CUT AND CONTINUOUS PASTE TOWARDS REAL-TIME DEEP FALL DETECTION

Sunhee Hwang, Minsong Ki, Seung-Hyun Lee, Sanghoon Park, and Byoung-Ki Jeon

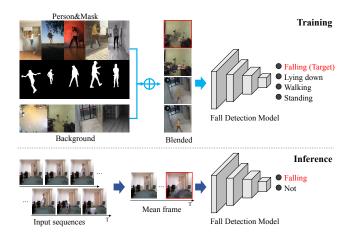LG Uplus Corp.

sunheehwang@lguplus.co.kr

ABSTRACT

Deep learning based fall detection is one of the crucial tasks for intelligent video surveillance systems, which aims to detect unintentional falls of humans and alarm dangerous situations. In this work, we propose a simple and efficient framework to detect falls through a single and small-sized convolutional neural network. To this end, we first introduce a new image synthesis method that represents human motion in a single frame. This simplifies the fall detection task as an image classification task. Besides, the proposed synthetic data generation method enables to generate a sufficient amount of training dataset, resulting in satisfactory performance even with the small model. At the inference step, we also represent real human motion in a single image by estimating mean of input frames. In the experiment, we conduct both qualitative and quantitative evaluations on *URFD* and *AIHub airport* datasets to show the effectiveness of our method.

Index Terms— Real-time Fall Detection, Video Surveillance, Image Blending, Deep Neural Networks

1. INTRODUCTION

Falling is one of the risky factors causing unintentional injury, particularly for the elderly, lone workers, and patients. According to the WHO [1], approximately 684,000 fatal falls occur, resulting in deaths each year globally. To enable quick responses to such accidents, automated monitoring systems are getting widely used in nursing homes, manufacturing industries, and hospitals. Specifically, a variety of fall detection methods are proposed to detect the human falling status based on diverse kinds of sensors, *e.g.*, accelerometers, gyroscopes, biometric, and visual sensors. Among these, a vision-based method is called an effective way, since it is non-invasive and there is no hassle of wearing devices. Moreover, it is easy to apply the method to existing video surveillance systems.

Most of the existing vision-based fall detection methods adopt a multiple-stage-based approach, which consists of two or more modules [2, 3, 4, 5, 6]. Conventionally, the modules

Fig. 1: Overview of the proposed fall detection method. Synthetic images and real videos are used for training and test, respectively.

are composed as follows: person detection, tracking person, and falling state classification modules. This approach can achieve high accuracy as it leverages the external knowledge of each module trained on large-scale datasets. On the other hand, they have difficulty with system maintenance, since one module can affect the overall performance in terms of accuracy and inference time. In addition, each module has to be updated to achieve better performance, which requires time-consuming data collection and labeling work.

Unlike the multiple-stage-based approach, several methods simplified the fall detection framework by adopting a single convolutional neural network (CNN) based models [7, 8, 9, 10]. To this end, they conventionally train the fall detection models utilizing raw images or pre-processed motional images. Then, they predict the falling state using the trained single CNNs. This is a simple way but has the following difficulties. Since most available fall detection datasets are obtained from limited person and place, they contain only a small number of samples for training the deep neural networks. For that reason, it is not easy to obtain outstanding performance without utilizing external knowledge. As an alternative way, several methods [7, 9] adapt large CNNs such as VGG-16 to improve performance. However, it is still hard to obtain excellent performance by training the deep models with small datasets.

| (a) Person & Mask | (b) Background | (c) r_1 | (d) r_2 | (e) r_3 | (f) r_N | (g) $avg(\mathbf{r})$ |

Fig. 2: The overall procedure of the proposed image blending method. Given person&mask (a) and background image (b), we cutout person region from (a) and paste it into (b) after applying transform $\mathbf{t}$: (c)-(f). Then, we average the images (c)-(f): (g). Finally, (f) and (g) are utilized to train the fall detection model as motionless and motional images, respectively.

