International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 09 | Sep 2019 www.irjet.net p-ISSN: 2395-0072

Human Fall Detection using Co-Saliency-Enhanced Deep Recurrent Convolutional Neural Networks

Chenjie Ge[1,2], Irene Yu-Hua Gu[1], Jie Yang[2]

[1]Dept. of Electrical Engineering, Chalmers Univ. of Technology, Gothenburg, Sweden
[2]Inst. of Image Processing and Pattern Recognition, Shanghai JiaoTong Univ., Shanghai, China

---

Abstract - *This paper addresses issues of fall detection from videos for e-healthcare and assisted-living. Instead of using hand-crafted features from videos, we exploit a dedicated recurrent convolutional network (RCN) architecture for fall detection in combination with co-saliency enhancement. In the proposed scheme, the recurrent neural network (RNN) is realized by Long Short-Term Memory (LSTM) connecting to a set of Convolutional Neural Networks (CNNs), where each video is modelled as an ordered sequence, containing several frames. In such a way, the sequential information in video is preserved. To further enhance the performance, we propose to employ co-saliency-enhanced video frames as the inputs of RCN, where salient human activity regions are enhanced. Experimental results have shown that the proposed scheme is effective. Further, our results have shown very good test performance (accuracy 98.12%), and employing the co-saliency-enhanced RCN has led to the improvement in performance (0.70% on test) as comparing to that without co-saliency. Comparisons with two existing methods have provided further support to effectiveness of the proposed scheme.*

Key Words: Human fall detection, E-healthcare, Recurrent convolutional network, Long Short-Term Memory, Co-saliency enhancement.

1. INTRODUCTION

Fall detection has drawn much interest in e-healthcare and assisted-living especially for elderly. When many elderly prefer to live a quality life at home, fall is one of the main risks that require attentions. Falls can cause significant impact, sometimes leading to, among many others, severe injury, bone fracture and stroke. Seeking immediate medical help after a fall is often not feasible as the person might be unable to move or seek emergent help. There is a growing interest in assisted-living that can provide such services and trigger emergent medical attentions after a severe fall accident. Wearable devices with motion sensors, such as accelerometers and gyroscopes[1], provide some solution to fall detection. However, elderly often feel uncomfortable after wearing such devices for a long time. Sometimes they forget to wear them, or to charge the battery of these devices. Extracting the information from visual monitoring can provide an alternative solution to the aforementioned problems when the privacy issue is properly handled.

Methods for classifying human activities can be roughly subdivided into two categories: one set of methods use hand-crafted features based on human experts' knowledge and machine learning models. For example, one way is to compass the target person in each frame using a bounding box and extract features, e.g. using HoG and HoF. Qian et al. [2] proposed a two-bounding-box strategy for characterizing and extracting features from the whole body and the lower part of body. Yun et al. [3] characterized the dynamic features of a target person by their appearance, shape and motion as points moving on a Riemannian manifold. Another commonly adopted way is to use videos captured from multiple cameras or depth cameras. Rougier et al. [4] proposed to calculate the cost between consecutive frames for shape deformation. Ma et al.[5] proposed to learn the curvature scale space (CSS) features from human silhouettes using depth images, where actions were represented by a bag of CSS words and then classified by an extreme learning machine (ELM).

Another category of methods is to automatically learn the features related to human activities by using deep learning. Simonyan et al. [6] proposed a two-stream convolutional network, where RGB frames and stacks of optical flow fields were used to separately capture spatial and temporal information. Tren et al. [7] exploited 3D ConvNets to learn spatio-temporal features from videos without calculating optical flow. Ng et al. [8] investigated several temporal feature pooling methods and LSTM to learn CNN features across long time periods, and showed temporal pooling of CNN features performed better. Ge et al. [9] proposed to extract two-stream CNN features followed by sparse dictionary learning to characterize each activity and detect human falls. Fan et al. [10] considered four phases in each fall (standing, falling, fallen and not moving) and trained deep CNN to distinguish four categories of dynamic images corresponding to the above four phases. Zhang et al. [11] proposed to use trajectory attention map to enhance the CNN feature of each frame, and rank-pooling-based encoding method was then used to obtain the feature descriptor for each activity.

