A Video-based Fall Detection Network by Spatio-temporal Joint-point Model on Edge Devices

Ziyi Guan*, Shuwei Li*, Yuan Cheng†, Changhai Man*, Wei Mao*, Ngai Wong‡, Hao Yu*
*School of Microelectronics, Southern University of Science and Technology, Shenzhen, China
†Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
‡Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong, China

***Abstract*—Tripping or falling is among the top threats in elderly healthcare, and the development of automatic fall detection systems are of considerable importance. With the fast development of the Internet of Things (IoT), camera vision-based solutions have drawn much attention in recent years. The traditional fall video analysis on the cloud has significant communication overhead. This work introduces a fast and lightweight video fall detection network based on a spatio-temporal joint-point model to overcome these hurdles. Instead of detecting falling motion by the traditional Convolutional Neural Networks (CNNs), we propose a Long Short-Term Memory (LSTM) model based on *time-series joint-point features*, extracted from a *pose extractor* and then filtered from a *geometric joint-point filter*. Experiments are conducted to verify the proposed framework, which shows a high sensitivity of 98.46% on Multiple Cameras Fall Dataset and 100% on UR Fall Dataset. Furthermore, our model can achieve pose estimation tasks simultaneously, attaining 73.3 mAP in the COCO keypoint challenge dataset, which outperforms the OpenPose work by 8%.**

***Index Terms*—fall detection, pose estimation, spatio-temporal model, joint-point features**

I. Introduction

Falling has become a severe growing problem in the elderly group, which causes numerous injuries and deaths. According to the World Health Organization [1], $646,000$ elderly persons died by falling, and around $28-35\%$ of the people over age 65 fall every year. Therefore, real-time fall detection systems are of an urgent need for the community. In the literature, the fall detection approaches can be mainly divided into two categories [2]: **1)** Sensor-based approaches, which use accelerometers to measure the vertical acceleration, velocities, and angles [3], [4]. **2)** Vision-based approaches, which capture the information directly from the video scenes. In fact, the sensor-based approaches require the older people to carry the sensor devices all the time, which has been widely perceived as inefficient and inaccurate [5]. Recently, driven by the advancement of deep learning networks, vision-based approaches using CNNs and LSTM have been successfully applied to the fall detection task [6]. Nevertheless, the CNN-based methods focus on each frame while lacking the capability of capturing temporal information in the video stream [7]. The LSTM-based schemes perform the sequence-to-sequence modelings, which however, contain a considerable number of redundant parameters and lead to an expensive computation cost.

To address these issues, we propose a novel video fall detection network by a spatio-temporal joint-point model. It detects the falling behavior using a tensorized LSTM based on the time-series joint-point features, which are extracted from a pose detector and then filtered from a geometric joint-point filter. Different from the traditional bottom-up approaches (e.g., OpenPose [8]) to firstly detect body points and then associate them with persons (resulting in a series of wrong connections and complicated post-processing), the proposed framework follows a top-down scheme firstly to locate persons and then determine body points of each person. The main contributions are fourfold:

- The proposed framework combines all *spatial, temporal, and geometric information*, for the first time, towards a video fall detection.
- We devise a *geometric joint-point filter* to filtrate the joint-point features from the former *pose extractor*, which delivers high accuracy.
- We make novel use of a *spatio-temporal joint-point model* based on the *time-series joint-point features* aggregated from the geometric joint-point filter.
- We develop a *tensorized compression* of the proposed spatio-temporal joint-point model, resulting in a fast and lightweight realization on an *ARM-core based IoT board*.

Benchmarking is made on 2 fall detection datasets and 1 pose keypoint dataset. Experimental results show that the proposed video fall detection framework achieves a high sensitivity of 98.46% on Multiple Cameras Fall [9] and 100% on UR Fall [10]. Moreover, our model attains 73.3 mAP in the COCO keypoint challenge [11] on pose estimation task, which outperforms the OpenPose [8] by 8%.

In the following, Section II reviews some related works. Section III presents the proposed video fall detection framework. Section IV proposes the tensorized compression of the spatio-temporal joint model and introduces the implementation flow on an ARM-core based IoT board. Section V reports the experimental results on several datasets. And Section VI draws the conclusion.

