Eliminating Environmental Context for Fall Detection based on Movement Traces

*Balamanikandan J.[1][0000-0003-1200-6496], Senthil Kumar Thangavel.[2][0000-0001-8160-7223], Maiga Chang[3][0000-0002-2827-6223]

[1]Amrita Vishwa Vidyapeetham, Tamil Nadu 641112, India
[1*]cb.en.p2aid20017@students.amrita.edu

[2]Amrita Vishwa Vidyapeetham, Tamil Nadu 641112, India
[2]t_senthilkumar@cb.amrita.edu

[3]Athabasca University, Alberta T5J-3S8, Canada
[3]maiga.chang@gmail.com

Abstract. Falls are a prominent cause of mortality and severe injury among the elderly. It can be prevented by tracking them and providing prompt treatment. Current fall detection systems use data from sensors or cameras in various ways. False positives are common in sensor-based systems, and operating system constraints make privacy a major concern in vision-based systems. This paper proposes a technique for detecting falls from RGB images using a convolution neural network (CNN) utilising movement trace characteristics generated by a modified structural similarity index (SSIM) that can be integrated into resource-constrained devices for in-house monitoring. The proposed approach uses a camera system and is tested against the UR-Fall detection (URFD) dataset, outperforming previous fall detection systems. Our method achieves 99% accuracy. The model's dependence on readily available sensors and superior performance on the URFD dataset makes it a viable option for reliable fall detection in the real world.

Keywords: Fall detection, healthcare, computer vision, pattern recognition, movement tracking, convolutional neural networks, environmental context.

1 Introduction

In the current society, young people are travelling throughout the country in quest of better employment opportunities. According to Report [13], nearly 14.7 million or 28% of elderly Americans live alone, up from 6% a century ago. This includes 21% of older men and 34% of older women. Every year, between 28 and 35% of the elderly fall and these statistics worsen with age. Elderly people living alone are more likely to be destitute and prone to fall. Furthermore, a decline in eyesight, a lack of attention, and myasthenia gravis are all factors that can contribute to falls.

Until recently, most elderly people were supervised by humans, which resulted in a slew of management issues and personnel expenditures. Building an intelligent elderly monitoring system that can automatically identify and alert caregivers is critical for post-fall survival. The medical consequences of a fall have been proven to be highly reliant on response time. Fall detection systems may assist medical workers react faster and alleviate the medical consequences of falls. Elders who fall are often unable to recover on their own, results in unconsciousness and increased chance of mortality. Thus, human fall detection has important theoretical and sociological research relevance.

Identifying and comprehending human behaviour continues to be a challenging and important undertaking. Falling is frequently regarded as an abnormal human activity distinct from other activities of daily living (ADL). A typical fall detection system should distinguish between falls and ADLs. There are several fall patterns available, each having its own motion posture and making it difficult to mimic the momentum of the fall. As a result, relatively few real-life falls have been recorded. With fewer data, it's challenging to create a model that can detect falls in all scenarios. To work on any device and in any environment, a new architecture must be devised that is both lighter and more precise. This paper proposes an approach that can quickly distinguish between different activities with computer resources.

2 Literature Survey

The past several years have seen a proliferation of research on various methods for fall detection systems being conducted. Several studies [1-3] examined these methods in detail and categorized those methods into wearable device-based systems and context-aware systems based on data collection equipment used. Wearable device-based solutions need the individual to wear sensors such as a tilt switch, an accelerometer or a gyro meter [2]. While context-aware systems often make use of piezoelectric sensors, acoustic sensors, infrared or RGBD cameras [3]. Video-based analysis has acquired significant appeal among context-aware systems as a result of the rapid development of computer vision architectures [1].

The research [4] describes the usage of an accelerometer sensor to detect falls. This was done by computing the subject's SVMA and comparing it to a specified threshold. They determine the heart rate and trunk angle if the value is above the threshold and achieved an accuracy of 97.5%. Researchers designed a wearable airbag device for the avoidance of fall injuries in [5]. They activated the airbag using both acceleration and angular velocity inputs using a thresholding mechanism. Although, the primary benefit of prediction through these devices is that it is inexpensive and does not jeopardize the user's privacy. The suggested

approaches generate high number of false positives.