Observing that a video object often occupies a very small image area, it is desirable to highlight object areas and let the deep learning focus on these specific areas, for more robust classification of dynamic human activities. Therefore, using a co-saliency detection method is expected to enhance foreground video object regions and

suppress unwanted background areas at the same time. It is anticipated that co-saliency enhancement not only may improve the performance, but also may lead to faster convergence in deep learning. Motivated by the ideas of exploring long-term spatio-temporal dynamic features, we propose to apply LSTM on a set of CNNs operated on the video segment-level to exploit the temporal relationship in dynamic human activities. A novel co-saliency-based recurrent convolutional network architecture (RCN) is proposed for distinguishing fall/non-fall human activities in videos. To the best of our knowledge, this is the first effective integration of co-saliency-enhancement of video objects with a recurrent convolutional network architecture that is dedicated for human fall detection in videos.

The main contributions of this paper include: 1) propose a novel integrated fall detection scheme by combining co-saliency-enhancement and recurrent convolutional network (RCN) architecture for fall detection; 2) employ a co-saliency method that is particularly suitable for enhancing dynamic human areas and suppressing static background areas in videos; 3) numerous empirical tests on selecting parameters and using different settings.

2. CO-SALIENCY-ENHANCED DEEP RECURRENT CONVOLUTIONAL NETWORK

In this section, we describe the proposed co-saliency-enhanced deep learning scheme.

2.1 Overview of the Proposed Scheme

The basic ideas behind the proposed scheme are: (a) Applying a co-saliency-based method to enhance dynamic human activity regions and suppress unwanted background regions. Integrating this may lead to obtaining more discriminant features of human falls from the deep learning, hence improvement on the overall performance. A co-saliency method, a special type of saliency-based methods, is designed to detect salient regions based on several image frames in video. (b) Exploiting the long-term spatio-temporal relationship of dynamic activities by applying LSTM on a set of convolutional networks (CNNs) in the segment-level. Applying segment-level CNNs is based on the observation that, by partitioning each video clip into small segments, images within each segment can be considered as approximately stationary, hence a CNN can be used effectively in such a segment-level. For the longer time scale, LSTM is the applied to take into account of the temporal dependency of segment-level images.

Fig.1 illustrates the pipeline of the proposed scheme, where co-saliency activity region enhancement is applied on the original video frames, followed by a recurrent convolutional neural network (RCN) architecture where a set of CNNs is applied in image frames in the segment-level.

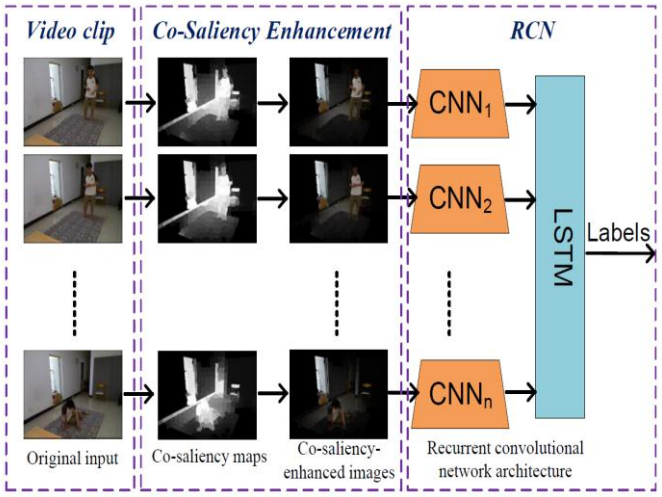

Fig. 1: The pipeline of the proposed fall detection method.

2.2 Deep Recurrent Convolutional Network (RCN)

This subsection describes the proposed deep RCN in detail. The proposed RCN in the 2nd block of Fig.1 can be further depicted by Fig.2. It shows the detailed layers and settings of the entire RCN architecture, which are obtained after our numerous empirical tests and hyperparameter tunings. This RCN architecture is dedicated to the fall detection task. For fall detection in our method, each video clip is first divided into N segments. In each segment, one key frame is extracted as the representative. Hence, video clips of different lengths can be normalized to the same one, independent of the speed of each activity. In the proposed RCN, there are N CNNs (each is of 5-layers) followed by a single-layer LSTM with 256 hidden units. The class labels (fall/non-fall) are obtained from the output of last FC layer.

