This ICCV Workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Falls Prediction Based on Body Keypoints and Seq2Seq Architecture

Minjie Hua
CloudMinds Technologies
michael.hua@cloudminds.com

Yibing Nan
CloudMinds Technologies
charlie.nan@cloudminds.com

Shiguo Lian
CloudMinds Technologies
sg_lian@163.com

Abstract

This paper presents a novel approach for predicting the falls of people in advance from monocular video. First, all persons in the observed frames are detected and tracked with the coordinates of their body keypoints being extracted meanwhile. A keypoints vectorization method is exploited to eliminate irrelevant information in the initial coordinate representation. Then, the observed keypoint sequence of each person is input to the pose prediction module adapted from sequence-to-sequence(seq2seq) architecture to predict the future keypoint sequence. Finally, the predicted pose is analyzed by the falls classifier to judge whether the person will fall down in the future. The pose prediction module and falls classifier are trained separately and tuned jointly using Le2i dataset, which contains 191 videos of various normal daily activities as well as falls performed by several actors. The contrast experiments with mainstream raw RGB-based models show the accuracy improvement of utilizing body keypoints in falls classification. Moreover, the precognition of falls is proved effective by comparisons between models that with and without the pose prediction module.

1. Introduction

Falls are a major cause of fatal injury especially for the elderly and create a serious obstruction for independent living [23]. Therefore, falls prediction is one of the most meaningful applications for elderly caring and home monitoring *etc*. The precognition of falls is the prerequisite for follow-up preventions and early warning, which will largely decrease the risks of falling accident. Although mainstream human action recognition (HAR) algorithms can be trained to recognize falls as one of the action classes, most of them utilize raw RGB information for classification and are not capable of predicting falls in advance.

However, the following characteristics make fall distinct from other actions: 1) Fall is highly relevant to the status of body keypoints, *i.e*., the body skeleton of a fallen person is significantly different. 2) Unlike smoking and handshaking, fall does not involve interactions with objects or other people. 3) Fall is an accident rather than a daily activity. So it's expected to 'foresee' its happening and alert emergency as soon as possible.

According to 1) and 2), our falls classifier gives prediction using human body keypoints instead of raw RGB information in the video frames. The essence is to decrease the dimensionality of features with valuable cues preserved. With regard to 3), we introduce a pose prediction module adapted from sequence-to-sequence (seq2seq) [30] to predict future keypoint sequence based on the observed keypoint sequence. By analyzing future pose, the falls classifier is able to give early prediction.

It is well known that mainstream keypoints detection models like OpenPose [4] and AlphaPose [7] represent each extracted keypoint by its coordinate in the image. However, coordinate representation involves the body's absolute position and scale, which contribute little to action classification. Consequently, we propose a keypoints vectorization method to transform coordinates to a feature vector, in which only the direction information remains.

Since the pose prediction module already encodes temporal features, the falls classifier can focus on the spatial cues in the predicted body pose. To train the falls classifier, we re-annotated Le2i dataset [5] to tag each frame a label. This operation largely increases the amount of training data, which makes the falls classifier converge better.

The main contributions of this paper are as follows:

- We proposed an end-to-end falls prediction model, which consists of a seq2seq-based pose prediction module and a falls classifier.
- We developed a keypoints vectorization method to extract salient features from coordinate representation.
- We evaluated our model by comparisons with mainstream HAR networks, and self-comparison between the pose prediction module enabled and disabled.

The rest of the paper is organized as follows: Section 2 reviews the related work on falls detection, action recognition and prediction. The proposed network is presented in Section 3. In Section 4, the dataset and experiments are described. Finally, Section 5 gives a conclusion.

2. Related Work

2.1. Falls Detection

Many early solutions for falls detection relied on the wearable devices equipped by the person to be monitored. Bourke *et al*. proposed a method in [3] to detect falls based on peak acceleration. Narayanan *et al*. developed a distributed falls management system, which is capable of real-time falls detection using a waist-mounted triaxial accelerometer [24]. Bianchi *et al*. enhanced previous falls detection system with an extra barometric pressure sensor in [1, 2] and found that the algorithm incorporating pressure information achieved higher accuracy.

Although wearable device based falls detection methods are computationally efficient and insensitive to the environment, they require users to wear corresponding sensors for data collection. This limitation causes huge inconvenience to the users, and also makes it expensive for widespread application. Moreover, these algorithms suffer from a high false-detection rate due to sudden acceleration change in daily activities such as squat and run *etc*.

Later, vision-based solutions were developed to bring more user-friendly experience. Ma *et al*. [21] proposed an approach that extracted curvature scale space (CSS) features of human silhouettes from a depth camera and represented each action by a bag of CSS words (BoCSS), which was then classified by the extreme learning machine (ELM) to identify falls. Stone *et al*. presented solutions for falls recognition using gait parameters in [28] and five handcrafted features in [29]. And both were based on the foreground extracted from the Kinect. Besides, Quero *et al*. adopted non-invasive thermal vision sensors in [25] to detect falls using thermal images.