The wearable sensors require rather precise sensor alignment, it is inconvenient for users, particularly seniors who often forget to wear them [6]. An image classification pipeline is most often used to analyze video data for identifying falls [7] since it is more cost-effective and has been widely used in the monitoring of most buildings in the world. In [6], the researchers suggested detecting falls by evaluating human shape deformation during a video sequence. A shape matching algorithm is utilized to monitor the person's silhouette throughout the video sequence, and then the Gaussian Mixture Model is employed for classification.

The authors of [8] intermediate feature was extracted from RGB video data using Hourglass Convolutional Auto-Encoder (HCAE), Hourglass residual units (HRU). Those characteristics were utilized to categorize the action and rebuild the frame using the HCAE decoder to increase the representation of the intermediate features. They evaluated their approach using the UR fall dataset and obtained an accuracy of 96.2%.

In [9], The authors suggested a Siamese network with one-shot classification, which learns to distinguish distinct video sequences with the use of a similarity score. Two architectures are used to classify RGB and optical flow features: one with 2D convolutional filters and another with depth filters. They obtained findings that were comparable to those obtained at the state of the art. In [10], The authors provided mathematical methods for detecting human falls using pretrained YOLO FCNN and Faster R-CNN architectures. They conducted a comparison of these models and concluded that their approach achieves 99.22% accuracy in the Fall Detection Dataset (FDD).

The authors of [11] developed a three stream CNN architecture for classification after preprocessing the surveillance footage. The first two streams of CNN receive silhouettes and motion history images from video, while the third stream receives dynamic images with constraints. It obtained 100% sensitivity and 99.9% specificity in the Multiple cameras fall dataset. This [12] is closely related to our work of integrating traditional algorithms with deep learning methodologies. They extract the skeletal information of important joints related to fall activity from depth maps. These features are inputted to 1D convolutional neural network (CNN) to classify the action. It achieves an accuracy of 99.2% in NTU-RGBD dataset.

Most of the prior research in this field has relied on hand-crafted features, which has its own set of issues, notably in terms of generating fair and consistent feature sets for tasks that demand a comprehensive understanding of their respective domains. The initial challenge when developing vision-based systems is

to identify the motion component of persons in a scene. Numerous authors created their own frameworks for motion detection, while others relied on predefined motion extraction approaches. Most deep learning algorithms are computationally expensive and need a powerful GPU to process. The detection in this study is accomplished via the combination of algorithms and neural network architecture. This has resulted in increased processing speed for devices with limited resources.

3 Algorithm

In this paper, an algorithm-based approach is employed for the identification of falls in the environment. This section will discuss the algorithms being utilized and the various methodologies used in order to accurately identify activity in real time.

3.1 Structural Similarity

It is observed that natural image signals are highly structured. The successive frames of the video demonstrate significant interdependence between them. These dependencies include crucial information about the structural features of the visual scene. By comparing the two subsequent frames, the structural variations between them can be identified. These pixel-by-pixel variations reveal the information on the movement.

Structural Similarity Index (SSIM) [14] in general helps in measuring the similarities and differences in each image. Suppose x and y are two subsequent frame images, the purpose of the system is to provide the similarity differences between them. To measure the differences, the SSIM combines three components of the image such as luminance, contrast and correlation term. Given an image, Luminance can be calculated by averaging the intensities of all the pixels using,

$$\mu_x = \frac{1}{N} \sum_{i=1}^{N} x_i \quad (1)$$

Then, the luminance comparison function is written as,

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1'} \quad (2)$$

where μ_x and μ_y corresponds to the luminance of x and y signals [14]. The mean intensity is removed from the signal,

$$\sum_{i=1}^{N} x_i = 0 \quad (3)$$

The Contrast [14] can be compared using the formula,

$$c(x,y) = \frac{2\sigma_x\sigma_y+C_2}{\sigma_x^2+\sigma_y^2+C_2} \quad (4)$$

After that, the image signal is normalized by its own standard deviation to have a unit standard deviation. The structural comparison is conducted on the normalized signals and can be represented as [14],

$$s(x,y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3} \quad (5)$$

Where C_1, C_2, C_3 are constants to assure numeric stability. The above values are computed locally to a particular location rather than globally. This nature of the method is adapted in this research where the comparison of local region pixels between successive images provides us the local change in pixel regions. They are represented in a binary image of the same size as the original frame with altered pixels have value of 1 and other pixels with a value of 0. These pixel-level changes are localized in the frame by bounding boxes using the Contour approximation approach, which is detailed in the next part.