From the existing methods, we observe that the fall detection models utilizing big-sized models or large-sized datasets tend to outperform others. Unlike priors, we aim to design a new fall detection framework that performs well on the small-sized models without utilizing the large-sized datasets. To this end, inspired by [11, 12], we introduce an image synthesis method for generating large-sized training datasets. We visualize the overall process in Figure 1. Specifically, given person and background images, we generate synthetic falling/non-falling samples. To simplify the fall detection framework, we represent all the samples in a single frame. Note that our method enables to generate the large-sized dataset without acting in dangerous situations for data acquisition. Then, we train the single and small-sized models with the generated training data. At the inference step, we also represent real motion in a single frame. For this, we estimate the mean of sequential images and predict its label through the trained model. This approach simplifies the fall detection method as an image classification task and enables real-time processing.

In experiment, we conduct two different fall detection datasets: *URFD* [13] and *AIHub airport*[1]. To verify the superiority of our method, we compare the fall detection procedures with the existing methods. We also compare the execution time and accuracy among various CNNs. Overall, the proposed approach achieves best performance on EfficientNet-B0 [14] with the accuracy of 97.14% and 92.25% on *URFD* and *AIHub* datasets, respectively.

In summary, our main contributions are as follows:

- We propose a simple yet effective fall detection framework, reducing hardware and dataset requirements.

- Through the quantitative and qualitative results on *URFD* and *AIHub* datasets, we validate the effectiveness of the proposed method.

2. PROPOSED METHOD

2.1. Image Synthesis

In this section, we describe the proposed image blending method, aiming to generate a large number of training samples. As the main task of this study is fall detection, we first define four actions to distinguish between falling and non-falling. The action categories are as follows: falling, walking, standing, and lying down. Figure 2 shows the overall procedure of our blending method. Given person images and their mask images (a), we cut them out and paste them into the background (b) to create the target images, *i.e.*, (f) and (g).

Let a person image as x and the corresponding binary mask as m. Before generating synthetic human motion images, we apply an online data augmentation approach with image transformation f to x, where f is composed of different kinds of randomly selected image jittering functions to ensure the diversity of the training images. Afterwards, given $\hat{x} = f(x)$, we cutout the person segments using m, and then apply geometrical transformation $\mathbf{t} \in \{t_1, t_2, ..., t_N\}$, where $\mathbf{t}$ and N denote a set of transformation functions to generate continuous human motion and the number of moments for the motion, respectively. In other words, the transformation $\mathbf{t}$ is designed to reflect the falling or walking motion, resulting in the N sequential images. Then, we individually paste the N images, *i.e.*, $t_1(\hat{x}), t_2(\hat{x}), ..., t_N(\hat{x})$, into background image b. As a results, we attain the blended images $\mathbf{r} \in \{r_1, r_2, ..., r_N\}$ representing a series of moment for human motion. While our goal is to train a single CNN without utilizing additional sequential modules, thus, we average $\mathbf{r}$ (a set of N blended images) to represent human motion in a single image. Finally, we utilize $r_N \in \{lying\ down, standing\}$ and $avg(\mathbf{r}) \in \{falling, walking\}$ for training samples of motionless image and motional image, respectively.

Table 1: Fall detection performance comparison between our approach and exiting methods on the URFD dataset. The Second column denotes the architecture is composed of a single CNN or not. The third column indicates the usage of URFD dataset for model training, where D_{tr} denotes the training dataset. We also note the number of parameters of single CNN-based models.

Ref.	Single CNN	URFD in D_{tr}	Architecture	Sensitivity	Specificity	Precision	Accuracy	Params
[2]	✗	✓	P-CNN + Tensorized-LSTM	**100**	97.44	-	**99.00**	-
[3]	✗	✓	OpenPose + LSTM	**100**	96.40	-	98.20	-
[4]	✗	✓	Mask-RCNN + VGG16 + Bi-LSTM	91.80	**100**	**100**	96.70	-
[5]	✗	✓	YOLO V3 + DeepSort + LSTM	93.10	-	94.80	-	-
[6]	✗	✗	PifPaf + Rule-based	83.33	**100**	-	-	-
[7]	✓	✓	VGG16	-	-	-	92.34	138M
[8]	✓	✓	BGS + LeNet	88.00	-	89.00	89.00	60K
[9]	✓	✓	Modified VGG16	**100**	92.00	-	95.00	138M+
[10]	✓	✓	Shallow-CNN	41.47	-	31.04	88.55	104K
Ours	✓	✗	EfficientNet-B0	93.33	**100**	**100**	**97.14**	4M