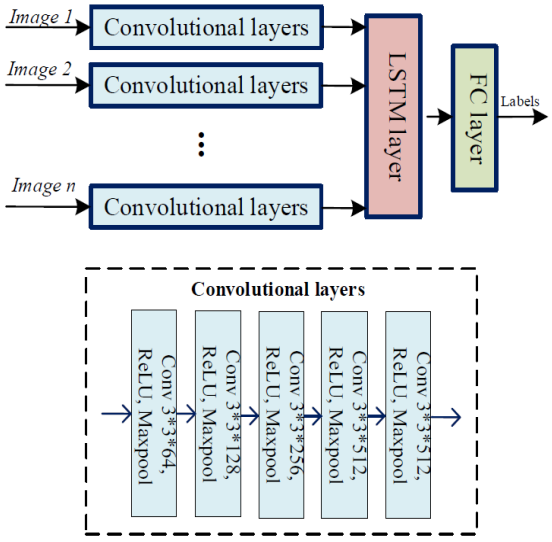

Fig. 2: Architecture of the deep recurrent convolutional network (RCN).

Each CNN in the proposed scheme consists of five layers, which is carefully selected after numerous empirical tests. We used a small 3*3 filter kernel in each layer, similar to the filter settings in VGG net [12]. By using small kernel filters in multiple layers, complex features of each frame can be learned. Each convolution is followed by ReLU activation function and maxpooling to decrease the size of feature map and thus generate higher-level semantic features. Features from CNNs are then extracted from the final convolutional layer (i.e., 5th layer) and fed to the LSTM.

LSTM is used to learning the temporal dynamics of static CNN features for each activity. Basic LSTM unit contains a single memory cell, an input activation function and four different gates (input gate i_t, forget gate f_t, output gate o_t and input modulation gate g_t) [13]. Input gate controls whether the incoming signal will alter the state of the memory cell or block it. Forget gate allows the cell to selectively forget something and can somehow prevent the gradient from vanishing or exploding during back propagation through time. Output gate makes the memory cell have an effect on other neurons or prevent it. Input modulation gate is the function of the current input and previous hidden state. With help of these gates, LSTM is able to capture extremely complex, long-term temporal dynamics and overcome the vanishing gradient problem as well. For an input x_t at time step t, memory cell state c_t encodes everything the cell has observed until time t, and finally LSTM outputs the hidden/control state h_t:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (2)$$

$$o_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (3)$$

$$g_t = \phi(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (5)$$

$$h_t = o_t \odot \phi(c_t) \quad (6)$$

where $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoidal nonlinear function which maps real-valued inputs into the interval [0, 1], $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ is the hyperbolic tangent non-linear function that maps its inputs into the interval [-1,1], $\odot$ is the element-wise product, W is the weight matrix, b is the bias.

In the proposed scheme single layer of LSTM is used as suggested in [14], since it is enough for learning the temporal information of CNN features. Adding more layers of LSTM does not improve the performance but increases computational cost according to our experimental results. At last, on fully-connected (FC) layer with 2 output neurons was connected to LSTM, to output the activity label (fall/non-fall).

2.3 Co-Saliency-Based Human Activity Area Enhancement

Co-saliency detection is an efficient method to detect common salient objects from a set of images in video. In our case, we only focus on foreground dynamic human objects in a video clip. In this approach, original image frames are first enhanced by the co-saliency-based method [15] through a two-stage saliency propagation. Fig.3 shows the block diagram.

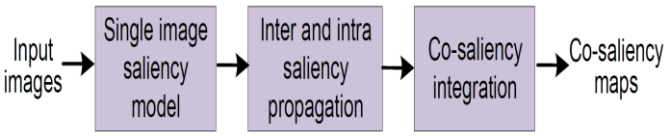

Fig. 3: The block diagram of co-saliency detection.

In the method, an initial saliency map is first generated by superpixel-based saliency detection in each frame. An inter-saliency propagation step is then used to propagate saliency values between two image frames according to their color similarity. Center-bias and geodesic distance-based smoothing are then applied as post-processing to alleviate the color fluctuation. This is followed by intra-saliency propagation. A graph-based regularized energy function is used by considering boundary connectivity-induced background cue, saliency mask-induced foreground cue and a smoothness constraint.

$$E(s) = \sum_{i=1}^{K_m} w_i^{bg} s_i^2 + \sum_{i=1}^{K_m} w_i^{fg} (s_i - 1)^2 + \sum_{i,j} w_{ij} (s_i - s_j)^2 \quad (7)$$

where s_i and s_j are the i th and j th saliency values of superpixels, K_m is the number of superpixels in the m th image. The first item in (7) applies a small weight value w_i^{bg} (near 0.0) if a superpixel s_i shows a large background cue (and vice versa); the second item uses a large weight w_i^{fg} (near 1.0) if a superpixel s_i has a large co-salient foreground cue; while the third item is the smoothness term, where the weight w_{ij} is set to ensure the continuous saliency values in nearby superpixels. More details can be found in [15].