To the best of our knowledge, there was a lack of monocular vision based algorithms specialized for falls detection. However, most of the HAR models would support falls recognition after being trained on the dataset including fall as one of the actions. For example, the UCF-101 dataset [27] contains 13320 videos divided into 101 action categories but without falls. By contrast, the HMDB-51 [17], another large-scale dataset containing 6849 videos, takes 'fall on the floor' as one of the 51 action classes.

2.2. Action Recognition and Prediction

As investigated in [14], convolutional neural networks (CNN) and temporal modeling are the two major variables for action recognition. Karpathy *et al*. [12] pioneered to introduce CNN in HAR problem by finetuning a general image recognition model pre-trained on ImageNet [6] using UCF-101 [27]. However, The inability of utilizing temporal information became a severe disadvantage of 2D CNN. This issue was resolved by the 3D CNN proposed in [32, 11]. By performing 3D convolutions, the 3D CNN was capable of extracting features from both spatial and temporal dimensions, thereby capturing the motion information in multiple adjacent frames. As for the temporal modeling methods like Two-Stream [26] and TSN [35], the main idea was to extract spatial features using 2D convolutions and encode temporal information by recurrent neural networks(RNN).

However, all the aforementioned models predict only one label for each video, even when multiple actions are existing in the meantime. As shown in Fig. 1, although two people are presenting different actions, the TSN model [35] trained on UCF-101 dataset [27] just predict 'swing' with the man's fall being ignored.

Figure 1. An example of multiple actions presented in the meantime. TSN model trained on UCF-101 dataset gives the prediction 'swing'. However, the man's fall is ignored because only one label will be predicted for a video.

Action recognition models must 'watch' the entire video to give prediction. But in some cases like incomplete data or the requirement of early alarm, predicting the future action based on partial clip is necessary. Early classification and motion prediction are two routes of action prediction towards different goals: Given t_{obs} observed frames $(f_1, f_2, ..., f_{t_{\text{obs}}})$, early classification tries to infer the label y in advance, while motion prediction tries to produce future motions in next t_{pred} frames $(f_{t_{\text{obs}}+1}, f_{t_{\text{obs}}+2}, ..., f_{t_{\text{obs}}+t_{\text{pred}}})$.

The work in [20] designed novel ranking losses to learn activity progressing in LSTMs for early classification. Kong *et al*. adopted an auto-encoder to reconstruct missing features from observed frames by learning from the complete action videos [16]. In [15], a mem-LSTM model was proposed to store several hard-to-predict samples and a variety of early observations in order to improve the prediction performance at early stage.

In recent years, motion prediction attracted much attention. Fragkiadaki *et al*. proposed the encoder-recurrent-decoder (ERD) model [8] that extended long short term memory (LSTM) [9] to jointly learn representations and their dynamics. Jain *et al*. proposed structural-RNN (S-RNN) based on spatiotemporal graphs (st-graphs) in [10] to capture the interactions between the human and the objects. Martinez *et al*. modified standard RNN models in sampling-based loss and residual architectures for better motion prediction [22]. Tang *et al*. proposed modified highway unit (MHU) to filter the still keypoints and adopted gram matrix loss in [31] for long-term motion prediction. For the same purpose, Li *et al*. proposed convolutional encoding module (CEM) in [18] to learn the correlations between each keypoint, which is hard for an RNN model.

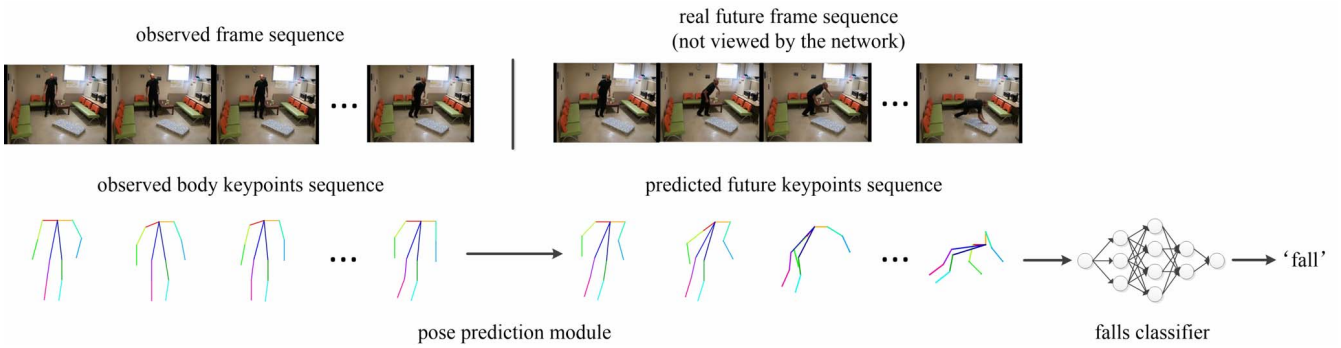

Figure 2. The major work flow of our model. A sequence of observed frames is input to the network. Then the human body keypoints are extracted from each frame to form a keypoints sequence, which is used to predict future keypoints sequence by the pose prediction module. At last, the predicted body pose is passed into a falls classifier to judge whether the person will fall down in the future.