2.2. Falling State Classification

We then train the fall detection model on the generated synthetic dataset to classify the four categories (*i.e.*, lying down, standing, falling, and walking). Note that we utilize one of the small and single CNNs for fall detection. Let $s^i \in \{r_N, avg(\mathbf{r})\}$ is one of the randomly generated human motional image. We compose the mini-batch with M images $\mathbf{s} \in \{s^1, s^2, ..., s^M\}$. Then, we train the model g to predict corresponding human motional label $\mathbf{y} \in \{y^1, y^2, ..., y^M\}$, where g is optimized by the following cross-entropy loss function:

$$\mathcal{L}_{ce} = -\sum_{i=1}^{M} y^i log(g(s^i)). \qquad (1)$$

For the inference, given real images from a camera, *i.e.*, $\mathbf{u} \in \{u^1, u^2, ..., u^\infty\}$, we start detecting falls from time T using N selected frames with a stride k. For example, given $\hat{\mathbf{u}} \in \{u^{T-k \times N}, u^{T-k \times (N-1)}, ..., u^T\}$, we estimate the average of those frames to represent the actual motion sequence as a single frame. Afterwards, we predict the target of human status using g from $avg(\hat{\mathbf{u}})$.

3. EXPERIMENTS

3.1. Experimental Settings

Image blending: For the image blending, we utilize two datasets: *HDD* [2] and *TikTok dancing* [3] from Kaggle. *HDD* consists of 236 different scenes without humans. For this, we manually select ten background images in which no object is placed in the center of it. *TikTok dancing* consists of 2,615 person images with corresponding binary masks. For this, all images are utilized to generate a training set. By blending images from both datasets with random image transformation, we obtain a tremendous amount of images for training.

Models: To validate our method on the various kinds of single and low-cost CNNs, we used four ImageNet pre-trained models: ResNet-18 [15], MnasNet [16], MobileNetV3 [17], EfficientNet18-B0 [14]. For all models, we modify the last fully connected layers into the proper dimensions.

Training and evaluation: At the training step, we only use the synthetic data generated by the proposed method. For the evaluation, we use two different benchmark datasets: *URFD* [13] and *AIHub airport*[1]. *URFD* consists of 30 falls and 40 adl (activity of daily living) clips. All of them are conducted as the test set for the video-level evaluation. To this end, if each video contains a frame predicted as falling by not less than one frame, the video is considered to be the fall. Consecutively, if a single frame in the video has never been predicted as falling, it is considered as non-falling. Experimentally, the interval for the detection was set to 1 second. *AIHub* consists of 10k clips with three different recording environments. Among those, we utilize fall and normal videos from *cubox* anomaly scene, which consist of 348 fall clips and 2,109 normal clips with longer duration than *URFD i.e.*, 10 seconds or longer with a higher frame rate as 30. For this data, the interval for the detection is set to 5 seconds, empirically. Since the two datasets were collected in different acquisition environments, we utilize two different image blending settings, *i.e.*, scale and transition.

Implementation: We utilize OpenCV and albumentation [18] for the image blending. Fall detection models are implemented using PyTorch [19]. All models are trained about 100 epochs using an AdamW [20] optimizer with a learning rate of 0.001 on a single GPU (RTX 3080).

3.2. Results

Evaluation on *URFD*: Table 1 shows the fall detection performance of our model compared to existing methods [2, 3, 4, 5, 6, 7, 8, 9, 10]. The top five rows are the results of multiple-stage-based approaches, and the other rows are the results of single CNN-based methods. The proposed method obtains comparable performance to the state-of-the-art methods

(a) Falling   (b) Non-falling

Fig. 3: Examples of the mean frame. Row 1-2 and 3 denote examples of *URFD* and *AIHub* dataset, respectively. Examples of the last column are the failure cases.

Table 2: Performance of prediction intervals on *URFD*.

Interval		Accuracy
Second	Frames	
0.4	10	90.0
0.6	15	95.71
0.8	20	95.71
1.0	25	**97.14**
1.2	30	91.43
1.4	35	91.43
1.6	40	85.71

Table 3: The computational speed of CNNs on *URFD*.