3. EXPERIMENTS AND RESULTS

3.1 Dataset and Setup

Dataset: Experiments were conducted on an open video dataset "ACT4^2 Dataset" from [16], where videos were captured by 4 Kinect sensors. Only RGB videos in this dataset were used in our tests. In the dataset, each activity was performed by 24 subjects, captured in 4 view angles and repeated 2 times. Hence, there are 192 videos for each video activity. Table-1 summarizes the dataset.

Table -1: Description of the dataset.

	Collapse	Stumble	Pickup	Sitdown	Total
#Video clips	192	192	192	192	768
#Views	4	4	4	4	4
#View repeat	2	2	2	2	2
#Subjects	24	24	24	24	24

In the proposed scheme, fall detection is formulated as a binary classification problem: we treat the falls (including Collapse and Stumble) as the positive class, and the remaining activities (including Pickup and Sitdown) as the negative class. Fig.4 shows some key video frames from these 2 classes in "ACT4^2 Dataset".

Fig. 3: Example of the key frames from "ACT4^2 Dataset". Top two rows: from human falls (1st row: collapse, 2nd row: stumble). Bottom two rows: from other activities (3rd row: pickup, 4th row: sitdown).

Setup: In our experiments, KERAS library [17] with TensorFlow [18] backend was used for the experiments. Network weights were learnt using Adagrad optimizer with learning rate 0.0005 and decay rate 0.0005. Batch size was set to 8. The number of epochs was chosen as 300 and cross-entropy loss was used for the training. Early stopping strategy [19] was adopted during the training process, where parameters of the network were fixed from a selected epoch when the best validation performance was achieved. For data augmentation, flip and crop strategies were used similar to [6]. All image frames were first resized to 144*192. In the training process, a 128*128 sub-image was randomly cropped from selected frames followed by random horizontal flipping. The central 128*128 part was cropped for validation and testing without random flipping. Noting that frames from a given video clip were cropped and flipped in the same way. Video clips in the dataset were partitioned into 3 subsets according to training (50\%), validation (12.5\%), testing (37.5\%).

For evaluation, we adopt overall accuracy, sensitivity and specificity as the performance criteria. These objective metrics are based on the following four kinds of samples.

True positive: the fall activity, and is correctly classified as fall activity.

False positive: the non-fall activity, but is incorrectly classified as the fall activity.

True negative: the non-fall activity, and is correctly classified as non-fall activity.

False negative: the fall activity, but is incorrectly classified as non-fall activity.

Let TP, FP, TN and FN be the number of true positives, false positives, true negatives and false negatives, the three metrics overall accuracy, sensitivity and specificity are defined as follows:

Overall accuracy is defined by (TP+TN)/(TP+FP+TN+FN); *Sensitivity* is defined by TP/(TP+FN);

Specificity is defined by TN/(FP+TN).

3.2 Performance of the Proposed Scheme

To test the effectiveness of the proposed method on fall detection, experiments were conducted for 5 runs, where partitions of training, validation and test subsets in the dataset were done randomly in each of the 5 runs. Table-2 shows the accuracy of the proposed method on the testing set as well as the sensitivity and specificity, including the average performance of 5 runs and standard deviation |σ|.

Table -2: Performance of the proposed scheme in 5 runs.

Run	Accuracy (%)	Sensitivity (%)	Specificity (%)
1	**98.26**	**97.91**	98.61
2	**98.26**	96.52	**100.00**
3	**98.26**	96.52	**100.00**
4	97.91	96.52	99.30
5	97.91	97.22	98.61
Average	98.12	96.93	99.30
|σ|	0.1917	0.6222	0.6950

Observing Table-2, the proposed method is shown to be effective on the testing set, with a relatively high average classification accuracy 98.12% in 5 runs (|σ|=0.1917), with sensitivity 96.93\% (|σ|=0.6222) and specificity 99.30\% (|σ|=0.6950). Furthermore, the proposed method seems to have relatively balanced performance on different classes since the sensitivity and specificity are relatively close to each other.