3.2 Contour Tracing Method

The contour tracing approach [15] is a segmentation technique used to determine the border pixels of a digital binary image. This approach is relatively straightforward in defining the hierarchical connections between the boundaries and efficiently distinguishes the outer borders attributes from the hole borders. It is applied to the binary image generated using the frame differencing method, it begins by scanning the image from the top left region in a row-by-row way in search of the boundary beginning point. After determining the contour start point, it performs pixel neighborhood search sequence in 8-connectivity and boundary tracing following certain criteria to avoid recirculating the same boundary.

This method in [15] assigns a unique number to the border pixels in order to distinguish them from one another and names them NBD. The NBD starts from ‘two’ as the entire frame is already termed as ‘one’. It begins by assigning the NBD value to the border pixels clockwise. When the current boundary pixel P_n equals the second boundary pixel P_2 and the prior boundary pixel P_{n-1} equals the first boundary pixel P_0, the procedure terminates with the contour boundary being determined. This tracing approach eliminates single-pixel contours or noise and locates each contour's outside edge.

After segmenting the contour area, a bounding box is used to localize the contour in the intermediate frame. As this frame captures the motion component

that occurs between successive frames, rather than just detecting the complete person. We could directly extract the internal motion that occurs between the human body's bodily sections. To eliminate noise in the motion component's representation, we execute several morphological procedures such as Dilation and Erosion. The bounding box is fitted to the contour representing the motion component, and the center point of the bounding box is determined. These procedures are performed on successive frames, and the moving trace is constructed using the center points of the complete frameset. The trajectory image is constructed on a binary digital image with dimensions 224x224, and pixels set to zero.

3.3 Neural Network

The architectures of vanilla-CNN designed for the prediction is shown in. In the proposed architecture, the Vanilla-CNN typically includes stack of three convolutional, pooling, and activation layers with fully connected layers attached at the end for prediction. Convolutional layers operate on the input that condenses the number of pixels in the receptive field of a filter into a single value. A filter "slides" across the 2D input data in a conv2D layer, performing elementwise multiplication and condensing the information into a single output pixel. The kernel will repeat this procedure for each location it traverses, transforming a 2D feature matrix into an informational 2D feature matrix. The max pooling layer chooses the most prominent feature from the region of the feature map covered by the filter, eliminating all other features. Finally, Dense layers categorize the input based on the retrieved characteristics.

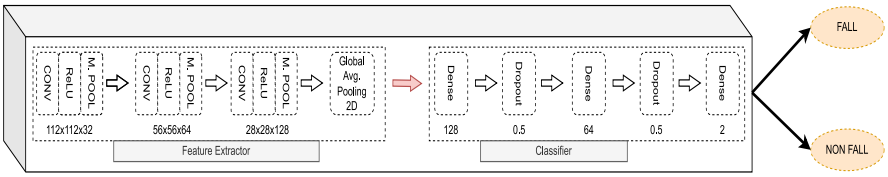

Fig. 1. Network architecture for classifation

The early phases of preprocessing and extracting the motion trace of the video clip produces the grayscale images with the size of 224x224x1. The first convolution layer consists of 32 filters with a filter size of 3 and padding to extract information and stack it along the depth dimension of the input. these processed feature maps are transferred to pooling layer, which lower their spatial dimension. The activation layer then generates nonlinearity between the feature maps. The input is processed successively by three sets of these layers, with the number of filters rising linearly in the convolutional layers, such as 64 and 128 in the second and third convolution layers. This process captures subtle information from the input and aids the Dense layers in categorization. There are two dense layers with 128,64 neurons each, and a final layer with two neurons for binary categorization.

