to rescale the x- and y-components of motion vectors $m_j^t \in \widetilde{M}^t$:

$$\overline{v}_{j,x}^t = v_{j,x}^t / s_j^t, \quad \overline{v}_{j,y}^t = v_{j,y}^t / s_j^t. \tag{4}$$

3. Obtain the floating-point resolution using motion scale $v_{j,s}^t$:

$$\tilde{v}_{j,x}^t = \overline{v}_{j,x}^t / v_{j,s}^t,\ \tilde{v}_{j,y}^t = \overline{v}_{j,y}^t / v_{j,s}^t. \tag{5}$$

4. Establish the new reference points Vectors

$$\tilde{x}_{j,src}^t = \text{Rescale}_x(x_{j,src}^t),$$

$$\tilde{y}_{j,src}^t = \text{Rescale}_y(y_{j,src}^t)$$

employing rescaled x- and y-components:

$$\tilde{x}_{j,src}^t = x_{j,dst}^t + s_j^t \tilde{v}_{j,x}^t,\ \tilde{y}_{j,src}^t = y_{j,dst}^t + s_j^t \tilde{v}_{j,y}^t \tag{6}$$

5. Fuzzification: For each action $a_i^{t-1} = (x_i^{t-1},\ y_i^{t-1}, w_i^{t-1},\ h_i^{t-1})$ in $A^{t-1}$, construct the fuzzy subset $\widehat{M}_i^t$ of motion vectors in $\widetilde{M}^t$ with respect to new reference point $(\tilde{x}_{j,src}^t,\ \tilde{y}_{j,src}^t)$

$$\begin{aligned} \widehat{M}_i^t = \{m_j^t \in \widetilde{M}^t | x_i^{t-1} - \frac{w_i^{t-1}}{2} \le \tilde{x}_{j,src}^t \le x_i^t - 1 + \frac{w_i^{t-1}}{2} \\ \wedge\ y_i^{t-1} - \frac{h_i^{t-1}}{2} \le \tilde{y}_{j,src}^t \le y_j^{t-1} + \frac{h_i^{t-1}}{2}\} \end{aligned} \tag{7}$$

with membership values given by

$$\mu_{\widehat{M}_i^t}(m_j^t) = min\{\mu_{v_x}(\tilde{v}_{j,x}^t), \mu_{v_y}(\tilde{v}_{j,y}^t)\} \tag{8}$$

where $\tilde{v}_{j,x}^t$ and $\tilde{v}_{j,y}^t$ are the fuzzy components of the motion vectors $m_j^t$ in $\widetilde{M}^t$ with membership values given by the Gaussian membership functions $\mu_{v_x}(.)$ and $\mu_{v_y}(.)$, respectively. The mean and standard deviation of the Gaussian function are computed as the mean and standard deviation of the x and y components of motion vectors in the event box given in eq. (7).

6. Defuzzification: For each fuzzy subset $\widehat{M}_i^t$, the defuzzification is done and the defuzzified motion vector is computed as:

$$m_j{}^{t*} = argmax_{m_j^t \in \widehat{M}_i^t}\{\mu_{\widehat{M}_i^t}(m_j^t)\} \tag{9}$$

which is the motion vector for which the membership function value is maximum, and its x and y components are denoted by $\tilde{v}_{j,x}^{t*}$ and $\tilde{v}_{j,y}^{t*}$.

7. Prediction the new event bounding box $a_i^t = (x_i^t, y_i^t, w_i^t, h_i^t)$ in $A^t$ by executing a shift operation on the center of the corresponding previous event box $a_i^{t-1}$ utilizing the defuzzified values of the motion vector components in $\widehat{M}^t$:

$$\begin{aligned} x_i^t = x_i^{t-1} + \tilde{v}_{j,x}^{t*},\ y_i^t = y_i^{t-1} + \tilde{v}_{j,y}^{t*}, \\ w_i^t = w_i^{t-1},\ h_i^t = h_i^{t-1} \end{aligned} \tag{10}$$

The aforementioned procedure is applicable to video codecs utilizing both P and B frames. Following the rescaling of motion vector components in step (b), a $\pi$-rotation operation is executed on the vectors in P. This ensures that both P and B frames have vectors aligned in the same direction, allowing for analogous processing in the subsequent computational steps.