3.σ Impact of Co-Saliency Enhancement

To further examine the proposed scheme, we compare the proposed scheme with and without applying co-saliency enhancement on the testing set (i.e. with/without the 2nd block in Fig.1). Table-3 shows the classification accuracy on the training, validation and test sets, and Fig.4 shows the performance curves (on the validation set) from with/without applying co-saliency enhancement.

Table -3: Performance comparison with/without co-saliency enhancement.

Metric	Without saliency	With saliency
Accuracy (%)	97.56	**98.26**
Sensitivity (%)	95.83	**97.91**
Specificity (%)	**99.30**	98.61

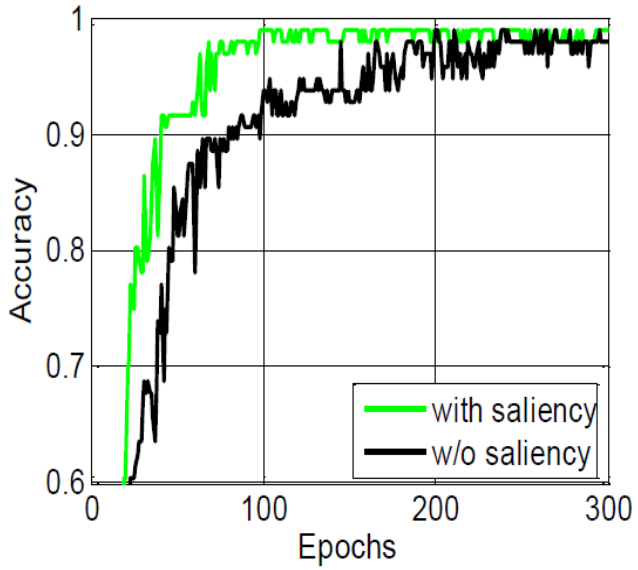

Fig. 4: Performance on the validation set with/without co-saliency enhancement: accuracy vs. epochs.

One can see from Table-3 that the proposed scheme has obtained better performance with the co-saliency enhancement. Accuracy of the method without saliency was already very high (accuracy 97.56% with sensitivity 95.83% and specificity 99.30%), and by employing saliency test accuracy increased 0.7%. It also indicates that the performance of fall detection depends on salient regions in videos. Observing Fig.4, it seems that introducing co-saliency enhancement has also led to faster convergence, where the proposed scheme converged at around epoch 100 while that without saliency converged at around epoch 200.

3.τ Performance from using Different Dataset Partitions

To examine the performance on the testing set, Table-4 shows the overall performance, sensitivity and specificity on using three different partitioned methods (i.e. different way of partitioning the dataset into training, validation, testing subsets) according to subjects, views and video clips. Observing the results in Table-4, dataset with a random video clip partition performs the best. This is due to all the video clips were mixed without considering different subjects and different view angles, such that videos from the same subject but different views may end up in training and testing sets at the same time. Different partitions according to subjects and views have also led to high accuracy (96.52% and 94.09%), which shows robustness of the proposed scheme.

Table -4: Performance of the proposed scheme using different data partitions: according to subjects/views/video clips.

Partition	Accuracy (%)	Sensitivity (%)	Specificity (%)
Subjects	96.52	95.13	97.92
Views	94.09	93.75	94.44
Videos	**98.26**	**97.91**	**98.61**

3.υ Analysis through Empirical Tests under Different Parameter Settings

To further analyze the performance and the robustness of the proposed scheme empirically, we have conducted numerous tests from using several different parameter settings, such as changing the number of segments in each activity, changing the number of LSTM layers and number of hidden units in LSTM.

(a) Using different number of segments for each video:

To further examine the proposed scheme, we compare the proposed scheme by changing the number of segments in each activity. Fig.5 shows the performance comparison of using 4,8,12 and 16 segments on the testing set.

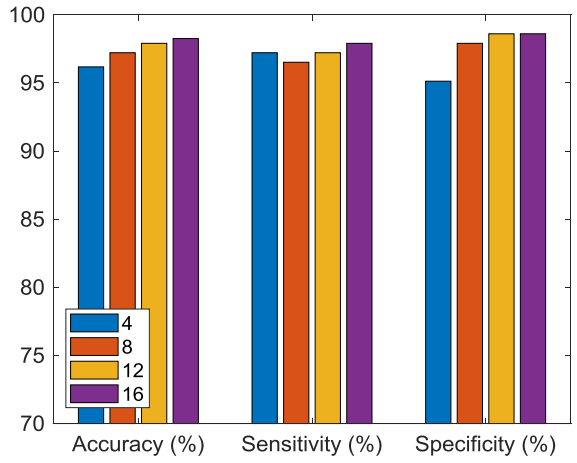

Fig. 5: Performance comparison on the testing set using different number of segments in each video.