II. Related Work

A. Vision-based Fall Detection

The vision-based fall detection approaches are mainly based on the static or dynamic features of human bodies in frames. The features of aspect ratio, joint-point locations, or human body axes are further processed to classify falls. Vaidehi et al. [12] apply the foreground image of the human body and determine the bounding box then judge the falling behavior according to the aspect ratio and the tilt angle. The method in [13] analyzes the ellipse around the human body, the

978-3-9819263-5-4/DATE21/©2021 EDAA

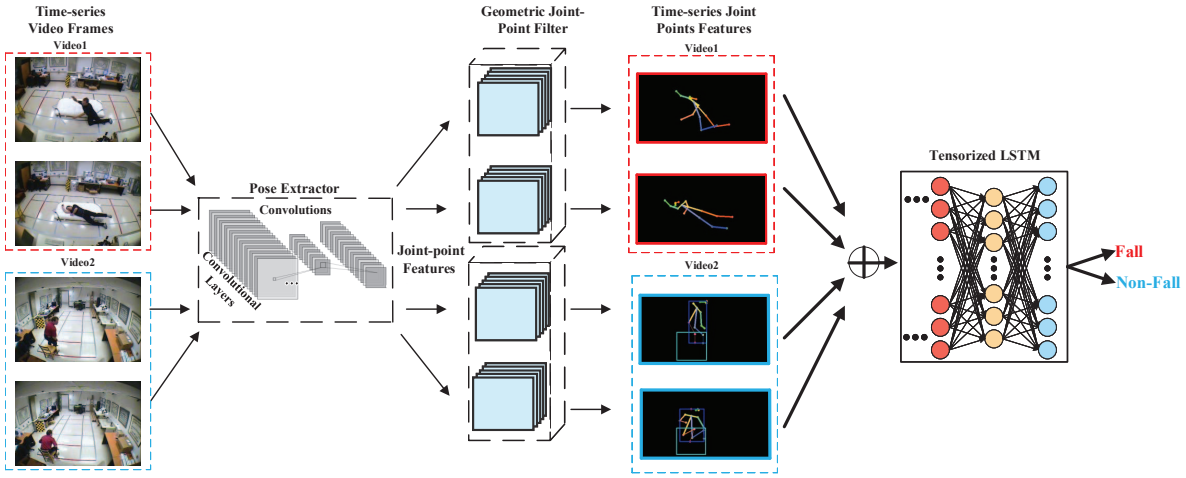

Fig. 1. The proposed video fall detection network by spatio-temporal joint-point model.

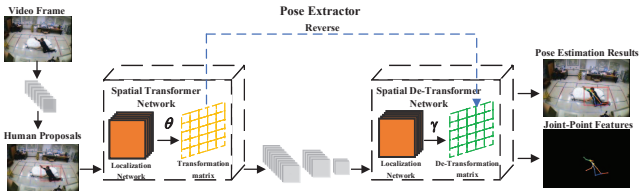

Fig. 2. Extracting the joint-point features from video frames.

projected histogram of the contour after segmentation, and the instantaneous change of the head position. Then these features are classified by a multi-layer perceptron (MLP). In [14], the characteristics of the wide-angle camera are used to calculate the projected gravity vector, and the falling behavior is judged according to the tilt angle. Vision-based systems have some advantages: 1) Multi-person detections being performed simultaneously. 2) No effect on the people being tested. 3) Low system cost. Nevertheless, all the aforementioned methods focus on handling the spatial features for fall detection in a single frame without temporal or geometric information, while suffering from unsatisfactory accuracy and speed. In the proposed framework, we combine all spatial, temporal, and geometric information for a fast and lightweight video fall detection network.

B. Pose Estimation

In the last decade, action recognition approaches are dominated by deep learning neural networks. For most *pose-based* action recognition frameworks, human pose features are extracted and categorized as joint-point locations for 2D pose and skeleton data for 3D pose. The pose-based CNN (P-CNN) [15] first extracts the patches of body parts like the left hand in RGB and optical flow for each frame. Then both information are fed into the two-stream networks. Since this method is based on the extracting body parts using CNN multiple times, it causes a relatively high computational cost. For 3D pose estimation, with depth sensors such as Microsoft Kinect, it is convenient to capture 3D skeletal data. Some methods [16], [17] construct

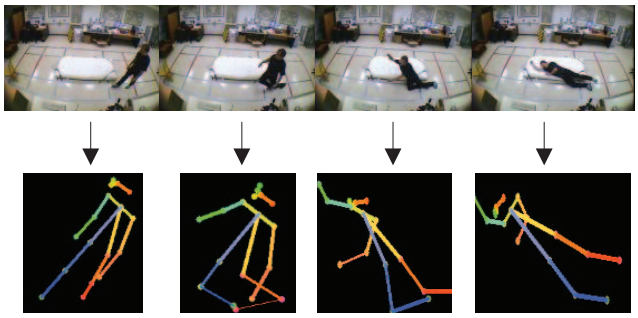

Fig. 3. Continuous frame samples and corresponding joint-point features

the spatio-temporal LSTM network to learn the association between skeleton sequences. However, the use of conventional LSTM networks incurs huge computational complexity and is difficult to be implemented on terminal devices. In the proposed framework, we apply a tensorized strategy during the network training, which results in the high compression ratio and fast inference speed.