3. Methodology

3.1. Overview of the Proposed Model

The problem to be solved in this paper is formulated as follow: Given t_{obs} observed frames $(f_1, f_2, ..., f_{t_{\text{obs}}})$, we try to predict whether the human(s) in the video will fall down in next t_{pred} frames (*i.e.*, from $f_{t_{\text{obs}}+1}$ to $f_{t_{\text{obs}}+t_{\text{pred}}}$).

The skeleton framework of our model is presented in Fig. 2. The input is a sequence of observed frames. We first adopted OpenPose [4] to extract keypoints coordinates of human(s) from each observed frame. The bounding boxes of detected persons were passed to DeepSort [36], a tracking algorithm, to cluster body keypoints belonging to the same person in different frames. As a result, the i-th person was corresponding to a sequence of observed keypoints $\mathbf{K}^i_{obs} = \left(\mathbf{k}^i_1, \mathbf{k}^i_2, ..., \mathbf{k}^i_{t_{\text{obs}}}\right)$, where $\mathbf{k}^i_j$ included the keypoints coordinates of the i-th person in frame j.

Based on the observation that fall is highly correlated to the relative position between body keypoints, we exploited a keypoints vectorization method to extract salient features from coordinate representation. The transformed sequence of the i-th person was denoted $\overline{\mathbf{K}^i_{obs}} = \left(\overline{\mathbf{k}^i_1}, \overline{\mathbf{k}^i_2}, ..., \overline{\mathbf{k}^i_{t_{\text{obs}}}}\right)$. Then, the pose prediction module adapted from seq2seq architecture [30] was used to predict body poses in next t_{pred} frames. Considering that applying excessive LSTM units made the network hard to converge, we encoded several consecutive keypoints vectors in one LSTM unit to shorten the lengths of both encoder and decoder LSTM layers. Moreover, shorter sequences also suppressed the mean-pose problem caused by long-term prediction [31, 18].

After that, $\overline{\mathbf{k}^i_{t_{\text{obs}}+t_{\text{pred}}}}$, the future keypoints vector of the i-th person at frame $f_{t_{\text{obs}}+t_{\text{pred}}}$, was predicted and used for classification. The falls classifier adopted fully connected network and was trained on re-annotated Le2i [5] dataset, in which each frame was labeled either 'fall' or 'no fall'. Combining the pose prediction module and falls classifier, our model was capable of predicting falls in advance.

3.2. Keypoints Vectorization

The keypoints coordinates extracted by OpenPose [4] cannot reflect the correlation between different keypoints, and suffered from the effects of body skeleton's absolute position and scale. With the motivation that the same pose should be represented by the same keypoints vector, we exploited the following keypoints vectorization method.

As we know, the 18 keypoints of MS COCO [19] are nose, neck, left and right shoulders, elbows, wrists, hips, knees, ankles, eyes and ears. Since we focused on the body keypoints, 5 face keypoints (*i.e.* nose, left and right eyes and ears) were ignored. For 13 concerned ones, we transformed their coordinates to vectors connecting with corresponding adjacent keypoints. As illustrated in Fig. 3, the left and right shoulders are connected to the neck, the left/right elbow is connected to left/right shoulder, and the left/right wrist is connected to left/right elbow. Similarly, the left and right hips are connected to the neck, the left/right knee is connected to left/right hip, and the left/right ankle is connected to left/right knee. As a result, 12 keypoints vectors were constructed from the coordinates information. Finally, we normalized all the vectors to unit length.

Formally, recall that the observed keypoints sequence of the i-th person was $\mathbf{K}^i_{obs} = \left(\mathbf{k}^i_1, \mathbf{k}^i_2, ..., \mathbf{k}^i_{t_{\text{obs}}}\right)$, where

$$\mathbf{k}^i_j = \left(x^i_{j,1}, y^i_{j,1}, x^i_{j,2}, y^i_{j,2}, ..., x^i_{j,18}, y^i_{j,18}\right). \quad (1)$$

$(x^i_{j,m}, y^i_{j,m})$ denoted the m-th keypoint coordinate of the i-th person in frame j. For the p-th connection pointing from the l-th keypoint to the r-th keypoint, the keypoints vector $\left(\overline{x^i_{j,p}}, \overline{y^i_{j,p}}\right)$ was calculated by:

$$\left(\overline{x^i_{j,p}}, \overline{y^i_{j,p}}\right) = \frac{\left(x^i_{j,r} - x^i_{j,l}, y^i_{j,r} - y^i_{j,l}\right)}{\sqrt{\left(x^i_{j,r} - x^i_{j,l}\right)^2 + \left(y^i_{j,r} - y^i_{j,l}\right)^2}}. \quad (2)$$

After 12 keypoints vectors were constructed, they were then concatenated to form $\overline{\mathbf{k}^i_j}$ as follows:

$$\overline{\mathbf{k}^i_j} = \left(\overline{x^i_{j,1}}, \overline{y^i_{j,1}}, \overline{x^i_{j,2}}, \overline{y^i_{j,2}}, ..., \overline{x^i_{j,12}}, \overline{y^i_{j,12}}\right). \quad (3)$$

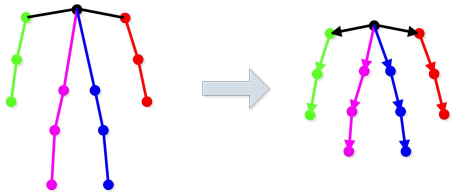

Figure 3. The illustration of the keypoints vectorization method. The arrows indicate 12 vectors constructed from the coordinates of 13 body keypoints. All the vectors are normalized to unit length with only direction information remains.