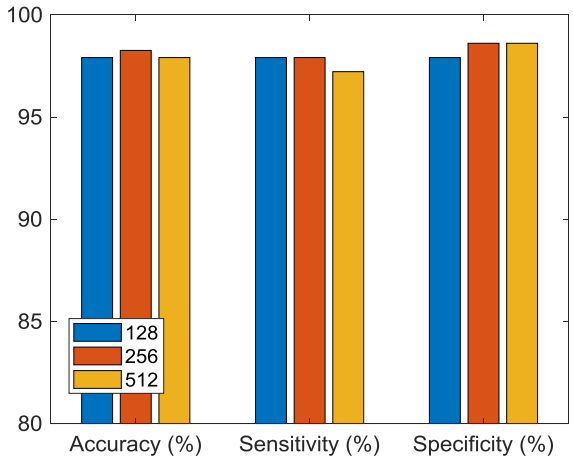

Fig. 6: Performance comparison on the testing set using different number of hidden units in LSTM.

It is observed that using 16 segments leads to the best accuracies in all three metrics, thus in the proposed scheme the number of segments is chosen to 16. Such result is reasonable as more temporal segments can capture richer temporal characteristics in an activity. The reason we use 16 instead of more segments is to tradeoff the computational cost and the accuracy, since the improvement of performance seems marginal when more segments are used.

(b) Using different number of LSTM layers:

We compare the proposed scheme by changing the number of LSTM layers in RCN. Table-5 shows the performance comparison of using 1 and 2 layers of LSTM on the testing set.

Table -5: Performance comparison using 1 and 2 layers of LSTM.

Metric	1-layer LSTM	2-layer LSTM
Accuracy (%)	**98.26**	96.52
Sensitivity (%)	**97.91**	95.13
Specificity (%)	**98.61**	97.91

Observing Table-5, adding one more layer of LSTM does not seem to increase the test accuracy, meaning that one layer of LSTM is sufficient to capture the spatio-temporal information of each activity.

(c) Using different number of hidden units in LSTM:

The number of hidden units in LSTM determines the dimension of cell state vector. To evaluate the performance when changing the number of units, Fig.6 shows the performance comparison on the testing set.

It is observed that using 256 segments leads to the best performance (98.26%), thus in the proposed scheme the number of hidden units is chosen to 256. It seems that by adding more hidden units (from 256 to 512) in LSTM does not increase the general performance on the testing set.

3.φ Comparison

To further evaluate the performance, we have compared the results of the proposed scheme with several state-of-the-art methods (that used the same dataset). Videos from different camera views and different subjects were mixed and treated equally in this test. For comparisons, Table-5 shows the results from 2 existing methods (using non-deep learning methods) and the proposed deep learning scheme. Observing the results in Table-6, one can see that the proposed method has obtained the highest performance, however, only a slight improvement from [20]. Despite that, the proposed scheme shows that co-saliency-based deep learning scheme, through automatically learning features from enhanced videos, may perform equally good or even better, as compared with the classification methods using hand-crafted features.

Table -6: Performance of 2 existing methods and the proposed scheme.

Method	Sensitivity (%)	Specificity (%)
Cheng [16]	89.06	92.71
Yun [20]	**98.13**	94.48
Proposed	96.93	**99.30**

3.χ Discussion

From the above empirical test results, the proposed scheme is shown to be effective for human fall detection

from videos. From the various sets of experiments and results, we may also draw the following conclusions:

1) Overall performance: The proposed scheme is effective, with excellent average test accuracy (98.12%), sensitivity (96.93%) and specificity (99.30%).

2) Using co-saliency enhancement: The co-saliency enhancement has led to an increase in test performance (from 97.56% to 98.26%). Introducing co-saliency enhancement has also led to faster convergence in the training process.

3) Using different dataset partitions: Empirical tests show that the proposed scheme has achieved high accuracies using three different partitions of the dataset. This shows the robustness of the proposed scheme.