III. Proposed Framework

In this section, we elaborate on the proposed video fall detection framework shown in Fig. 1. First, a pose extractor is applied to detect persons from each frame and extract joint-point features. Subsequently, the joint-point features are further filtered by a joint-point filter with several geometric criteria (as described in Section III-B). Finally, the selected joint-point features are fed into the spatio-temporal joint-point model for video fall detection.

A. Pose Extractor

The workflow of the proposed pose extractor is a "two-step" process, as shown in Fig. 2. First, we adopt several convolutional layers to generate the region proposals for persons. In order to get high-quality person proposals, we apply a Spatial Transformer Network (STN) for human detection. It should be noted that the traditional object detection methods make

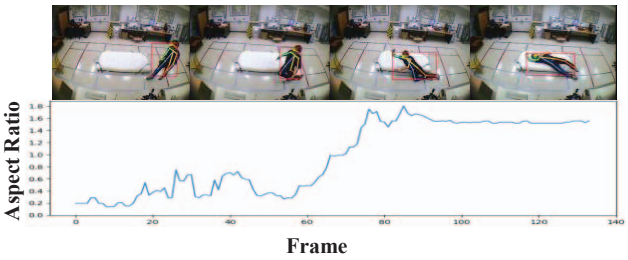

Fig. 4. Aspect ratio v.s. frame as a person falls rightward.

networks adapt to the distortion of the objects, and thus generating poor proposals. Unlike the traditional object detection methods whose receptive fields are fixed and local, the proposed STN can generate an appropriate spatial transformation such as cropping and rotations for each input image. In this way, STN is optimized to select normalized and expected regions to simplify the process of subsequent layers. The spatial transformation in STN can be expressed as:

$$\begin{pmatrix} x_i^b \\ y_i^b \end{pmatrix} = [\boldsymbol{\theta_1}\ \boldsymbol{\theta_2}\ \boldsymbol{\theta_3}] \begin{pmatrix} x_i^a \\ y_i^a \\ 1 \end{pmatrix}, \tag{1}$$

where $\boldsymbol{\theta_1}, \boldsymbol{\theta_2}$ and $\boldsymbol{\theta_3}$ are vectors in $\mathbb{R}^2$, forming the transformation matrix together. These parameters are generated by the localization network as shown in Fig. 2. $\{x_i^b, y_i^b\}$ are the original coordinates and $\{x_i^a, y_i^a\}$ are the corresponding coordinates after the transformation. Applying the STN has the advantages of providing aligned spatial feature graphs with low noise, as well as capturing the robust spatial structure of body parts (e.g., locations and angles). Subsequently, the revised high-quality proposals are processed to generate a group of heatmaps to indicate the human joint-point locations. Specifically, a Spatial De-Transformer Network (SDTN) is applied to remap the estimated pose heatmaps to the original coordinates and produce the joint-point information. For every joint of the body, the de-transformation matrix can be represented as: $[\boldsymbol{\gamma_1}\ \boldsymbol{\gamma_2}\ \boldsymbol{\gamma_3}]$, where $\boldsymbol{\gamma_1}, \boldsymbol{\gamma_2}$ and $\boldsymbol{\gamma_3}$ are the are vectors in $\mathbb{R}^2$, which are also determined by the localization network. The de-transformation process can be shown below:

$$\begin{pmatrix} x_i^a \\ y_i^a \end{pmatrix} = [\boldsymbol{\gamma_1}\ \boldsymbol{\gamma_2}\ \boldsymbol{\gamma_3}] \begin{pmatrix} x_i^b \\ y_i^b \\ 1 \end{pmatrix} \tag{2}$$

Assuming $\boldsymbol{\mathcal{I}_t} \in \mathbb{R}^{l_1 \times l_2}$ are the video frame data and $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_d}$ are the time-series joint-point features, where subscript t represents the time sequence, l and d indicate the dimension sizes. The process of pose extractor can be represented as:

$$\boldsymbol{\mathcal{P}_t} = extract(\boldsymbol{\mathcal{I}_t}), \tag{3}$$

where the $extract$ represents the extraction method in the pose extractor. Each coordinate of the joint-point features can be further divided into $\boldsymbol{\mathcal{P}_x}$ and $\boldsymbol{\mathcal{P}_y}$ for x and y axes. The continuous frame samples of joint-point features from the pose extractor are shown in Fig. 3.