The transformation from $\mathbf{k}_j^i$ to $\overline{\mathbf{k}_j^i}$ eliminates the absolute position and scale of body skeleton, and preserves direction information between adjacent keypoints. It not only ensures that the same body pose has the same representation, but also extracts salient features for better falls classification.

3.3. Seq2Seq-based Pose Prediction Module

The authors in [30] proposed a seq2seq network, which was applied to machine translation at first and achieved excellent performance. Later, they introduced this architecture to conversational modeling in [34]. In analogy to mapping a sentence from one language to another in machine translation, the conversational model maps a query sentence to a response sentence. Generally, the seq2seq framework uses an LSTM [9] layer to encode the input sentence to a vector of a fixed dimensionality, and then another LSTM layer to decode the target sentence from the vector. This encoder-decoder architecture is widely used in sequence mapping problems such as machine translation [30], conversation modeling [34] and even video caption [33] because of its powerful capabilities.

Inspired by the great success of seq2seq network in the sequence mapping problems, we implemented a seq2seq-based pose prediction module to predict body poses in the future t_{pred} frames. Formally, recall that the observed keypoints vector sequence of the i-th person was $\overline{\mathbf{K}_{obs}^i} = \left(\overline{\mathbf{k}_1^i}, \overline{\mathbf{k}_2^i}, ..., \overline{\mathbf{k}_{t_{\text{obs}}}^i}\right)$. The pose prediction module was designed to generate future keypoints vector sequence $\overline{\mathbf{K}_{pred}^i} = \left(\overline{\mathbf{k}_{t_{\text{obs}}+1}^i}, \overline{\mathbf{k}_{t_{\text{obs}}+2}^i}, ..., \overline{\mathbf{k}_{t_{\text{obs}}+t_{\text{pred}}}^i}\right)$.

As illustrated in Fig. 4, the pose prediction module is composed of two LSTM [9] layers as the encoder and decoder respectively. The encoder analyzes the sequence of observed keypoints vectors with each LSTM unit parsing one keypoints vector. The hidden vector calculated by the previous unit is passed to the next one. In the decoder, the keypoints vector is generated one at a step. The first decoder LSTM unit accepts the last hidden vector from the encoder and outputs the first prediction. Latter units take the previous outcome as input and produce a new one. As all the LSTM units in the decoder complete predictions, the output keypoints sequence will be generated.

Recall the mechanism of LSTM layer: Assume that the input sequence is $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m)$, the t-th LSTM unit updates the states based on the states at $t-1$:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_i\right) \\
\mathbf{f}_t &= \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_f\right) \\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_o\right) \\
\widetilde{\mathbf{c}_t} &= \tanh\left(\mathbf{W}_c\left[\mathbf{h}_{t-1}, \mathbf{x}_t\right] + \mathbf{b}_c\right) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \widetilde{\mathbf{c}_t} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right),
\end{aligned} \tag{4}$$

where σ denotes the sigmoid function, $\tanh$ is the hyperbolic tangent function, $\odot$ denotes the element-wise multiplication, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ and $\mathbf{h}_t$ represent input gate, forget gate, output gate, cell state and hidden state of the t-th LSTM unit respectively. $\mathbf{W}$ and $\mathbf{b}$ are trainable weights.

We attempted to apply t_{obs} LSTM units to the encoder and t_{pred} LSTM units to the decoder (*i.e.*, each LSTM unit only deals with a single keypoints vector). However, we found that as the increment of t_{obs} and t_{pred}, the network was getting harder to converge. To mitigate the negative effect of long LSTM layer, we packed every n_p consecutive keypoints vectors in one for decreasing the length of keypoints sequence and the number of LSTM units. Formally, the packed sequence of $\overline{\mathbf{K}_{obs}^i} = \left(\overline{\mathbf{k}_1^i}, \overline{\mathbf{k}_2^i}, ..., \overline{\mathbf{k}_{t_{\text{obs}}}^i}\right)$ was $\widetilde{\mathbf{K}_{obs}^i} = \left(\widetilde{\mathbf{k}_1^i}, \widetilde{\mathbf{k}_2^i}, ..., \widetilde{\mathbf{k}_{t_{\text{obs}}/n_p}^i}\right)$, where

$$\widetilde{\mathbf{k}_j^i} = \overline{\mathbf{k}_{n_p(j-1)+1}^i} \oplus \overline{\mathbf{k}_{n_p(j-1)+2}^i} \oplus ... \oplus \overline{\mathbf{k}_{n_p j}^i}, \tag{5}$$

where $\oplus$ denotes the concatenation of two vectors. If there are no enough keypoints vectors for the last package (it will always happen when n_p is not divisible by t_{obs}), zero-paddings are filled to its tail for the dimensional equality. Correspondingly, since the vectors in output sequence are also packed, they need to be unpacked before classification to obtain the future pose at each frame.