### 3.5. Data Association by Hungarian Method

The Hungarian method is used to the process of assignment of predicted action $A_i^{t+k}$ to detected action $D_j^{t+k}$ optimally with the cost $C_{A_i^{t+k} D_j^{t+k}}$ of assignment of predicted action $A_i^{t+k}$ to detected action $D_j^{t+k}$ being measured in terms of Intersection-over-Union (IoU) of the bounding boxes of $i^{th}$ action $A_i^{t+k}$ and $j^{th}$ action $D_j^{t+k}$. Thus, the cost matrix is given as:

$C_{A_i^{t+k} D_j^{t+k}}$ = -IoU of bounding boxes of $A_i^{t+k}$ and $D_j^{t+k}$,

where the minus sign signifies minimization of cost (= -IoU), i.e. maximization of IoU. This algorithm filters out matches with low similarity based on the specified threshold, ensuring that only pairs with low costs (high similarity) are considered as valid matches. The algorithm returns the matched pairs, unmatched predicted actions, and unmatched detected actions. The matched pairs are used for updated actions shown in the Fig.1.

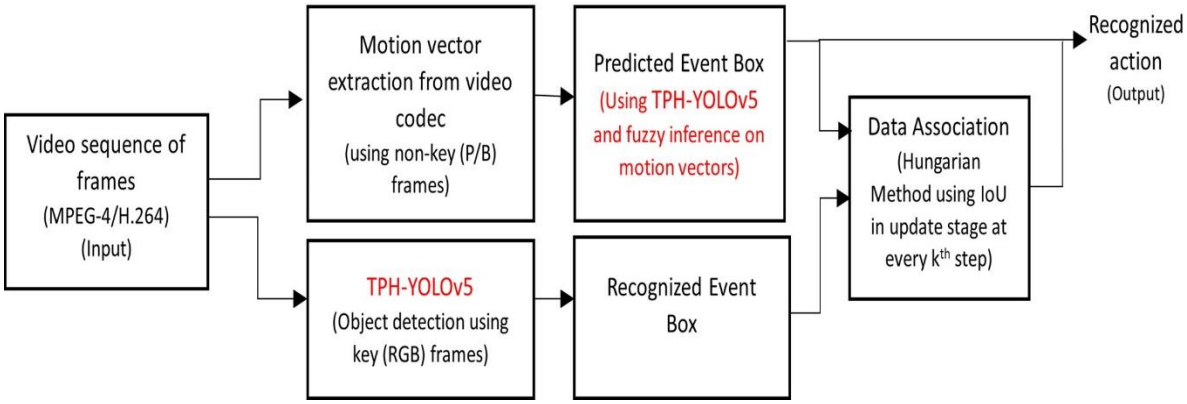


Figure 1: Architecture diagram of the proposed FMVCNN

a. b.

Figure 2: Sample video frames: (a) MPEG-4 (motion vectors in red (for P)), (b) H.264 (motion vectors in red (for P) and green (for B)).

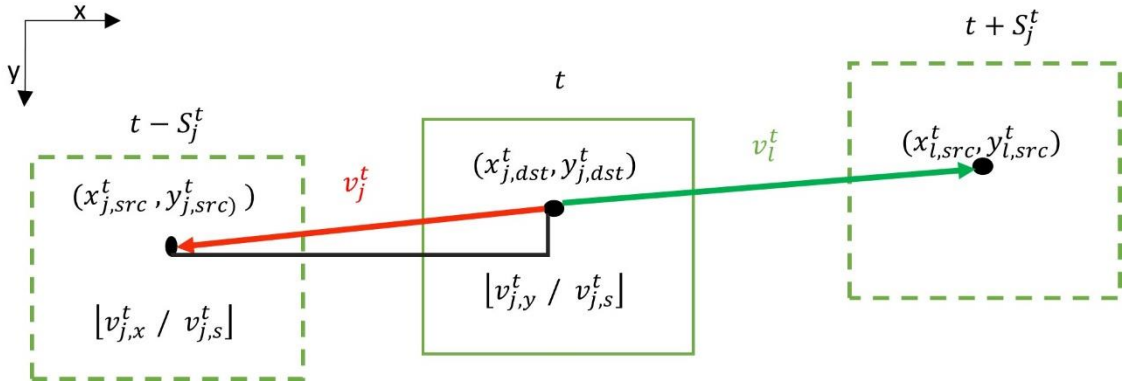


Figure 3: An inter-coded macroblock (solid border) in MPEG-4/H.264 P frame at time t features a motion vector $v_j^t$, which directs to a sub-image exhibiting similar characteristics in the preceding frame at $t - s_j^t$. In H.264, macroblocks utilize motion vectors that direct to a corresponding sub-image at $t + s_l^t$ in the subsequent frame.