4) Using different parameter settings: After careful tuning of the parameters including different number of segments in each video, different number of LSTM layers and different number of hidden units in LSTM, the proposed scheme has achieved the best performance.

5) Comparison with the state-of-the-art: The proposed deep learning scheme has achieved the performance comparable to the two existing methods, suggesting that it can perform equally good or even better compared with the classification methods using handcrafted features.

Future work: Currently the proposed scheme has been tested on one dataset for human fall detection. Future work includes tests on more datasets and extending the proposed scheme to classification on more classes of daily activities for assisted-living and e-healthcare.

4. CONCLUSION

We have proposed and tested the fall detection scheme which is an integration of co-saliency-enhancement and recurrent convolutional network. Our results have shown that the co-saliency-detection method is effective for enhancing the dynamic human activities and suppressing the unwanted background in videos, and the recurrent convolutional network architecture is simple and effective for learning the time-dependent features of human activities. Integrating the co-saliency enhancement and recurrent convolutional network has led to the improved classification performance. Our test results have also shown that parameters and settings of the proposed scheme may impact the performance and hence require to be carefully tuned. Furthermore, test results from the proposed scheme by using different partitioned ways (to generate different split of training, validation and test subsets) have also shown that the scheme is robust to unseen views and unseen subjects.

REFERENCES

[1] X. Yu, “Approaches and principles of fall detection for elderly and patient,” In HealthCom 2008-10th International Conference on e-health Networking, Applications and Services. IEEE, 2008, pp. 42–47.

[2] H. Qian, Y. Mao, W. Xiang, and Z. Wang, “Home environment fall detection system based on a cascaded multi-svm classifier,” in Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on. IEEE, 2008, pp. 1567–1572.

[3] Y. Yun and I. Gu, “Human fall detection in videos via boosting and fusing statistical features of appearance, shape and motion dynamics on riemannian manifolds with applications to assisted living,” Computer Vision and Image Understanding, vol. 148, pp. 111–122, 2016.

[4] C. Rougier, J. Meunier, Alain St-Arnaud, and Jacqueline 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.

[5] X. Ma, H. Wang, B. Xue, M. Zhou, B. Ji, and Y. Li, “Depth-based human fall detection via shape features and improved extreme learning machine,” IEEE journal of biomedical and health informatics, vol. 18, no. 6, pp. 1915–1922, 2014.

[6] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in Advances in neural information processing systems, 2014, pp. 568–576.

[7] 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, 2015, pp. 4489–4497.

[8] J. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, “Beyond short snippets: Deep networks for video classification,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 4694–4702.

[9] C. Ge, I. Gu, and J. Yang. "Human fall detection using segment-level CNN features and sparse dictionary learning." In 2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP), pp. 1-6. IEEE, 2017.

[10] Y. Fan, M. Levine, G. Wen, and S. Qiu. "A deep neural network for real-time detection of falling humans in

naturally occurring scenes." Neurocomputing 260 (2017): 43-58.

[11] Z. Zhang, X. Ma, H. Wu, and Y. Li. "Fall Detection in Videos With Trajectory-Weighted Deep-Convolutional Rank-Pooling Descriptor." IEEE Access 7 (2018): 4135-4144.

[12] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[13] W. Zaremba and I. Sutskever. "Learning to execute." arXiv preprint arXiv:1410.4615 (2014).

[14] J. Donahue, L. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko and T. Darrell. "Long-term recurrent convolutional networks for visual recognition and description." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2625-2634. 2015.

[15] C. Ge, K. Fu, F. Liu, L. Bai and J. Yang. "Co-saliency detection via inter and intra saliency propagation." Signal Processing: Image Communication 44 (2016): 69-83.

[16] Z. Cheng, L. Qin, Y. Ye, Q. Huang, and Q. Tian. "Human daily action analysis with multi-view and color-depth data." In European Conference on Computer Vision, pp. 52-61. Springer, Berlin, Heidelberg, 2012.

[17] F. Chollet et al., "Keras," https:// github.com/fchollet/keras, 2015.

[18] M. Abadi, A. Agarwal et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, Software available from tensorflow.org.

[19] Y. Yao, L. Rosasco, and A. Caponnetto. "On early stopping in gradient descent learning." Constructive Approximation 26, no. 2 (2007): 289-315.

[20] Y. Yun and I. Gu. "Human fall detection in videos by fusing statistical features of shape and motion dynamics on Riemannian manifolds." Neurocomputing 207 (2016): 726-734.