Benefiting from the vector packing technique, the pose prediction module required less time to get convergent in the training phase, and the mean-pose problem raised in [31, 18] was also suppressed.

3.4. Falls Classifier

The falls classifier was trained to perform inference on the future keypoints vector at frame $f_{t_{\text{obs}}+t_{\text{pred}}}$. Considering that $\overline{\mathbf{k}_{t_{\text{obs}}+t_{\text{pred}}}^i}$ only contained 24 features, we simply adopted a traditional fully connected neural network for classification. The input layer was embedded with 24 neurons to fit the dimensionality of the input vector. And we set up five hidden layers containing 96, 192, 192, 96, 24 neurons respectively. The output layer with 2 neurons was used to give the final prediction: 'fall' or 'no fall'.

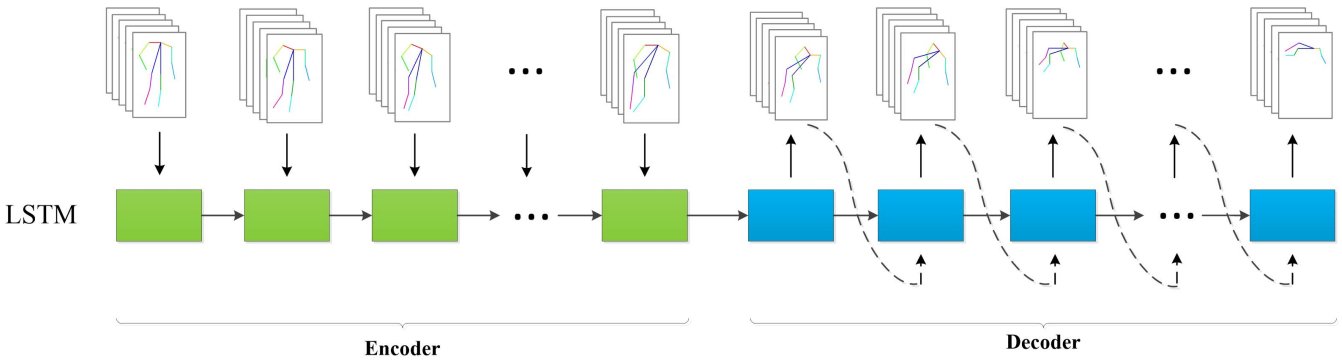

Figure 4. The architecture of our seq2seq-based pose prediction module, which is composed of an encoder (colored green), and a decoder (colored blue). Each LSTM unit in the encoder parses an observed keypoints vector (visualized in the figure to present more intuitive result) and produces a hidden vector. The first unit in the decoder accepts the last hidden vector from the encoder and generates the first prediction. Latter units receive the previous prediction and produce a new one. Note that the vector packing is reflected in the figure.

4. Experiments

4.1. Dataset Overview

We trained and evaluated our model on Le2i falls detection dataset [5], which consists of 191 videos captured under four different scenes: home, coffee room, office and lecture room. The frame rate is 25 frames per second and the resolution is 320×240 pixels. In each video, an actor performs various of normal activities and falls (fall might be absent in several videos). The official annotations provide the falling-start frame stamp and the falling-end frame stamp for each video. If there is no fall in a video, both frame stamps will be marked as 0 in its annotation file.

For the requirement of tagging a label for each frame, we first looked through all the videos and manually annotated an extra getting-up frame stamp for each one. If there is no fall appears, this value will be set to 0. And if the actor does not get up until the video ends, this value will be set to the last frame. Suppose that the frame stamps of falling start, falling end and getting up are denoted S_{fs}, S_{fe}, S_{gu} respectively, we attempted three automatic frame-annotation principles in our experiments:

1. Frames between S_{fs} and S_{fe} are labeled 'fall';
2. Frames between S_{fs} and S_{gu} are labeled 'fall';
3. Frames between S_{fe} and S_{gu} are labeled 'fall'.

And all the excluded frames are labeled 'no fall'. The first principle only regarded the falling proceeding as fall, which means 'no fall' will be annotated to a fallen person. Under this principle, the trained falls classifier could not recognize falls normally. The second one resulted in a high false positive rate because many 'precursors' of falling event will be predicted as 'fall' even when they are not leading to a real fall. The third principle achieved great balance by labeling 'fall' after the actor was already in the fallen state. So we finally adopted this principle for frame annotations.

4.2. Experiments Setup

We implemented our model on a workstation with double Nvidia 1080Ti GPUs. The seq2seq-based pose prediction module and the falls classifier were trained separately and tuned jointly.

To train the pose prediction module, we preprocessed all videos in the Le2i dataset. For each video, we utilized OpenPose to extract the actor's keypoints coordinates frame by frame, and transformed them to keypoints vectors with the method proposed in Section 3.2. However, due to the effect of illuminance or camera perspective *etc*., the actor's keypoints might be partially or completely missed. We dealt with the partial missing by setting vectors connecting undetected keypoint to $(0, 0)$ in the vectorization step. For the complete missing, the corresponding frames would be discarded. To ensure the coherence of keypoints sequence, consecutive 10 discarded frames would break the sequence into two segments. Then we filtered out the sequences containing less than 10 frames and finally obtained 139 sequences. The maximum, minimum and average frames of these sequences are 1773, 13 and 241.26 respectively.