4 Methodology

In sports science, it is common to use image-based human motion analysis to analyses subject's posture and activity. Each action has its own pattern, irrespective of the person performing it. For instance, walking has a distinct pattern from that of sitting. However, there is no dissimilarity between several walking individuals. These behavioral patterns are extracted and categorized and demonstrate that this method outperforms other algorithms that look at the whole image. This section will describe about removing the surroundings generalizability after extracting motion components, then constructing movement trace images, and finally classifying the activity in multiple sections. The Fig. 2 depicts the suggested framework's procedure.

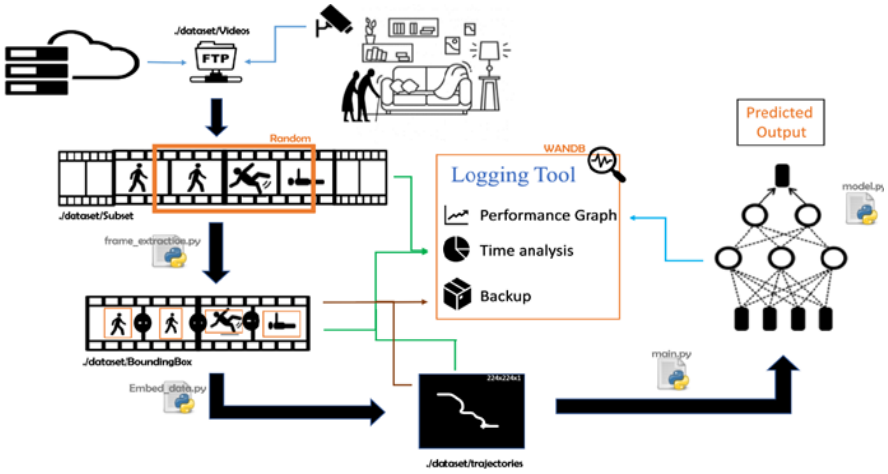

Fig. 2. The pipeline proposed in this research

4.1 Preprocessing Stage

Each URFD video is pipelined to separate the frames and are saved in a folder named after the video. The CSV file contains information on each video in the URFD dataset. The CSV file contains frame-specific information such as frames with falling, intermediate, and other positions. To enable frame preprocessing, the frames are renamed to match their folders. The pre-processing stage also ensures the cleaning of data for classification. Once frames are retrieved, they can be considered as sequentially stacked images and can be pre-processed as such. The frames are then normalised and standardised to improve the model's convergence.

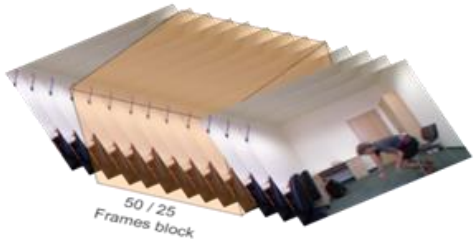

Fig. 3. Extraction and selection of random frames during preprocessing

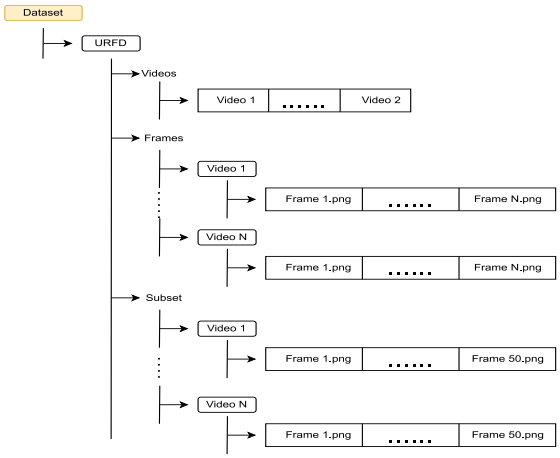

Fig. 4. Structure of the dataset organized in folders

Falling activity occurs in a matter of seconds. The frame rate of most security cameras is around 25 frames per second (FPS). Hence, the 50 frames are chosen at random from a different video each time as depicted in Fig. 3. The selection of fall videos is deliberate, as certain frames do not correspond to the individual falling. However, based on the information of subject's intermediate or falling frames from CSV file, the sampling is done to contain at least a single fall frame in the sample. The sampled frames are then placed in the dataset's subset folder along with the videos. Fig. 4 shows the dataset folder structure after preprocessing.