## 4. Results and Discussion

### 4.1. Dataset

The proposed FMVCNN approach for fall event identification is implemented on publically accessible benchmark fall event video datasets, specifically Le2i [20], UR [21], and Multiple Cameras [23]. The attributes of these datasets are presented in Table 1. Le2i contains 143 video clips depicting fall events and 48 video clips depicting non-fall events, occurring across four scene types:residential space, coffee shop, workplace, and classroom. The dataset of Multiple Cameras includes 22 fall events along with 2 mixed daily occurrences. The view synchronization of videos from Multiple Cameras can used for 3D scene reconstruction for detection of fall events. The UR Dataset consists of films captured using the Kinect sensor, resulting in synchronized recordings of RGB and depth data. The recognition velocity is assessed in frames per second (FPS). The training and testing sets are created from the benchmark datasets in an 80:20 ratio. The proposed FMVCNN is executed on an Intel Core i7 CPU (3.7GHz, 64 GB RAM) with Python programming on the Ubuntu 20.04 platform. The ablation study is conducted on the identical platform. Each sample video from the training set is divided into segments of 12 frames, consistent with previous relevant studies. Each frame is scaled to 255 x 255 for training the FMVCNN network. On average, an MPEG-4 video contains 11 P frames after each I frame, and an H.264 video comprises 11 P/B frames succeeding each I frame. A sample video in the test set has 15 clips, with each clip consisting of a 12-frame sequence, specifically one I frame followed by 11 P frames for MPEG-4, or 11 P/B frames for H.264. Each clip in the train/test set is cropped randomly from the videos containing the target actions in the benchmark datasets considered.

***Compute Resource Evaluation***

The estimated computational requirements of the proposed FMVCNN for Fall event action recognition are: 15.6 GFLOPS and 7.0[M] parameters that are involved. During the inference stage, the execution time for frame loading and then motion vector extraction from frame is $O(1.5ms/frame)$. The FMVCNN and its variants in the ablation study are trained using Google colab GPU (Tesla T4).

### 4.2 Ablation Study

The ablation study of the proposed FMVCNN involves the following four aspects:

1. Fuzzification of set of motion vectors using Gaussian, Triangular, and Trapezoidal membership functions.
2. Two currently popular codecs: MPEG-4 and H.264.
3. Performance analysis by using three benchmark datasets, namely, Le2i, UR and Multiple Cameras, for implementation.
4. Comparing the FMVCNN Model with its variants, that is TPH-YOLOv5 without fuzzification and YOLO variants.

#### 4.2.1 FMVCNN Variants with Different Fuzzification Methods

In Table 2, the performance of the proposed FMVCNN implemented in MPEG-4 and H.264 compressed domains is evaluated, with a specific focus on comparing the different methods of fuzzification of motion vector components using Gaussian, Triangular, and Trapezoidal membership functions. Notably, it is evident that Gaussian fuzzification has yielded significant improvement over the other two fuzzification techniques. This enhancement can be attributed to its ability to capture the underlying distribution characteristics of motion vectors, which is crucial in the context of fall event detection in videos. Gaussian fuzzification, by modeling the motion vectors according to Gaussian distributions of the x, and y components, effectively accounts for the subtleties in motion patterns, leading to more accurate and robust fall detection results as compared to alternative fuzzification methods, in terms of F1-Score and speed(FPS).

#### 4.2.2 Compression Domain Variants

The efficacy of FMVCNN (utilizing Gaussian, Triangular, and Trapezoidal fuzzification variations) is assessed across several codecs, specifically, (i) MPEG-4 (P frames) and (ii) H.264 (P/B frames), by computing F1-Score (%) and processing speed (FPS) for event detection. The results in Table 2 indicate that the proposed algorithm

exhibits superior performance in the MPEG-4 domain, achieving a higher F1-Score and speed compared to the H.264 domain. In H.264, the F1-Score exhibits a slight enhancement due to the Gaussian fuzzification version as compared to MPEG-4 with triangular and trapezoidal fuzzification variants, albeit with a corresponding decrease in speed. The detection speed of fall events in MPEG-4 video surpasses that of H.264 due to (i) the smaller macroblock size in H.264 frames compared to MPEG-4, and (ii) MPEG-4's exclusive use of P frames, whereas H.264 incorporates both P and B frames, leading to increased motion vector computations for event box prediction in each frame. The observations from the experimental results are consistent across all three datasets examined.

#### 4.2.3 FMVCNN versus the State-of-the-art Methods

In Tables 3, 4, and 5, the results of proposed FMVCNN are compared with that of the state-of-the-art methods implemented on UR, Multiple Cameras and Le2i. The '-' indicates that the corresponding results are not reported in the literature. The accuracy of action recognition is found to depend on the (i) model architecture, (ii) training dataset, and (iii) video clip size with/without compression. The tabulated experimental results show enhanced performance of FMVCNN in terms of recognition accuracy, model parameters, GFLOPS, and recognition speed. For the purpose of comparison, only the Gaussian fuzzification variant of the proposed FMVCNN is considered, since it has yielded superior results as compared to other fuzzification variants, which is observed in Table 2.