During the training phase, the network loaded all the processed keypoints sequences and acquired training samples according to the configurations of t_{obs} and t_{pred}. For each sequence, all sub-sequences with length of $t_{\text{obs}} + t_{\text{pred}}$ frames were segmented and used as valid training samples. The former t_{obs} frames were input to the encoder of the pose prediction module, and the latter t_{pred} ones were regarded as ground truth. After the network completed inference, the mean square error (MSE) was applied to calculate the loss between ground truth and the predicted sequence. And the network was optimized using Adam [13] algorithm. In our experiments, the learning rate was set to 0.001.

To evaluate performance of the pose prediction module trained with different selections of t_{obs}, t_{pred} and n_p mentioned in Section 3.3, we exploited mean cosine similarity

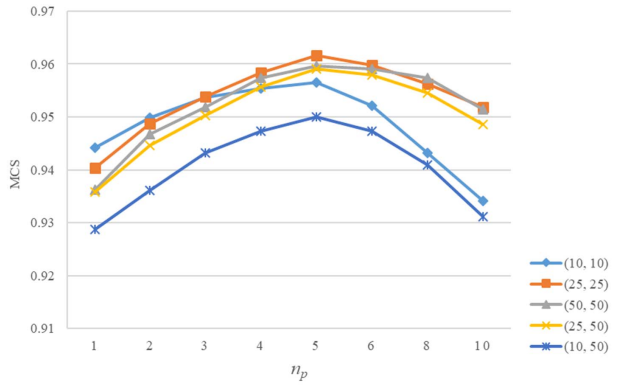

Figure 5. The MCS comparisons among several parameter configurations. We first selected five combinations of t_{obs} and t_{pred}, which is denoted by (t_{obs}, t_{pred}) in the figure, then trained and evaluated the pose prediction module with different n_p.

(MCS) as the metrics. Specifically, suppose that there are m test samples, the ground truth and the predicted sequence of the i-th sample are $\mathbf{g}^i$ and $\mathbf{p}^i$ respectively. Note that $\mathbf{g}^i$ and $\mathbf{p}^i$ both contain t_{pred} keypoints vectors ($\mathbf{p}^i$ has been unpacked). The j-th vector of $\mathbf{g}^i$ and $\mathbf{p}^i$ are denoted $\mathbf{g}^i_j$ and $\mathbf{p}^i_j$. Then the MCS was calculated by:

$$\text{MCS} = \frac{1}{m}\sum_{i=1}^{m} C^i, \tag{6}$$

where

$$C^i = \frac{1}{t_{\text{pred}}}\sum_{j=1}^{t_{\text{pred}}} \frac{\mathbf{g}^i_j \cdot \mathbf{p}^i_j}{\|\mathbf{g}^i_j\| \, \|\mathbf{p}^i_j\|}. \tag{7}$$

We tried several configurations of t_{obs}, t_{pred} and n_p to see the effect to the pose prediction module. It can be seen from Fig. 5 that observing 10 frames was insufficient for the network to give accurate predictions, especially for long predictions. Using 25 frames to predict 25 frames and using 50 frames to predict 50 frames both resulted in high MCS. Generally, the network was capable of predicting future 50 frames based on 25 observed frames with a tolerable MCS decrease. And $n_p = 5$ showed the best performance among all the candidate values. Hoping that the pose prediction module could predict further future poses with acceptable performance for earlier falls warning, we adopted $t_{\text{obs}} = 25$, $t_{\text{pred}} = 50$ and $n_p = 5$ in the deployment.

With respect to the training of falls classifier described in 3.4, we mapped each keypoints vector to the label of corresponding frame. Samples with less than 8 detected body keypoints were discarded because they contained too many null features. (In the inference phase, these samples would be prejudged as 'unknown' before the classification.) All the valid vector-label pairs were used to train the network and cross-entropy function was adopted to calculate the loss. Adam [13] was still selected as the optimizer.

4.3. Evaluations

Evaluations are designed concerning about the following questions: 1) Whether our falls classifier utilizing body keypoints information outperforms mainstream raw RGB based models. 2) What effects will the pose prediction module bring to the accuracy of falls classification.

4.3.1 Body Keypoints *vs*. Raw RGB

We conducted comparisons between our falls classifier and popular raw RGB-based HAR models including C3D [32], Two-Stream [26] and TSN [35] *etc*. However, considering that these models just predict one label to a complete video rather than a single frame, we converted the annotations from frame-label pairs to clip-label pairs.