4.2 Frame Differencing

The optical flow approach is one of the most used methods for extracting motion components from video. The optical flow features retain an internal representation of motion in the video by computing displacement vectors between two successive frames. [16] Convolutional neural networks are used to extract and categorise optical flow data, with the CNN architecture adjusted to account for temporal aspects of optical properties in consecutive stacks of images. Here, the entire environment is evaluated, including the motion field, even if the motion

component is very small or non-existent in the frame.

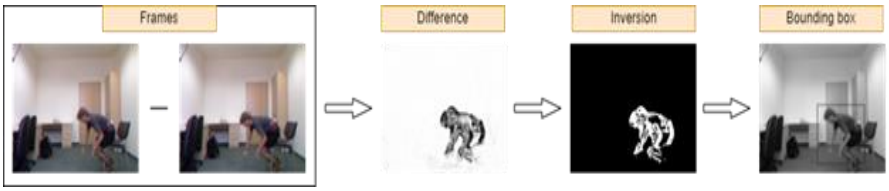

Fig. 5. Processing of frames using frame differencing method

The primary necessity is to detect movement between the regions of the human body that contain the information necessary to accomplish the activity. The trajectory of a person may be determined by analyzing the motion between frames. For the identification of motion components, a modified version of the structural similarity index technique is being used. It creates a local component by extracting the internal motion that occurs between bodily components. This method can localize the motion component in 0.05 seconds. Fig. 5 represents the process of subtracting two consecutive frames provides the image with pixel change, this image is inverted and passed through contour approximation method for localizing the pixel change with bounding box. These bounding box center points are used to create a trajectory image as depicted in Fig. 6 for classification.

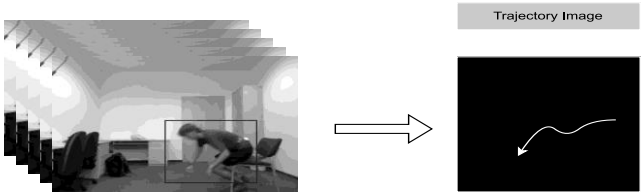

Fig. 6. Trajectory image prepared and passed for classification

4.3 Classification

The motion trace image is utilized to classify the action category using the vanilla-CNN architecture discussed previously. By using convolutional filters, the neural network retrieves different information from the image that is required to differentiate it from other images for prediction. This information includes, but is not limited to, corners, texture, patterns, blobs, and color. Convolutional filters containing additional information are layered in the image's depth dimension. The pooling layers are used to consolidate the characteristics learnt by the convolutional layer feature map and decrease the image's spatial dimension.

Using the backpropagation technique [18], the parameters of the network are modified during the training phase to map the input image and the target label. The network constructs a multidimensional space in which the input and output

target labels are represented. Thus, during inference, the input is projected to the same space based on its extracted features and a label closely comparable to the existing mappings is inferred. This architecture provides two class probabilities that indicate the input's most probable class.

5 Evaluation

The proposed methodologies are implemented in python using TensorFlow framework, and its performance is assessed in URFD dataset. The tests are conducted in CPU with Intel Core I5 processor and 16 Gb RAM.

5.1 Dataset

All these tests are performed using the UR fall detection dataset [17], which was originally proposed in 2014 by researchers from the University of Rzeszow's Centre for Computational Modelling. The dataset includes videos captured with two Kinect cameras at a frame rate of 31 frames per second and a resolution of 640x480. One camera is mounted on the top of the environment, while another is mounted on the front side to capture the subject. The videos are accessible in both RGB and Depth formats, along with both horizontal and vertical viewing angles.