The performance analysis of the proposed method for identifying fall event actions in comparison to other approaches using the UR dataset is presented in the Table 3, and that using Multiple Camera dataset in the Table 4, also that using Le2i in the Table 5. It is observed that the proposed method has exhibited superior performance in terms of the performance matrix reported in the literature.

Table 1: Overview of Video Specifications in Le2i [20], Multiple Camera [22], and UR [21] Datasets.

| Specification | Le2i [20] | Multiple Camera [22] | UR [21] |
|---|---|---|---|
| Total Videos | 191 | 192 | 70 |
| Videos with Falls | 143 | 176 | 30 |
| Video Resolution | 320 × 240 pixels | 720 × 480 pixels | 640 × 480 pixels |
| Frame Rate (FPS) | 25 | 30 | 25 |
| Scene Setting | Indoor (individual) | Indoor (individual) | Indoor(individual) |

However, it is important to note that the above methods that are used for comparison in the Tables 3, 4 and 5 work on non-compressed domain which requires extra decoding, and hence it is more time-consuming and less suitable for real-time applications due to its reliance on non-compressed video data. The proposed FMVCNN method is evaluated in terms of F1-Score against prominent compressed domain techniques in Table 6, utilizing the UR, Multi Cameras, and Le2i fall event datasets for comparison. The proposed method is found to yield high F1-Score and exhibit robust performance across the three datasets.

The findings of the fall event action detection in compressed domains, using the Le2i Dataset that encompasses solitary fall event scenarios, are illustrated in Fig. 4. Figure 4 illustrates the identification of 'walking' (a) and 'fallen' (c) actions in the MPEG-4 domain, as well as 'walking' (b) and 'fallen' (d) actions in H.264, accompanied by the extraction of motion vectors. Figure 5 presents the sample results of fall event detection utilizing the Multi Camera dataset. Figure 5 illustrates the identification of 'walking' (a), 'fallen' (c) actions in the MPEG-4 domain, as well as 'walking' (b) and 'fallen' (d) actions in the H.264 format, accompanied by the extracted motion vectors. For the UR dataset, Figure 6 illustrates the identification of 'walking' (a), 'fallen' (c) in the MPEG-4 domain, as well as 'walking' (b) and 'fallen' (d) actions in H.264, accompanied by the extraction of motion vectors.

Table 2: The Lei2 Fall, Multiple Camera Fall, and UR Fall datasets were used to evaluate different versions of the proposed FMVCNN algorithm in MPEG-4 and H.264 formats.

| Dataset | Proposed method FMVCNN | | F1-Score (%) | Speed (FPS) |
|---|---|---|---|---|
| | Fuzzification variants | Codec variants | | |
| UR Fall Detection Dataset | Gaussian Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 99.06<br>96.25 | 524.0<br>405.0 |
| | Triangular Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 95.00<br>93.25 | 482.0<br>385.0 |
| | Trapezoidal Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 96.25<br>94.46 | 425.0<br>305.0 |
| Camera Fall Dataset Multiple | Gaussian Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 97.01<br>95.3 | 502.0<br>396.0 |
| | Triangular Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 94.85<br>93.04 | 455.0<br>325.0 |
| | Trapezoidal Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 93.25<br>92.14 | 405.0<br>295.0 |
| Lei2Fall Detection Dataset | Gaussian Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 99.52<br>95.21 | 495.0<br>325.0 |
| | Triangular Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 94.25<br>92.23 | 425.0<br>296.0 |
| | Trapezoidal Fuzzification | MPEG-4 (P)<br>H.264 (P, B) | 92.45<br>90.43 | 386.0<br>254.0 |

The proposed FMVCNN demonstrates superior performance compared to other prominent methods, achieving improvements in accuracy, sensitivity, specificity, F1-Score, and speed. Additionally, there has been a notable decrease in computational expenses measured in number of model parameters[M] and GFLOPS.

### 4.3. Dataset

Table 7 presents a performance comparison of five models, namely:

1. Model I: The proposed FMVCNN (Motion vectors + TPH-Yolov5 + Fuzzy Logic)
2. ModelII: TPH-YOLOv5 without fuzzification (Motion vectors+TPH-YOLOv5+ Mean/Median)
3. Model III: YOLOv5 without fuzzification (Motion vectors + YOLOv5 + Mean/Median)
4. Model IV: YOLOv8 without fuzzification (Motion vectors + YOLOv8 + Mean/Median)
5. Model V: YOLOv9 without fuzzification (Motion vectors + YOLOv9 + Mean/Median)

which are trained and tested on UR, Multiple Cameras, and Le2i Fall Detection Datasets. Notably, the Models II, III, IV and V use either mean or median variant as a tracker. Further, this comparison encompasses two distinct codecs: MPEG-4 and H.264. The results displayed in Table 7 yield several key observations.