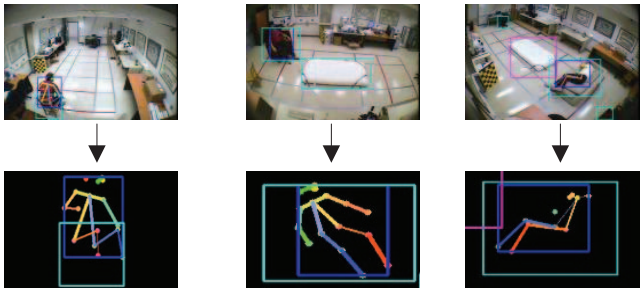

Fig. 5. Analysis of overlap proportion between persons and chairs/sofas.

B. Geometric Joint-point Filter

When a person falls, the falling motion has four directions: rightward, leftward, upward, downward. After obtaining the joint-point features from the pose extractor, we can further filtrate the joint points based on several criteria in different directions as follows.

The first rule is based on aspect ratio. In the pose extractor, the joint points are detected and encapsulated in a rectangle. When a person falls rightward or leftward, the aspect ratio of width and height of the rectangle changes dramatically. Therefore, we can consider the aspect ratio into falling motion judgment, which is defined as $width/height$ of the detected rectangle. The aspect ratio can be formulated as follows:

$$r_1 = \frac{\mathcal{P}_{x,max} - \mathcal{P}_{x,min}}{\mathcal{P}_{y,max} - \mathcal{P}_{y,min}}, \tag{4}$$

where $\mathcal{P}_{x,max}$ and $\mathcal{P}_{y,max}$ are the maximum values of joint points in the x and y axes, while the $\mathcal{P}_{x,min}$ and $\mathcal{P}_{y,min}$ are the minimum values respectively. As shown in Fig. 4, a person falls rightward, which makes the curve of aspect ratio rises significantly. Therefore, we can detect falls by the rule of aspect ratio r_1 higher than the threshold value.

The second rule is based on lower and upper body ratio. When a person falls upward, the upper body goes away from the cameras and its projection size becomes smaller. Hence, we can detect the falling behavior by the rule of lower and upper body ratio expressed as follows:

$$\begin{aligned} r_2 &= \tfrac{l_{down}}{l_{up}}, \\ l_{up} &= \sqrt{(\mathcal{P}_{x,head} - \mathcal{P}_{x,center})^2 + (\mathcal{P}_{y,head} - \mathcal{P}_{y,center})^2}, \\ l_{down} &= \sqrt{(\mathcal{P}_{x,center} - \mathcal{P}_{x,foot})^2 + (\mathcal{P}_{y,center} - \mathcal{P}_{y,foot})^2}, \end{aligned} \tag{5}$$

where $\mathcal{P}_{x,head}, \mathcal{P}_{x,center}$ and $\mathcal{P}_{x,foot}$ are the coordinates in the x axis, which are the same in the y axis. l_{up} and l_{down} are the lengths of the upper body and lower body, respectively. Similarly, the falling behavior is detected when the lower and upper body ratio r_2 is higher than the threshold value.

The third rule is based on a coordinate relationship. When a person falls downward, the y-coordinate of head becomes greater than the feet.

The detected joint points of sitting in chairs or sofas are similar to falling according to the first and second rules. To distinguish falling from sitting, we introduce a rule which is based on the overlap of bounding boxes of persons and

chairs/sofas. As shown in Fig. 5, we can define the overlap ratio $r3$ as follows:

$$r_3 = \frac{S_{overlap}}{S_{person}}, \tag{6}$$

where $S_{overlap}$ means the overlap area between bounding box of persons and sofa/chair, S_{person} denotes the area of person bounding box. We can judge the sitting behavior to distinguish falling when the $r3$ is higher than the threshold value. Consequently, we can further filtrate the joint-point features with the proposed four rules and provide the distilled information for the following spatio-temporal joint-point model, leading to high efficiency and speed.

C. Spatio-temporal Joint-point Model

After obtaining the joint-point features by pose extractor and geometric joint-point filter, we construct the spatio-temporal joint-point model to process the sequence-to-sequence information for fall detection in the video stream.