The dataset contains 30 falls and 40 everyday activity sequences like bending, kneeling, and walking. Each video features an actor performing or simulating a single activity from any category. To protect the actors, mattresses were placed on the floor where the subject fall, and any material that could injure the subject was removed. The data on real falls is scarce and difficult to procure. Thus, actors are used to imitate the fall. These fall actions may not exactly reflect a person falling in real life, but they are the closest representations accessible at this moment. This dataset comprises only static items and events that occurred in confined environments during daylight. In absolute terms, there is no motion greater than that of a subject.

5.2 Evaluation Metrics

This is a binary classification problem to identify if a fall occurred within a sequence of frames. Recall, precision, and accuracy are widely used to assess such classification problems. These measures perform well when datasets are skewed, which makes them more suitable for our task of fall recognition since fall samples are often fewer than those from other activities. The evaluation is based on True Positives (TP) and True Negatives (TN). True Positives (TP) are the number of correctly categorised falls, whereas True Negatives (TN) are the number of correctly classified non-falls. False Positives (FP) indicate the number of non-falls that are incorrectly classified as falls, while False Negatives (FN) indicate the

number of falls that are incorrectly classified as non-falls.

Recall. A recall value is the fraction of correctly classified falls across the entire dataset. This can be formulated as,

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

Precision. Precision is measured as the fraction of correctly classified falls among all classified falls. This can be formulated as,

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

Accuracy. Accuracy is calculated as the ratio of correctly classified falls to correctly classified non-falls. This can be formulated as

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (8)$$

5.3 Results

The performance of the approach is validated with different hyperparameter values. The frame count hyperparameter in the pipeline is tuned to different values such as 25, 50 to assess the model's performance. Numerous methods are used to prepare and retrieve information for recognition before using the neural network in its entirety. The pipeline takes a cubical video data and returns a picture with trajectory of the subject. The neural network was trained for 15 epochs in all experiments. The data size was investigated with values 120, 180, and 240. The training is done incrementally, i.e., the model trained with 120 data points is retrained with a bigger number to determine model performance in unseen larger data. This technique helps us identify crucial characteristics for successful prediction and design of leaner classification architecture.

Table 1. Summary of results obtained in different settings

Setting	Loss	Accuracy	Precision	Recall
Size: 120 **Frames:** 50	**Train:** 0.028	**Train:** 100	**Train:** 97.8	**Train:** 95.8
	Val:0.023	**Val:** 100	**Val:** 100	**Val:** 100
	Test: 0.112	**Test:** 96.6	**Test:** 96.6	**Test:** 96.6
Size: 180 **Frames:** 50	**Train:** 0.042	**Train:** 99.3	**Train:** 98.6	**Train:** 98.6
	Val: 0.08	**Val:** 100	**Val:** 100	**Val:** 96
	Test: 0.073	**Test:** 96.1	**Test:** 96.1	**Test:** 96.1
Size: 240 **Frames:** 50	**Train:** 0.009	**Train:** 100	**Train:** 100	**Train:** 100
	Val: 0.25	**Val:**100	**Val:** 100	**Val:** 100
	Test: 0.041	**Test:** 99.5	**Test:** 99.5	**Test:** 99.1

Size: 120 **Frames:** 25	**Train:** 0.01 **Val:** 0.051 **Test:** 0.208	**Train:** 100 **Val:** 100 **Test:** 91.6	**Train:** 98.9 **Val:** 94.7 **Test:** 93.9	**Train:** 100 **Val:** 100 **Test:** 90.8
Size: 180 **Frames:** 25	**Train:** 0.002 **Val:** 0.002 **Test:** 0.135	**Train:** 100 **Val:** 100 **Test:** 97.2	**Train:** 100 **Val:** 100 **Test:** 97.2	**Train:** 99.3 **Val:** 100 **Test:** 97.2
Size: 240 **Frames:** 25	**Train:** 0.008 **Val:** 0.017 **Test:** 0.055	**Train:** 100 **Val:** 100 **Test:** 98.7	**Train:** 99.4 **Val:** 100 **Test:** 98.7	**Train:** 100 **Val:** 97.2 **Test:** 98.7