The Table 7 shows the comparison of performance of the proposed fuzzy inference based method FMVCNN (Model I) with that of the Models II-V based on different YOLO variants and non fuzzy approach. It is observed that the TPH-YOLOv5 with fuzzy inference for event box prediction (Model I) has yielded higher accuracy in terms of F1-Score as compared to the non-fuzzy approach based Models II-V.

Further, it is significant to note that the advanced YOLO architectures YOLOv8 and YOLOv9 yield marginal increase in F1-Score, as compared to TPH-YOLOv5 (Model II), but at the cost of considerable decrease in speed, which makes these architectures unsuitable for real-time applications. This can be attributed to the fact that the TPH-YOLOv5 is light weight architecture compared to the advanced YOLOv8 and YOLOv9. Thus, the primary aim being the real-time action recognition in compressed videos, the proposed method FMVCNN, being the light-weight and low cost in terms of GFLOPS and parameters, is suitable for real-time applications.

Table 3: An analysis of the performance of the proposed method for identifying fall event actions in comparison to other approaches using the UR dataset.

| Method/Author | Domain | Accuracy (%) | F1-Score (%) |
|---|---|---|---|
| Transformer-based (jointly fine-tuned, w/o oversampling) (2024) [10] | Non-Compressed | 91.3 | 89.73 |
| Transformer-based (jointly fine-tuned, w/ oversampling) (2024) [10] | Non-Compressed | 95.45 | 94.76 |
| DFFCV-FDC (2024)[11] | Non-Compressed | 96.36 | 92.72 |
| YOLO V4 (2023) [23] | Non-Compressed | 95.7 | 85.4 |
| Attention-guided LSTM (2023) [23] | Non-Compressed | 96.6 | 94.6 |
| Attention-guided Bi-LSTM (2023) [23] | Non-Compressed | 96.9 | 95.8 |
| Yue Wang et al. (2024) [12] | Non-Compressed | 89.99 | 90.02 |
| Khalili et al. (2022) [24] | Non-Compressed | 95 | - |
| Alam et al. (2023) [25] | Non-Compressed | 84.38 | - |
| Lin et al. (2022) [26] | Non-Compressed | 91.1 | - |
| Qian et al. (2022) [27] | Non-Compressed | 94.88 | - |
| Lian Wu et al. (2024) [15] | Non-Compressed | 88.5 | - |
| Proposed Motion vectors + TPH-Yolov5 + Fuzzy logic | Compressed | 99.6 | 99.06 |

Table 4: Evaluation of the proposed method's efficacy relative to alternative employing a multi-camera dataset.

| Method/Author | Domain | Accuracy (%) | F1-Score (%) | Sensitivity (%) | Specificity (%) |
|---|---|---|---|---|---|
| Attention-guided LSTM (2023) [23] | Non-Compressed | - | - | 91.8 | 94.8 |
| Attention-guided Bi-LSTM (2023) [23] | Non-Compressed | - | - | 92 | 96 |
| STFT and 1D-CNN (2024) [28] | Non-Compressed | 91 | 88 | 87 | 91 |
| CNN (2023) [29] | Non-Compressed | - | - | 66 | 95 |
| OpenPose-SVM (2021) [30] | Non-Compressed | - | - | 95 | 92.5 |
| DFFCV-FDC (2024) [11] | Non-Compressed | 96.82 | 95.45 | - | - |
| Proposed Motion vectors + TPH-Yolov5 + Fuzzy logic | Compressed | 98.2 | 97.01 | 98.36 | 99.2 |

The Table 7 shows the comparison of performance of the proposed fuzzy inference based method FMVCNN (Model I) with that of the Models II-V based on different YOLO variants and non fuzzy approach. It is observed that the TPH-YOLOv5 with fuzzy inference for event box prediction (Model I) has yielded higher accuracy in terms of F1-Score as compared to the non-fuzzy approach based Models II-V.

Table 5: Comparison of the performance of the proposed Fall event action detection approach against other methods utilizing the Le2i dataset.

| Method/Author | Domain | Accuracy(%) | FI-Score (%) |
|---|---|---|---|
| Yue Wang et. al. (2024) [12] | Non-Compressed | 89.99 | 90.02 |
| Fall-GCN (2024) [31] | Non-Compressed | 98.50 | - |
| H. Eruder et. al. (2024) [31] | Non-Compressed | 98.95 | 98.95 |
| Jiangjiao Li et. al. (2024) [33] | Non-Compressed | 99.33 | - |
| Yuan et.al. (2022) [34] | Non-Compressed | 98.43 | - |
| Lian Wu et.al. (2024) [15] | Non-Compressed | 91.60 | - |
| Proposed Motion vector + TPH-Yolov5+ Fuzzy Logic | Compressed | 99.60 | 99.52 |