Instead of directly using the raw frame data $\boldsymbol{\mathcal{I}_t} \in \mathbb{R}^{l_1 \times l_2 \times l_3}$, we input the time-series joint-point features $\boldsymbol{\mathcal{P}_t} \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_d}$ to the proposed spatio-temporal joint-point model. Each LSTM cell keeps track of an internal state representing its memory and updates its state over time based on the current input and past states. Therefore, all the video information can be obtained from the beginning to the current frame [18]. This process can be written as the following equations:

$$\begin{gathered}
\boldsymbol{K_t} = \sigma_s(\boldsymbol{W_k}\boldsymbol{\mathcal{P}_t} + \boldsymbol{U_k}\boldsymbol{H_{t-1}} + \boldsymbol{B_k}), \\
\boldsymbol{F_t} = \sigma_s(\boldsymbol{W_f}\boldsymbol{\mathcal{P}_t} + \boldsymbol{U_f}\boldsymbol{H_{t-1}} + \boldsymbol{B_f}), \\
\boldsymbol{O_t} = \sigma_s(\boldsymbol{W_o}\boldsymbol{\mathcal{P}_t} + \boldsymbol{U_o}\boldsymbol{H_{t-1}} + \boldsymbol{B_o}), \\
\boldsymbol{\tilde{C}_t} = \sigma_h(\boldsymbol{W_c}\boldsymbol{\mathcal{P}_t} + \boldsymbol{U_c}\boldsymbol{H_{t-1}} + \boldsymbol{B_c}), \\
\boldsymbol{C_t} = \boldsymbol{K_t} \odot \boldsymbol{C_{t-1}} + \boldsymbol{F_t} \odot \boldsymbol{\tilde{C}_t}, \\
\boldsymbol{H_t} = \boldsymbol{O_t} \odot \sigma_h(\boldsymbol{C_t}),
\end{gathered} \tag{7}$$

where $\odot$ denotes the element-wise product, σ_s represents the sigmoid function and σ_h represents the hyperbolic tangent function. $\boldsymbol{H_t}$ and $\boldsymbol{H_{t-1}}$ are the current and previous hidden states, while $\boldsymbol{C_t}$ and $\boldsymbol{C_{t-1}}$ are the current and previous update factors. The input joint-point features $\boldsymbol{\mathcal{P}_t}$ are multiplied by weight matrices $\boldsymbol{W_*}$ and $\boldsymbol{U_*}$, updating the factor $\boldsymbol{\tilde{C}_t}$ and three sigmoid gates $\boldsymbol{K_t}, \boldsymbol{F_t}$ and $\boldsymbol{O_t}$.

IV. Tensorized Compression On Edge device

The conventional LSTM has a massive number of parameters and high dimensional inputs. To release the computational complexity of the proposed fall detection framework, we present a tensorized compression strategy in LSTM (named LSTM^{T}) as follows. Given a d-dimension tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{l_1 \times l_2 \times \ldots \times l_d}$, it can be factorized in the form of:

$$\boldsymbol{\mathcal{W}}(h_1, h_2, ..., h_d) = \boldsymbol{\mathcal{G}}_1(h_1)\boldsymbol{\mathcal{G}}_2(h_2)\ldots\boldsymbol{\mathcal{G}}_d(h_d), \tag{8}$$

where $\boldsymbol{\mathcal{G}}_k(i_k) \in \mathbb{R}^{\alpha_{k-1} \times \alpha_k}$ is a 2-dimensional slice of 3-dimensional tensor $\boldsymbol{\mathcal{G}}_k$. For every integer l_k, it can be factorized as $l_k = m_k n_k$. Then the double-index trick [19] can be used to rewrite Eq. (8) as:

$$\begin{gathered}
\boldsymbol{\mathcal{W}}\left((i_1, j_1), (i_2, j_2), \ldots, (i_d, j_d)\right) = \boldsymbol{\mathcal{G}}_1^*(i_1, j_1)\,\boldsymbol{\mathcal{G}}_2^*(i_2, j_2)\ldots \\
\boldsymbol{\mathcal{G}}_d^*(i_d, j_d).
\end{gathered} \tag{9}$$

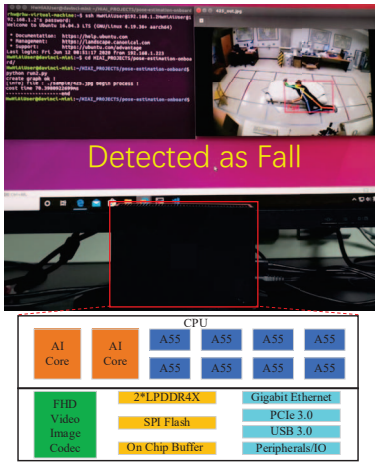

Fig. 6. Implementation and Architecture of ARM-core based IoT board.