The main idea was to segment 3-second (*i.e*. 75 frames) clips from the original video and labeled 'fall' to those including the whole falling proceeding and 'no fall' to those not involving falls at all. The choice of 3-second length is based on the statistic that the average duration of falling proceeding among all the videos is 1.26 seconds, which ensures the acquirement of positive samples. Formally, recall that S_{fs}, S_{fe} and S_{gu} are three frame stamps that mark the falling start, falling end and getting up respectively. Suppose a clip is segmented from the video's S_l-th frame to S_r-th frame ($S_r - S_l = 75$), its annotation is decided by:

$$\begin{array}{ll} S_l \le S_{fs} \text{ and } S_{fe} \le S_r \le S_{gu} & \text{labeled 'fall'} \\ S_r \le S_{fs} \text{ or } S_l \ge S_{gu} & \text{labeled 'no fall'} \end{array} \tag{8}$$

To avoid ambiguities, clips with partial falling proceeding were excluded. We finally obtained 9549 samples, which were randomly divided into a training set and a test set with the ratio of 7:3. We finetuned C3D (3 nets), Two-Stream and TSN (2 modalities) on the training set and evaluated them on the test set. For each clip, our falls classifier directly classified the keypoints vector from the frame S_r. The limitation of $S_r \le S_{gu}$ in Eq. (8) ensured the label consistency between each clip and its last frame.

The test set consists of 2865 samples, including 531 positive samples (fall) and 2334 negative samples (no fall). We adopted accuracy, precision, recall and F1-score to evaluate each model. As illustrated in Table 1, our keypoints-based classifier showed better performance than mainstream raw RGB-based models, which proved the salience of body keypoints features for falls classification problem.

Table 1. Comparisons between mainstream RGB-based models and our keypoints-based model on falls classification problem

Model	Acc.	Prec.	Rec.	F1
C3D [32]	89.4%	66.1%	87.9%	0.755
Two-Stream [26]	91.6%	71.6%	91.0%	0.801
TSN [35]	94.6%	79.8%	94.5%	0.866
Ours	**97.8%**	**90.8%**	**98.3%**	**0.944**

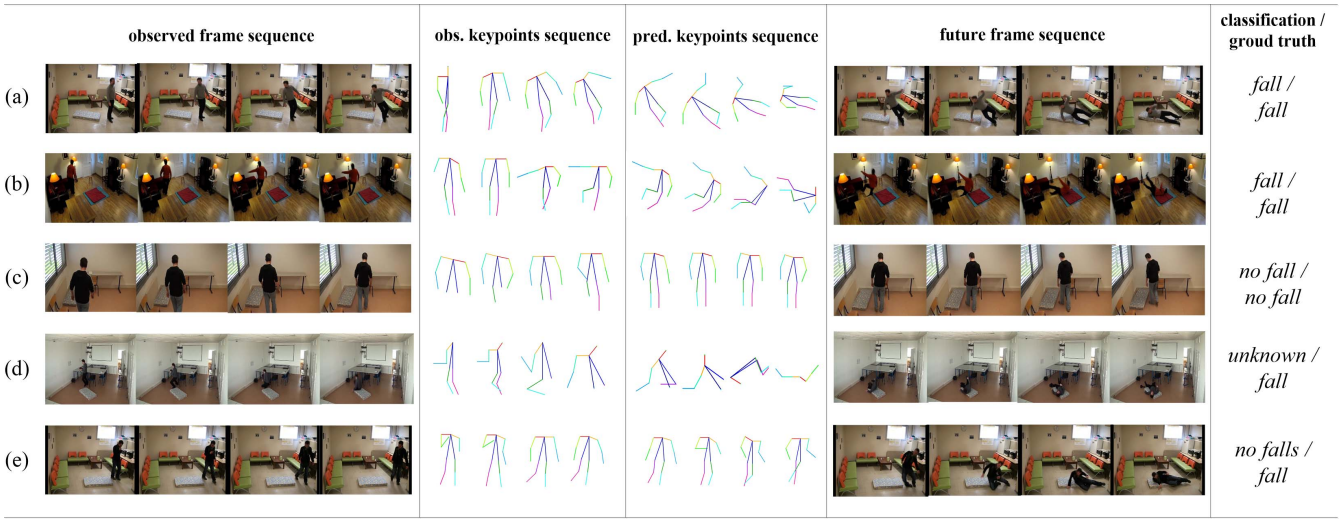

Figure 6. Some experimental results of our model. In case (a)-(c), the pose prediction module successfully predicts future keypoints sequence based on the observed frame sequence, and the falls classifier correctly infers the label. (d) and (e) are two failure cases. In (d), there are too many missing detections of the body keypoints due to the camera perspective. Consequently, the pose prediction module generates absurd keypoints prediction, which is insufficient for classification. In (e), the observed keypoints sequence does not includes any 'precursors' of falling event, thus leads to wrong prediction and classification.

Table 2. Comparison between $Model_{\text{cls}}$ and $Model_{\text{pred+cls}}$

Model	Acc.	Prec.	Rec.	F1
$Model_{\text{cls}}$	**99.2%**	**95.1%**	**98.8%**	**0.970**
$Model_{\text{pred+cls}}$	98.7%	92.6%	98.0%	0.952

4.3.2 Effects of the Pose Prediction Module

The pose prediction module was evaluated by self comparisons with this module enabled (denoted as $Model_{\text{pred+cls}}$) and disabled (denoted as $Model_{\text{cls}}$).