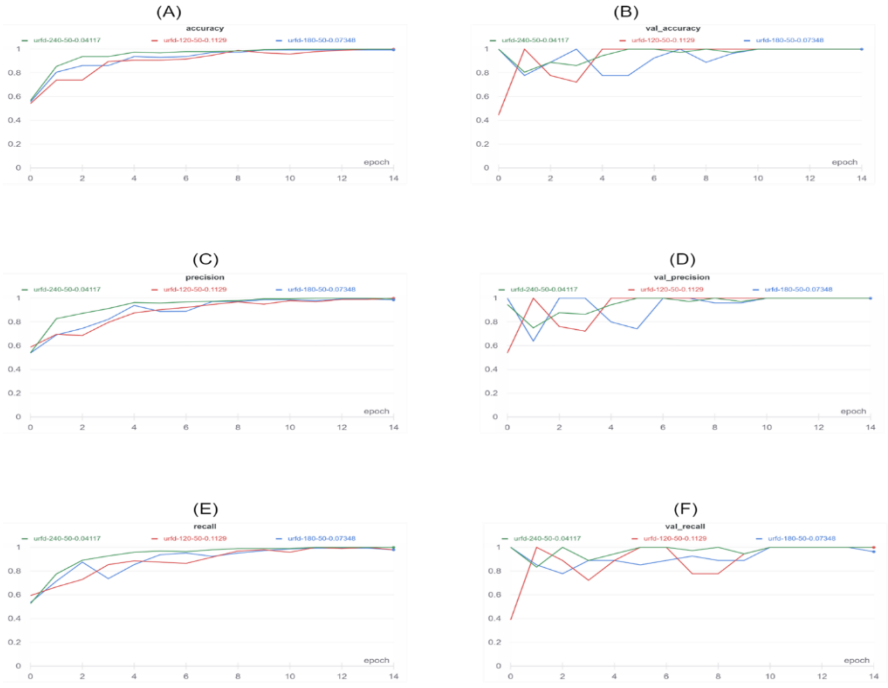

Fig. 7. Graphical results of evaluation of the approach with random selection of 50 frames

Following the data pipeline, experiments are conducted with several network architectures. It is noted that increasing the number of network layers and hidden nodes causes the network to overfit. As this is a binary classification problem, the model is trained using binary cross-entropy loss and stochastic gradient descent (SGD). Fig 6 depicts the results of several performance measures on the training and validation set for the 50-frame data processing approach. The performance of the approach in the training dataset is represented by Fig 7 (A, C, E), while the performance of the approach in the validation dataset is represented by Fig 7 (B,

D, F). Each graph shows the value of Epochs in the X-axis and its corresponding value in the Y-axis. As previously indicated, the experiments are carried out with varying numbers of data samples. The color lines in Fig. 6 Line graphs illustrate the various data sample settings, which are 240, 120, and 180.

Fig. 8. Time consumed for preprocessing frame blocks in different settings

Fig 6(A, B) demonstrates that the approach converges after ten training epochs and that accuracy achieves 100% for all data settings. Similar patterns can be noticed for Precision Fig. 7(C, D) and Recall Fig. 7(E, F). More precisely, it achieves 98.97% accuracy and 100% recall at 13th epoch when 120 data samples are selected from each video. In Fig. 6, blue line indicating model trained with 180 data samples performs better. The results in validation Fig 7(B, D, F) shows 99.31% accuracy, precision at the 13th epoch, and recall of 98.61%. Although the performance seems to be comparable to the model trained with 120 data samples, the loss on test-set reduced from 0.1120 to 0.0735, demonstrating the approach's enhanced generalization with more data.

The data sample size was raised by one-fold 240 to confirm the performance findings. Additionally, Fig 7 (B, D, F) demonstrates that the technique achieves 100% accuracy, precision, and recall and works well in unseen data samples with a loss value of 0.04117. This demonstrates that the amount of data samples is critical to performance. Although, further increasing the data size does not result in performance gains. As seen in Fig 8(A), the average time needed to analyze a data sample of 50 frames is around 2 seconds.