In general, the computational bottleneck in LSTM is the fully-connected operation in its four gating units, which can be expressed as:

$$\begin{gathered}
\boldsymbol{q}(j) = \sum_{i=1}^{M} \boldsymbol{W}(i,j) \cdot \boldsymbol{p}(i) + \boldsymbol{b}(j), \\
\forall j \in [1, N], \boldsymbol{p} \in \mathbb{R}^M, \boldsymbol{q} \in \mathbb{R}^N,
\end{gathered} \tag{10}$$

where $\boldsymbol{p} \in \mathbb{R}^M$ is the input vector, $\boldsymbol{b} \in \mathbb{R}^N$ is the bias vector, $\boldsymbol{q} \in \mathbb{R}^N$ is the output vector and $\boldsymbol{W} \in \mathbb{R}^{M \times N}$ is the weight matrix. We can approximate $\boldsymbol{W} \cdot \boldsymbol{p}$ with much fewer parameters, making $M = \prod_{k=1}^{d} m_k$ and $N = \prod_{k=1}^{d} n_k$. Similarly, we can reshape the input vector and output vector as $\boldsymbol{\mathcal{P}} \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_d}$ and $\boldsymbol{\mathcal{Q}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$. Combining with the aforementioned tensor decomposition in Eq. (9) , we can rewrite the Eq. (10) as

$$\begin{gathered}
\boldsymbol{\mathcal{Q}}(j_1, j_2, \ldots, j_d) = \sum_{i_1=1}^{m1} \sum_{i_2=1}^{m2} \cdots \sum_{i_d=1}^{md} \boldsymbol{\mathcal{G}}_1^*(i_1, j_1) \\
\boldsymbol{\mathcal{G}}_2^*(i_2, j_2)\ldots\boldsymbol{\mathcal{G}}_d^*(i_d, j_d) \cdot \boldsymbol{\mathcal{P}}(i_1, i_2, \ldots, i_d) + \boldsymbol{\mathcal{B}}(j_1, j_2, \ldots, j_d)
\end{gathered} \tag{11}$$

After the tensorized compression, the computational complexity can be expressed as $O\left(dr^2 n_m\right)$ instead of $O\left(n^d\right)$ for original FC layer, where r is the maximum rank of core tensors $\boldsymbol{\mathcal{G}_k}$ and n_m is the maximum mode size $m_k n_k$ of tensor $\boldsymbol{\mathcal{W}}$.

The tensorized compression of the proposed framework delivers significant reduction in storage size and computational complexity, leading to a hardware-friendly implementation on edge device. To realize the proposed fall detection framework on terminal devices, we choose an ARM-core based IoT board, with the efficient and programmable AI chip. As shown in Fig. 6, the board includes a digital vision pre-processing module, an ARM CPU, two AI cores, and an on-chip buffer with a capacity of 8MB.

The most expensive computation parts in the proposed framework are the extraction of joint-point features in the pose extractor and the fall detection in the spatio-temporal joint-point model. For the fall detection part, we have performed the tensorized compression algorithm to reduce the complexity of LSTM model. Moreover, for the feature extraction part, we

TABLE I
PERFORMANCE COMPARISON BETWEEN THE PROPOSED FRAMEWORK AND THE OTHER APPROACHES ON MULTICAM DATASET.

Method	Sensitivity	Specificity	Accuracy
SVM [22]	93.70	92.00	-
skeleton feature [23]	90.90	**93.75**	92.59
PCANet [24]	89.20	90.30	-
Multi-model [25]	96.28	-	96.27
CNN+LSTM [26]	91.60	93.50	-
The proposed framework	**98.46**	91.82	**96.28**

TABLE II
PERFORMANCE COMPARISON BETWEEN THE PROPOSED FRAMEWORK AND THE OTHER APPROACHES ON URFD DATASET.

Method	Sensitivity	Specificity	Accuracy
Hourglass Convolution [27]	100.0	93.00	96.20
MEWMA [28]	100.0	92.00	95.00
VGG-16 CNN [29]	100.0	92.00	95.00
SVM [30]	98.00	89.40	93.96
Shi-Tomasi algorithm [31]	96.66	95.00	93.54
The proposed framework	**100.0**	**97.44**	**99.00**

apply the AI acceleration core in the IoT board to accelerate the network inference. We implement the proposed framework in ARM-core based IoT board in Fig. 6.

V. EXPERIMENT

The experimental results of the proposed framework are summarized in this section. The pose extractor is based on Alphapose [20] and ResNet-101 [21] is chosen as our localization network in STN.

A. Experimental Settings

Our experiment is based on TensorFlow in Keras with a single NVIDIA GTX-1080Ti. To verify the capability of the proposed framework for fall detection, we evaluate the framework on 2 datasets. The first one is the Multiple cameras fall dataset (Multicam) [9], which consists of 24 scenarios recorded with 8 different video cameras installed at different locations. In each scenario, 9 different activities are performed including falling, walking, sitting, and lying on the ground. The second dataset is UR Fall Detection Dataset (URFD) [10], which contains 30 falls and 40 ADL (activities of daily living) recorded with Microsoft Kinect cameras. All videos are segmented in frames at the FPS of 8 and these frames are divided into falls and non-falls cases. Furthermore, we validate the pose estimation performance in the COCO keypoint challenge dataset [11], which contains over $150,000$ instances for training and $80,000$ instances for testing.