Model	Frames per second		Accuracy
	Pytorch	ONNX	
ResNet-18 [15]	**707.01**	**1092.80**	94.26
MnasNet [16]	472.22	1017.88	92.86
MobileNetV3 [17]	320.34	722.93	94.29
EfficientNet-B0 [14]	162.46	204.04	**97.14**

using multiple modules, in terms of accuracy. Besides, our method achieves the best performance among single CNN-based methods using EfficientNet-B0 [14]. Table 2 shows the results of the ablation study on the prediction intervals for calculating mean frames. Among the intervals, we achieved the best result in one second. We also compare the computational speed and accuracy of the conducted models as shown in Table 3. Overall, our method has a slight performance degradation although adopting a model for high speed.

Evaluation on *AIHub*: We also evaluate on *AIHub* dataset, as shown in Table 4. The results are estimated by frame-level since the falling frames are partial ($< 50\%$) of whole video

Table 4: Fall Detection results on *AIHub*.

	Frame-level Accuracy
Sensitivity	71.15
Specificity	95.76
Precision	73.59
Accuracy	92.25

clips. For this dataset, our method achieves an accuracy of 92.25% for frame-level detection.

Qualitative results: Figure 3 shows the results on *URFD* and *AIHub*. As shown, the falling state is well detected for samples similar to the generated synthetic data. Besides, ours also correctly predict the samples of multiple persons. While, ours rarely fails of fall detection in a few samples of a small person or unknown motion, which are not in the training data.

4. CONCLUSION

In this paper, we introduce the new simple fall detection framework. We propose the simple image blending method to generate human motional and motionless images, representing motion information in a single image. Then, we train the single CNN with the synthetic training data. At the inference step, we also represent the human motion in the single frame by estimating the mean image of the selected sequences and using it as input of the detection model. Through the extensive evaluation on *URFD* and *AIHub* datasets, we show the proposed method achieves the best performance among baselines. While, the handled datasets are containing few humans in one video. We plan to collect hard samples and expand our research based on localization and object detection tasks to analyze more diverse cases.

5. REFERENCES

[1] Fact Sheets (WHO), "Falls," `https://www.who.int/news-room/fact-sheets/detail/falls`, 26 April, 2021.

[2] Ziyi Guan, Shuwei Li, Yuan Cheng, Changhai Man, Wei Mao, Ngai Wong, and Hao Yu, "A video-based fall detection network by spatio-temporal joint-point model on edge devices," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 422–427.

[3] Chuan-Bi Lin, Ziqian Dong, Wei-Kai Kuan, and Yung-Fa Huang, "A framework for fall detection based on openpose skeleton and lstm/gru models," *Applied Sciences*, vol. 11, no. 1, pp. 329, 2021.

[4] Yong Chen, Weitong Li, Lu Wang, Jiajia Hu, and Mingbin Ye, "Vision-based fall event detection in complex background using attention guided bi-directional lstm," *IEEE Access*, vol. 8, pp. 161337–161348, 2020.

[5] Qi Feng, Chenqiang Gao, Lan Wang, Yue Zhao, Tiecheng Song, and Qiang Li, "Spatio-temporal fall event detection in complex scenes using attention guided lstm," *Pattern Recognition Letters*, vol. 130, pp. 242–249, 2020.

[6] Bogdan Kwolek and Michal 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.

[7] Xi Cai, Xinyue Liu, Suyuan Li, and Guang Han, "Fall detection based on colorization coded mhi combining with convolutional neural network," in *2019 IEEE 19th International Conference on Communication Technology (ICCT)*. IEEE, 2019, pp. 1694–1698.

[8] Wei Liu, Jie Guo, Zheng Huang, and Weidong Qiu, "Falling-action analysis algorithm based on convolutional neural network," in *Proceedings of the International Conference on Communication and Electronic Information Engineering (CEIE)*. 2016, Atlantis Press.

[9] Adrián Núñez-Marcos, Gorka Azkune, and Ignacio Arganda-Carreras, "Vision-based fall detection with convolutional neural networks," *Wireless communications and mobile computing*, vol. 2017, 2017.

[10] Carlos Menacho and Jhon Ordoñez, "Fall detection based on cnn models implemented on a mobile robot," in *2020 17th International Conference on Ubiquitous Robots (UR)*. IEEE, 2020, pp. 284–289.

[11] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister, "Cutpaste: Self-supervised learning for anomaly detection and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 9664–9674.

[12] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 2918–2928.

[13] Bogdan Kwolek and Michal 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.

[14] Mingxing Tan and Quoc Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[17] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al., "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[18] Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, pp. 125, 2020.

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[20] Ilya Loshchilov and Frank Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.