The technique is also validated using 25-frame selected for each data sample, as seen in Fig. 9. Like the previous graphs, which depict epochs on the X-axis and their associated values on the Y-axis. Fig. 9 (A, C, E) depicts the findings on the training dataset and seems to be comparable to what was obtained before with convergence after the 10th epoch. however, the Fig. 9 (B, D, E), demonstrates a significant amount of noise in the training results, implying an inadequate trace in each data sample. Fig. 8(B) demonstrates that the average time required to process each data sample is around one second for 25 frames. It is

noticed that the processing time for these data samples is greatly reduced, while performance has increased significantly as the number of data samples grows.

Fig. 9. Graphical results of evaluation of the approach with random selection of 25 frames

6 Conclusion

The primary insight obtained from the studies with various parameter settings was the effect of data size on action prediction. With fewer parameters, the suggested technique obtained more than 99% accuracy in the URFD dataset. The inference takes two seconds in total, including preparation time. This enables the proposed approach to operate in near real time and to generalize more effectively in the unseen data. This architecture can be retrained overnight in less than five minutes to include fresh data and improve recognition over time. In future, end-to-end neural network pipeline with fewer parameters will be designed to detect falls accurately in real-time with higher generalization capability.

References

1. Mubashir, M., Shao, L., & Seed, L.: A survey on fall detection: Principles and approaches. Neurocomputing, 100, 144-152 (2013)
2. Pannurat, N., Thiemjarus, S., & Nantajeewarawat, E.: Automatic fall monitoring: A

review. Sensors, 14(7), 12900-12936 (2014)
3. Stone, E. E., & Skubic, M.: Fall detection in homes of older adults using the Microsoft Kinect. IEEE journal of biomedical and health informatics, 19(1), 290-301 (2014)
4. Wang, J., Zhang, Z., Li, B., Lee, S., & Sherratt, R. S.: An enhanced fall detection system for elderly person monitoring using consumer home networks. IEEE transactions on consumer electronics, 60(1), 23-29 (2014)
5. Tamura, T., Yoshimura, T., Sekine, M., Uchida, M., & Tanaka, O.: A wearable airbag to prevent fall injuries. IEEE Transactions on Information Technology in Biomedicine, 13(6), 910-914 (2009)
6. Rougier, C., Meunier, J., St-Arnaud, A., & Rousseau, J.: Robust video surveillance for fall detection based on human shape deformation. IEEE Transactions on circuits and systems for video Technology, 21(5), 611-622 (2011)
7. Popoola, O. P., & Wang, K.: Video-based abnormal human behavior recognition—A review. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6), 865-878 (2012)
8. Cai, X., Li, S., Liu, X., & Han, G.: Vision-based fall detection with multi-task hourglass convolutional auto-encoder. IEEE Access, 8, 44493-44502 (2020)
9. Berlin, S. J., & John, M.: Vision based human fall detection with Siamese convolutional neural networks. Journal of Ambient Intelligence and Humanized Computing, 1-12 (2021)
10. Singh, K., Rajput, A., & Sharma, S.: Vision based patient fall detection using deep learning in smart hospitals. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 9(2) (2019)
11. Kong, Y., Huang, J., Huang, S., Wei, Z., & Wang, S.: Learning spatiotemporal representations for human fall detection in surveillance video. Journal of Visual Communication and Image Representation, 59, 215-230 (2019)
12. Tsai, T. H., & Hsu, C. W.: Implementation of fall detection system based on 3D skeleton for deep learning technique. IEEE Access, 7, 153049-153059 (2019)
13. World Health Organization. (n.d.). Dementia. World Health Organization, https://www.who.int/news-room/fact-sheets/detail/dementia
14. Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 13(4), 600-612 (2004)
15. Suzuki, S.: Topological structural analysis of digitized binary images by border following. Computer vision, graphics, and image processing, 30(1), 32-46 (1985)
16. Núñez-Marcos, A., Azkune, G., & Arganda-Carreras, I.: Vision-based fall detection with convolutional neural networks. Wireless communications and mobile computing, (2017)
17. Kwolek, B., & Kepski, M.: Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer methods and programs in biomedicine, 117(3), 489-501 (2014)
18. Rumelhart, D. E., Hinton, G. E., & Williams, R. J.: Learning representations by back-propagating errors. nature, 323(6088), 533-536 (1986)