B. Evaluation of Accuracy

Fall detection can be regarded as a binary classification problem, which requires the classifier to classify whether there is a fall or not. To evaluate the proposed framework's performance, we use sensitivity, specificity and accuracy as metrics, which are commonly used in the fall detection field and defined as follows:

$$\text{Sensitivity} = \frac{TP}{TP+FN}, \tag{12}$$

$$\text{Specificity} = \frac{TN}{TN+FP}, \tag{13}$$

TABLE III
PRECISION COMPARISON OF POSE ESTIMATION ON COCO DATASET.

Method	mAP
OpenPose [8]	65.3
Mask R-CNN [32]	63.1
Personlab [33]	68.7
Multiposenet [34]	70.5
Our pose estimation	**73.3**

TABLE IV
SPEED EVALUATION ON THE PROPOSED FRAMEWORK

Method	FPS
Original joint points + Original LSTM	6.22
Joint-point features + LSTM^{T}	**17.85**

TABLE V
ABLATION STUDY OF PROPOSED FRAMEWORK ON MULTICAM.

Method	Sensitivity	Specificity	Accuracy
Original LSTM	88.37	40.00	79.24
Original joint points + LSTM	91.71	87.95	90.53
Joint-point features + LSTM	96.67	86.67	93.33
Joint-point features + LSTM^{T}	**98.46**	**91.82**	**96.28**

TABLE VI
SPEED COMPARISON BETWEEN THE IMPLEMENTATIONS ON CPU AND ARM-CORE BASED IOT BOARD ON MULTICAM DATASET.

Device	Method	FPS
CPU	Original joint points + LSTM	6
ARM CPU+AI core	Joint-point features + LSTM^{T}	**48**

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

where TP (True Positive) and TN (True Negative) are the numbers of correct predictions of falls and non-falls, respectively. FN (False Negative) and FP (False positive) are the numbers of actual falls missed by the system and false predictions of non-falls, respectively. Moreover, to evaluate the pose estimation function, we adopt the metric of mean Average Precision (mAP) of all body parts.

Specifically, we compare the proposed framework with 5 state-of-the-art vision-based approaches on the Multicam dataset in Table I and 5 existing methods (including one sensor-based method) on the URFD dataset in Table II. As seen in Table I, the proposed framework achieves the best sensitivity and accuracy as 98.46% and 96.28%, and the specificity is close (91.82%) to the best one. In Table II, it can be seen that our model significantly outperforms all the other approaches, which demonstrates the superiority of the proposed framework. We also validate the mAP on pose estimation with 4 state-of-art approaches in Table III. It can be observed that the proposed framework achieves 73.3 mAP on the COCO dataset, which outperforms all other four approaches.

We further test the inference speed in Table IV, which shows that the proposed framework achieves $2.87\times$ speedup compared with the original LSTM with joint points. Furthermore, to verify every component in the proposed framework, we report the ablation study of the proposed framework on Multicam in Table V. From empirical experiments, we set the thresholds of $r1$ as 1.0, $r2$ as 1.85 and $r3$ as 0.6. We can observe that the proposed framework with pose extractor, joint-point filter

with geometric criteria, and tensorized compression is more accurate, robust and efficient.

After the evaluation on GPU, we further report the experiment results of the proposed framework on an ARM-core based IoT board. The test verification platform is shown in Fig. 6, which consists of a computer monitor and an ARM-core based IoT board. Using the Multicam dataset, we can obtain visual results of fall detection from the monitor, as seen in Fig. 6. Accelerated by the AI acceleration core and the proposed tensorized compression, the inference speed on the board is measured to be 48 FPS, which is six times faster than a high-end CPU, as shown in Table VI.

VI. Conclusion

This paper has proposed an accurate and robust video fall detection system based on a spatio-temporal joint-point model, which performs both pose estimation and fall detection. We extract and filtrate the time-series joint-point features to capture the spatial and temporal dynamics, which are subsequently fed to the tensorized spatio-temporal joint-point model to predict falls. Experiments using two fall datasets have shown the remarkable efficacy of the proposed framework. Especially on the URFD dataset, 100% sensitivity and 99% accuracy have been recorded, which is the best result ever reported.

References

[1] W. H. Organization, W. H. O. Ageing, and L. C. Unit, *WHO global report on falls prevention in older age*. World Health Organization, 2008.

[2] L. Ren and Y. Peng, "Research of fall detection and fall prevention technologies: A systematic review," *IEEE Access*, vol. 7, pp. 77 702–77 722, 2019.