Table 6: Comparative analysis of the proposed fall event action detection method against state-of-the-art techniques in the compressed domain using the UR, Multiple Cameras, and Le2i datasets, focusing on F1 score

| Method | UR | Multiple Cameras | Le2i |
|---|---|---|---|
| HOF+MBH (C Vishnu et al.) (2021) [35] | 97.1 | - | 99.1 |
| Deep Features (Mahta Darvish et al.) (2023) [36] | 95 | 94 | - |
| MVCNN (Praveenkumar et al.) (2024) [37] | 93.35 | 91.25 | 89.06 |
| Proposed Motion vectors + TPH-Yolov5+ Fuzzy logic | 99.06 | 97.01 | 99.52 |

Table 7: Comparative analysis of the proposed FMVCNN approach for fall event action identification with its modifications.

| Fall Event Dataset | Model I (Proposed Model)<br>Proposed Motion vectors + TPH-Yolov5+ Fuzzy logic<br>Parameters[M]=7 GFLOPs=15.7 | | | | Model II<br>Motion vectors + TPH-Yolov5+ Mean/Median)<br>Parameters[M] =7 GFLOPs=15.6 | | Model III<br>Motion vectors + TPH-Yolov5+ Mean/Median)<br>Parameters[M]=7.2 GFLOPs=16.6 | | Model IV<br>Motion vectors + TPH-Yolov8+ Mean/Median)<br>Parameters[M]=11.1 GFLOPs=28.6 | | Model V<br>Motion vectors + TPH-Yolov9+ Mean/Median)<br>Parameters[M] =25.4 GFLOPs=103.2 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Variants (Fuzzy, Domain) | FI-Score (%) | Speed (FPS) | Variants (Mean /Median, Domain) | FI-Score (%) | Speed (FPS) | FI-Score (%) | Speed (FPS) | FI-Score (%) | Speed (FPS) | FI-Score (%) | Speed (FPS) |
| UR | Gaussian MPEG-4 (P) | 99.06 | 524 | Mean, MPEG-4 (P) | 96.8 | 422 | 95.05 | 410 | 97.85 | 396 | 98.01 | 305 |
| | Gaussian H.264 (P, B) | 96.25 | 405 | Median H264(P.B) | 92.15 | 162 | 91 | 152 | 94.25 | 168 | 95.15 | 125 |
| Multi-Camera | Gaussian MPEG-4 (P) | 97.01 | 502 | Mean, MPEG-4 (P) | 95.25 | 319 | 93.25 | 302 | 96.45 | 256 | 96.58 | 215 |
| | Gaussian H.264 (P, B) | 95.3 | 396 | Median H264(P.B) | 93.3 | 224 | 91.3 | 201 | 94.05 | 135 | 94.55 | 110 |
| Le2i | Gaussian MPEG-4 (P) | 99.52 | 495 | Mean, MPEG-4 (P) | 96.06 | 408 | 91.06 | 396 | 97.95 | 310 | 98.25 | 296 |
| | Gaussian H.264 (P, B) | 95.21 | 325 | Median H264(P.B) | 91.21 | 250 | 87.21 | 242 | 93.21 | 222 | 94.58 | 155 |


a b c d

Figure 4: Results obtained from the Le2i dataset: 'walking' and 'fallen' actions are presented in (a,c) MPEG-4 format and (b,d) H.264 format

a b c d

Figure 5: Results from the Multiple Cameras dataset: 'walking'/'fallen' actions observed in (a,c) MPEG-4 format and (b,d) H.264 format.

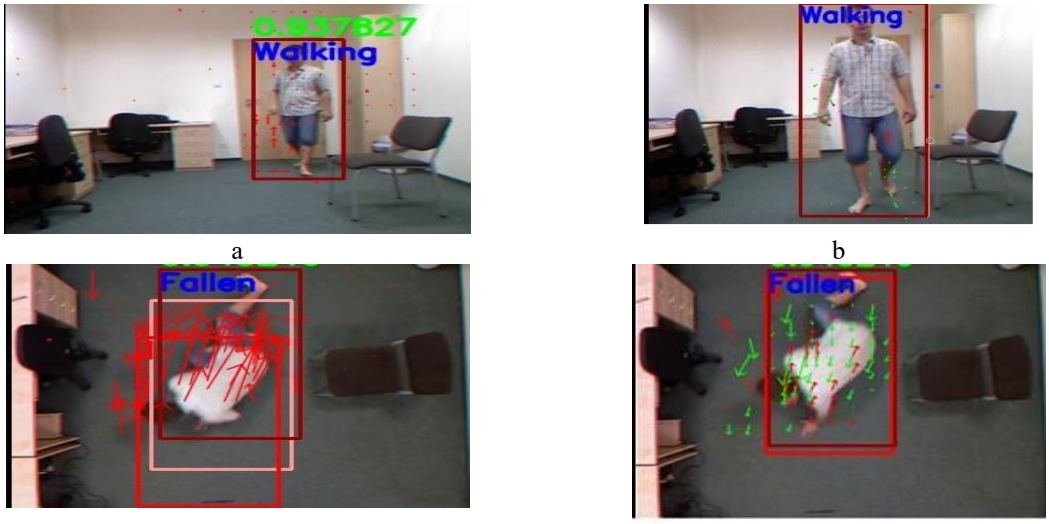


Figure 6: The UR dataset's sample findings include "walking" and "falling" motions in (a,c) MPEG-4 and (b,d) H.264