The first experiment was designed to compare the classification results between predicted keypoints and directly observed ones. Specifically, while predicting the label of the i-th frame in a video, $Model_{\text{pred+cls}}$ utilized frames from $i-74$ to $i-50$ (25 frames) to predict the keypoints vectors of frames from $i-49$ to i (50 frames). Then the predicted keypoints vector of frame i was input to falls classifier to produce a label. As for $Model_{\text{cls}}$, it was directly given the keypoints vector of the i-th frame for classification.

The comparison result is presented in Table 2. In accordance with expectations, the pose prediction module brought a reduction to the accuracy of falls classification.

However, $Model_{\text{pred+cls}}$ is capable of predicting falls in advance. For a fairer comparison, we conducted another experiment to enable $Model_{\text{cls}}$ to 'foresee' the falls by early annotation. Specifically, the labels of the falling proceeding (*i.e.*, from frame S_{fs} to S_{fe}) were modified to 'fall' in the training of $Model_{\text{cls}}$. As shown in Table 3, compared to the pose prediction method, early annotation improves the recall at the cost of an obvious drop on the precision, which is caused by the significant increase of false positive.

Table 3. Comparison between $Model_{\text{cls}}$ and $Model_{\text{pred+cls}}$ with modified annotations

Model	Acc.	Prec.	Rec.	F1
$Model_{\text{cls}}$	98.1%	87.7%	**99.7%**	0.933
$Model_{\text{pred+cls}}$	**98.7%**	**92.6%**	98.0%	**0.952**

4.4. Results

We present several experimental results in Fig. 6. In most scenarios, our model can correctly predict the falling event in advance. However, (d) and (e) show two failure cases caused by different reasons. In (d), many keypoints are undetected by OpenPose due to the invisibility of the person's upper body under that camera perspective. As a consequence, the pose prediction module produces absurd keypoints sequence providing insufficient features for classification. (Recall that vectors with less than 8 detected keypoints will be prejudged as 'unknown' before the falls classifier). While in (e), no 'precursor' of fall is discovered in the observed frame sequence, which is beyond the ability of the pose prediction module to predict the future falls.

5. Conclusion

In this work, we propose a model combining a pose prediction module and a falls classifier for the precognition of falls. The pose prediction module generates future keypoints vector sequence based on the observation. Then the falls classifier takes the predicted keypoints vector as input and judges whether it's a fall. Evaluations has proved the superiority of keypoints features for falls classification and the effectiveness of the pose prediction module.

References

[1] F. Bianchi et al. Falls event detection using triaxial accelerometry and barometric pressure measurement. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6111–6114, 2009.

[2] F. Bianchi et al. Barometric pressure and triaxial accelerometry-based falls event detection. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(6):619–627, 2010.

[3] A. K. Bourke, J. V. O'brien, and G. M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait & posture*, 26(2):194–199, 2007.

[4] 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.

[5] 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*, 22(4):041106, 2013.

[6] J. Deng et al. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

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

[8] K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2012.

[12] A. Karpathy et al. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[13] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Y. Kong and Y. Fu. Human action recognition and prediction: A survey. *arXiv preprint arXiv:1806.11230*, 2018.

[15] Y. Kong, S. Gao, B. Sun, and Y. Fu. Action prediction from videos via memorizing hard-to-predict samples. In *AAAI Conference on Artificial Intelligence*, 2018.

[16] Y. Kong, Z. Tao, and Y. Fu. Deep sequential context networks for action prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1473–1481, 2017.

[17] H. Kuehne et al. Hmdb: a large video database for human motion recognition. In *International Conference on Computer Vision*, pages 2556–2563, 2011.

[18] C. Li, Z. Zhang, L. W. Sun, and L. G. Hee. Convolutional sequence to sequence model for human dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5226–5234, 2018.

[19] T. Y. Lin et al. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755, 2014.

[20] S. Ma, L. Sigal, and S. Sclaroff. Learning activity progression in lstms for activity detection and early detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1942–1950, 2016.

[21] X. Ma et al. Depth-based human fall detection via shape features and improved extreme learning machine. *IEEE journal of biomedical and health informatics*, 18(6):1915–1922, 2014.

[22] J. Martinez, M. J. Black, and J. Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.

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

[24] M. R. Narayanan et al. Falls management: detection and prevention, using a waist-mounted triaxial accelerometer. In *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4037–4040, 2007.

[25] J. Quero et al. Detection of falls from non-invasive thermal vision sensors using convolutional neural networks. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2, page 1236, 2018.

[26] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[27] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[28] E. E. Stone and M. Skubic. Evaluation of an inexpensive depth camera for passive in-home fall risk assessment. In *5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 71–77, 2011.

[29] E. E. Stone and M. Skubic. Fall detection in homes of older adults using the microsoft kinect. *IEEE journal of biomedical and health informatics*, 19(1):290–301, 2014.

[30] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[31] Y. Tang, L. Ma, W. Liu, and W. Zheng. Long-term human motion prediction by modeling motion context and enhancing motion dynamic. *arXiv preprint arXiv:1805.02513*, 2018.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[33] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[34] O. Vinyals and Q. Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[35] L. Wang et al. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36, 2016.

[36] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE, 2017.