[3] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.

[4] Q. Li, J. A. Stankovic, M. A. Hanson, A. T. Barth, J. Lach, and G. Zhou, "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," in *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*. IEEE, 2009, pp. 138–143.

[5] R. Igual, C. Medrano, and I. Plaza, "Challenges, issues and trends in fall detection systems," *Biomedical engineering online*, vol. 12, no. 1, p. 66, 2013.

[6] N. Lu, Y. Wu, L. Feng, and J. Song, "Deep learning for fall detection: Three-dimensional cnn combined with lstm on video kinematic data," *IEEE journal of biomedical and health informatics*, vol. 23, no. 1, pp. 314–323, 2018.

[7] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7083–7093.

[8] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: realtime multi-person 2d pose estimation using part affinity fields," *arXiv preprint arXiv:1812.08008*, 2018.

[9] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Multiple cameras fall dataset," *DIRO-Université de Montréal, Tech. Rep*, vol. 1350, 2010.

[10] 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.

[11] 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 *European conference on computer vision*. Springer, 2014, pp. 740–755.

[12] V. Vaidehi, K. Ganapathy, K. Mohan, A. Aldrin, and K. Nirmal, "Video based automatic fall detection in indoor environment," in *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*. IEEE, 2011, pp. 1016–1020.

[13] H. Foroughi, B. S. Aski, and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," in *2008 11th international conference on computer and information technology*. IEEE, 2008, pp. 219–224.

[14] M. Bosch-Jorge, A.-J. Sánchez-Salmerón, Á. Valera, and C. Ricolfe-Viala, "Fall detection based on the gravity vector using a wide-angle camera," *Expert systems with applications*, vol. 41, no. 17, pp. 7980–7986, 2014.

[15] G. Chéron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3218–3226.

[16] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[17] J. C. Nunez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Velez, "Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition," *Pattern Recognition*, vol. 76, pp. 80–94, 2018.

[18] Y. Cheng, G. Li, N. Wong, H.-B. Chen, and H. Yu, "Deepeye: A deeply tensor-compressed neural network for video comprehension on terminal devices," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 3, pp. 1–25, 2020.

[19] Y. Cheng, G. Huang, P. Zhen, B. Liu, H.-B. Chen, N. Wong, and H. Yu, "An anomaly comprehension neural network for surveillance videos on terminal devices," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1396–1401.

[20] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "Rmpe: Regional multi-person pose estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2334–2343.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[22] K. Wang, G. Cao, D. Meng, W. Chen, and W. Cao, "Automatic fall detection of human in video using combination of features," in *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2016, pp. 1228–1233.

[23] Y.-T. Chen, Y.-C. Lin, and W.-H. Fang, "A hybrid human fall detection scheme," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 3485–3488.

[24] S. Wang, L. Chen, Z. Zhou, X. Sun, and J. Dong, "Human fall detection in surveillance video based on pcanet," *Multimedia tools and applications*, vol. 75, no. 19, pp. 11 603–11 613, 2016.

[25] 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 *Machine Learning for Health Workshop*, 2020, pp. 39–51.

[26] Q. Feng, C. Gao, L. Wang, Y. Zhao, T. Song, and Q. Li, "Spatio-temporal fall event detection in complex scenes using attention guided lstm," *Pattern Recognition Letters*, vol. 130, pp. 242–249, 2020.

[27] X. Cai, S. Li, X. Liu, and G. Han, "Vision-based fall detection with multi-task hourglass convolutional auto-encoder," *IEEE Access*, vol. 8, pp. 44 493–44 502, 2020.

[28] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, "Vision-based fall detection system for improving safety of elderly people," *IEEE Instrumentation & Measurement Magazine*, vol. 20, no. 6, pp. 49–55, 2017.

[29] A. Núñez-Marcos, G. Azkune, and I. Arganda-Carreras, "Vision-based fall detection with convolutional neural networks," *Wireless communications and mobile computing*, vol. 2017, 2017.

[30] N. Zerrouki, F. Harrou, A. Houacine, and Y. Sun, "Fall detection using supervised machine learning algorithms: A comparative study," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 665–670.

[31] S. Bhandari, N. Babar, P. Gupta, N. Shah, and S. Pujari, "A novel approach for fall detection in home environment," in *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2017, pp. 1–5.

[32] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[33] G. Papandreou, T. Zhu, L.-C. Chen, S. Gidaris, J. Tompson, and K. Murphy, "Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 269–286.

[34] M. Kocabas, S. Karagoz, and E. Akbas, "Multiposenet: Fast multi-person pose estimation using pose residual network," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 417–433.