### 4.4 Limitation of Proposed Method

The accuracy of FMVCNN on H.264 is slightly lower, and its speed of action recognition is slower compared to MPEG-4. The slow speed (measured in FPS) in H.264 for action recognition is attributed to the generation of a large number of motion vectors per frame. Notably, B-frames in H.264 contain approximately twice as many motion vectors as P frames, contributing to the overall decrease in speed. H.264 is lossy compression codec, with higher compression rate compared to MPEG-4, leading to greater loss of visual information. This information loss affects the important details like motion patterns, textures, and subtle changes in appearance features. H.264 uses quantization to represent colors and intensities with fewer bits. This quantization process often leads to noise and distortions in the image, which impact the accuracy of action recognition algorithms.

### Refrences

[1] United Nations, World Population Ageing: Highlights. United Nations, 019.
[2] L. M. Dang et al., "Sensor-based and vision-based human activity recognition: A comprehensive survey," Pattern Recognition Letters, vol. 108, p. 107561, 2020.
[3] M. Saleh, M. Abbas, and R. L. B. Jeannès, "Fallalld: An open dataset of human falls and activities of daily living for classical and deep learning applications," IEEE Sensors Journal, pp. 1–1, 2020.
[4] M. Saeidi and A. Ahmadi, "A novel approach for deep pedestrian detection based on changes in camera viewing angle," Signal, Image and Video Processing, vol. 14, pp. 1–9, 2020.
[5] L. Bommes, X. Lin, and J. Zhou, "MVmed: Fast multi-object tracking in the compressed domain," in Proc. 15th IEEE Conf. Industrial Electronics and Applications (ICIEA), 2020, pp. 1419–1424. doi: 10.1109/ICIEA48937.2020.9248145.
[6] E. Alam, A. Sufian, P. Dutta, and M. Leo, "Vision-based human fall detection systems using deep learning: A review," Computers in Biology and Medicine, vol. 146, p. 105626, 2022.
[7] S. Nooruddin et al., "Sensor-based fall detection systems: A review," Journal of Ambient Intelligence and Humanized Computing, vol. 12, pp. 1–17, 2021.
[8] M. Yu, L. Gong, and S. Kollias, "Computer vision based fall detection by a convolutional neural network," in Proc. 19th ACM Int. Conf. Multimodal Interaction, 2017, pp. 416–420.
[9] H. L. Le, D. N. Nguyen, T. H. Nguyen, and H. N. Nguyen, "A novel feature set extraction based on accelerometer sensor data for improving the fall detection system," Electronics, vol. 11, no. 7, p. 1030, 2022.
[10] A. Núñez-Marcos and I. Arganda-Carreras, "Transformer-based fall detection in videos," Engineering Applications of Artificial Intelligence, 2024.

[11] W. S. Almukadi, F. Alrowais, M. K. Saeed et al., "Deep feature fusion with computer vision driven fall detection approach for enhanced assisted living safety," Scientific Reports, vol. 14, p. 21537, 2024. doi: 10.1038/s41598-024-71545-6.
[12] Y. Wang and T. Deng, "Enhancing elderly care: Efficient and reliable real-time fall detection algorithm," Digital Health, vol. 10, p. 20552076241233690, 2024. doi: 10.1177/20552076241233690.
[13] N. Thome, S. Miguet, and S. Ambellouis, "A real-time, multiview fall detection system: A LHMM-based approach," IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, no. 11, pp. 1522–1532, 2008. doi: 10.1109/TCSVT.2008.2005606.
[14] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Robust video surveillance for fall detection based on human shape deformation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, no. 5, pp. 611–622, 2011. doi: 10.1109/TCSVT.2011.2129370.
[15] L. Wu et al., "Video-based fall detection using human pose and constrained generative adversarial network," IEEE Transactions on Circuits and Systems for Video Technology, vol. 34, no. 4, pp. 2179–2194, 2024. doi: 10.1109/TCSVT.2023.3303258.
[16] S. M. Praveenkumar, P. Patil, and P. S. Hiremath, "A novel algorithm for human action recognition in compressed domain using attention-guided approach," Journal of Real-Time Image Processing, vol. 20, p. 122, 2023. doi: 10.1007/s11554-023-01374-9.
[17] R. V. Babu, M. Tom, and P. Wadekar, "A survey on compressed domain video analysis techniques," Multimedia Tools and Applications, vol. 75, pp. 1043–1078, 2016. doi: 10.1007/s11042-014-2345-z.
[18] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," in Proc. IEEE/CVF Int. Conf. Computer Vision Workshops (ICCVW), 2021, pp. 2778–2788. doi: 10.1109/ICCVW54120.2021.00312.
[19] H. W. Kuhn, "The Hungarian method for the assignment problem," Naval Research Logistics Quarterly, vol. 2, pp. 83–97, 1955.
[20] 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," Journal of Electronic Imaging, vol. 22, p. 041106, 2013.
[21] B. Kwolek and M. Kepski, "Human fall detection on embedded platform using depth maps and wireless accelerometer," Computer Methods and Programs in Biomedicine, vol. 117, no. 3, pp. 489–501, 2014.
[22] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Multiple cameras fall dataset," DIRO–Université de Montréal, 2010.
[23] M. Agrawal and S. Agrawal, "Enhanced deep learning for detecting suspicious fall event in video data," Intelligent Automation and Soft Computing, vol. 36, no. 3, pp. 2653–2667, 2023. doi: 10.32604/iasc.2023.033493.
[24] S. Khalili, H. Mohammadzade, and M. M. Ahmadi, "Elderly fall detection using CCTV cameras under partial occlusion of the subject's body," arXiv preprint arXiv:2208.07291, 2022.
[25] E. Alam, A. Sufian, P. Dutta et al., "Real-time human fall detection using a lightweight pose estimation technique," in Computational Intelligence in Communications and Business Analytics. Cham, Switzerland: Springer, 2023, pp. 30–40.
[26] B. S. Lin, T. Yu, C. W. Peng et al., "Fall detection system with artificial intelligence-based edge computing," IEEE Access, vol. 10, pp. 4328–4339, 2022.
[27] Z. Qian, Y. Lin, W. Jing et al., "Development of a real-time wearable fall detection system in the context of internet of things," IEEE Internet of Things Journal, vol. 9, pp. 21999–22007, 2022.
[28] M. Dutt, A. Gupta, M. Goodwin, and C. W. Omlin, "An interpretable modular deep learning framework for video-based fall detection," Applied Sciences, vol. 14, no. 11, p. 4722, 2024. doi: 10.3390/app14114722.
[29] K. G. Gunale, P. Mukherji, and S. N. Motade, "Convolutional neural network-based fall detection for the elderly person monitoring," Journal of Advanced Information Technology, vol. 14, pp. 1169–1176, 2023.
[30] Y. Chen, R. Du, K. Luo, and Y. Xiao, "Fall detection system based on real-time pose estimation and SVM," in Proc. 2021 IEEE 2nd Int. Conf. Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), 2021.
[31] X. Ren, D. Niu, K. Ren, X. Chen, Z. Pan, and X. Chen, "A fall detection method in elevators based on graph convolutional networks," in Proc. 36th Chinese Control and Decision Conf. (CCDC), 2024, pp. 1496–1500. doi: 10.1109/CCDC62350.2024.10588331.
[32] H. Ergüder, T. Uzun, and M. Baday, "Advancing fall detection utilizing skeletal joint image representation and deformable layers," Image Analysis and Stereology, vol. 43, no. 1, pp. 97–107, 2024. doi: 10.5566/ias.3087.

[33] J. Li, M. Gao, P. Wang et al., “Fall detection algorithm based on pyramid network and feature fusion,” Evolutionary Systems, vol. 15, pp. 1957–1970, 2024. doi: 10.1007/s12530-024-09601-9
[34] 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. 2022 IEEE Int. Conf. Real-time Computing and Robotics (RCAR), 2022.
[35] C. Vishnu, R. Datla, D. Roy, S. Babu, and C. K. Mohan, “Human fall detection in surveillance videos using fall motion vector modeling,” IEEE Sensors Journal, vol. 21, no. 15, pp. 17162–17170, 2021. doi: 10.1109/JSEN.2021.3082180.
[36] M. Darvish and A. Mansouri, “Compressed domain human fall detection using deep features,” Preprint, 2023. doi: 10.21203/rs.3.rs-3085740/v1.
[37] S. M. Praveenkumar, P. Patil, and P. S. Hiremath, “Swift detection of human fall events in compressed videos,” in T. S., L. D., and S. Rajesh, Eds., Intelligent Systems in Computing and Communication, vol. 2232, Communications in Computer and Information Science (ISCComm 2023). Cham, Switzerland: Springer, 2025, doi: 10.1007/978-3-031-75608-5_18.