ISSN 2409-2665
Journal of Logistics, Informatics and Service Science
Vol. 9 (2022) No.4, pp.51-71
DOI:10.33168/LISS.2022.0405

Vision-based In-room Fall Detection Application

Yi-Heng Cho , Mat-Desa Shahbe

Faculty of Computing and Informatics, Multimedia University, Malaysia

Abstract. In recent years, fall detection is a hot study topic as video surveillance becomes more universal in public and private spaces. The consequence of falls not only can result in physical damage but as well as psychological issues, especially among the elderly. With the help of the great processing power of a computer, deep learning has come into help to compete with traditional fall detection that required hand-crafted features. This paper presents a fall detection application using a convolutional neural network (CNN) for web-based and mobile-based implementations. A deep network can learn to detect a fall automatically by showing it with an ample amount of examples. A deep learning network is built to detect falls in an image with a CNN structure. A technical representation of the proposed design and methodology is organized and presented in this paper. As a result, a fall detection web application was built with the integration of a fall detection model with 98.37% specificity and 96.47% sensitivity.

Keywords: Fall detection, machine learning, CNN, mobile application, web application

1. Introduction

In recent years, falls are said to cause about 37.3 million serious injuries and 646,000 deaths per year, making them a global public health hazard (Santos, et al., 2019). Falls can result in a wide range of outcomes, from no injury or slight injury to major injury or death. For example, pain, bruising, scratches, and bleeding are the common impacts of the fall. A fall has significant psychological implications since it significantly affects the self-confidence and independence of those who are afflicted. This could lead to more serious falls in the future, or it could lead to a loss in health. Other common repercussions include early nursing home admission and a constant dread of falling, both of which reduce the quality of life (Cumming, Salkeld, Thomas, & Szonyi, 2000).

Various approaches based on modern equipment have been proposed in recent years for the detection of fall incidents. Because wearable devices rely on sensors attached to the user's body, such as tilt sensors, accelerometers, gyroscopes, interface pressure sensors, and magnetometers, they have been widely used in past studies to capture and analyze body movement. These approaches managed to achieve decent performance in fall detection for the elderly but there are still some limitations. For example, the sensor is considered invasive as the elderly must wear or attach the sensor to their body parts for the sensor to function. The elderly who have mobility issues might find it troublesome and uncomfortable to wear the sensors that added weight to their body. To address this shortcoming of the wearable sensor, the camera-based fall detection method is short-listed to be the most suitable approach.

The expected output of this project is an application that can detect fall events in a live video sequence with the use of machine learning techniques. The application also provides some settings and adjustments for the user to turn on or off the application notification and receive an email notification when a fall is reported.

2. Related Work

2.1. Background Subtraction Methods

Generally, the human body should be extracted from the background for further feature extraction including the human position, shape, and motion. In the study (Lin, Wang, Hong, Kang, & Huang, 2016), a Gaussian mixture model (GMM) is used to build or establish a background model for the segmentation of foreground objects.

Optical flow images are used by (Núñez-Marcos, Azkune, & Arganda-Carreras, 2017) as the input of their neural network. Extracted optical flow images represent only the motion of the video frames while ignoring appearance-related information.

In the work done by (Harrou, Zerrouki, Sun, & Houacine, 2017), the body silhouette is extracted from the video by their segmentation module. Background subtraction is used to extract the human silhouette by computing the difference between the background template and the input image. Similar to the research of

(Harrou, Zerrouki, Sun, & Houacine, 2019) , the author uses the background subtraction method and morphological processing to produce a cleaner silhouette.

For the image processing stage in the research by (Panahi & Ghodsb, 2018), frames are repaired with the application of a median filter and the detection of the invalid pixels where zeros values will be substituted with the nearest valid pixel. After that, the first frame subtraction is applied to eliminate the background, followed by morphological operations and contour extraction to remove the noise.

(Chen, Li, Wang, Hu, & Ye, 2020) had suggested the use of Mask R-CNN for the background subtraction. Mask R-CNN is a deep learning method extents from the Fast R-CNN (Girshick, 2015) with an additional parallel processing branch to predict the object mask, and substitute the original ROI Align with ROI pooling.

2.2. Fall Classification Methods

After going through the necessary background subtraction process, the obtained silhouette can be used for further feature extraction to obtain the important attributes, which will become the subsequence input of the classification algorithm.

In the research by (Harrou, Zerrouki, Sun, & Houacine, 2017), a multivariate exponentially weighted moving average (MEWMA) table and support vector machine (SVM) are utilized to detect falls from the camera. The MEWMA chart is first trained with the fall-free data, as its ability to detect small shifts, it can identify fall and fall-like motion from daily activity. To further distinguish between real fall and other fall-like actions for example lying down, an SVM is used to determine the best separation hyperplane.

(Harrou, Zerrouki, Sun, & Houacine, 2019) uses a generalized likelihood ratio (GLR) detector to detect falls in a video feed. The main goal of the GLR test is to distinguish between two hypotheses H0 and H1 based on observed data by comparing the decision statistic to the control limits. So, fall-free data will be used to train or retrieve the threshold value of the GLR. When the GLR decision rule exceeds the threshold calculated in the training phase, a fall event is triggered.

Other than that, (Panahi & Ghodsb, 2018) had proposed using the threshold method where a lying flag (LF) is set at a specific frame if the distance between the center of an individual's silhouette and the ground was less than a predetermined threshold by fitting ellipse and rectangle to the silhouette. To distinguish between falls and normal activities, (Lin, Wang, Hong, Kang, & Huang, 2016) calculated the acceleration and angular acceleration from the ellipse. The motion history image (MHI) is used by (Lin, Wang, Hong, Kang, & Huang, 2016) to detect fall events in a real-time situation. MHI is a series of continuous images within a period where the most recent movement has the highest intensity, which then steadily decreases as time passes. Activities with large motion and high acceleration and angular acceleration are classified as fall events.

(Panahi & Ghodsb, 2018) performed mathematical calculations to detect the fall with features from the fitted ellipse such as the inclination angle, the aspect ratio of the fitted rectangle to the ellipse, and the distance between the ellipse's centroid and the ground. These features are fed into SVM for fall classification.

With the optical flow images generated by (Núñez-Marcos, Azkune, & Arganda-Carreras, 2017), the author has chosen the VGG-16 architecture as their convolution neural network (CNN). The author modified the input layer of the model to receive a stack of optical flow images. Transfer learning is applied to the model with the following training steps. The neural network is first trained on the Imagenet dataset (Deng, et al., 2009) to perform generic image recognition, after that the network is trained on UCF101 (Soomro, Zamir, & Shah, 2012) for human action classification and finally, the network is frozen and fine-tuned for detecting fall events.

Multi-task hourglass convolutional auto-encoder (HCAE) is used by (Cai, Li, Liu, & Han, 2020) to avoid the loss of information during a multilayer deep network. The original video frames are fed into the HCAE in the first phase, and the encoder of the HCAE with the hourglass unit (HRU) is used to create an intermediate feature with more behavior information. The intermediate feature is used in a multi-task mechanism in the second phase, in which fall detection is completed as the main task by classifying the intermediate feature using a classifier, and frame reconstruction is realized as a secondary task by the decoder to improve the representativeness of the intermediate feature.

2.3. Evaluation Matrix

In the detection of falls, a True Positive (TP) and True Negative (TN) means that a real fall is detected as a fall, and a non-fall is detected as a non-fall. False Positive (FP), or type I error is a false alarm where a non-fall is detected as a fall, and False Negative (FN), or type II error is a miss detection where a fall is detected as a non-fall. The confusion matrix is described in Fig. 1.

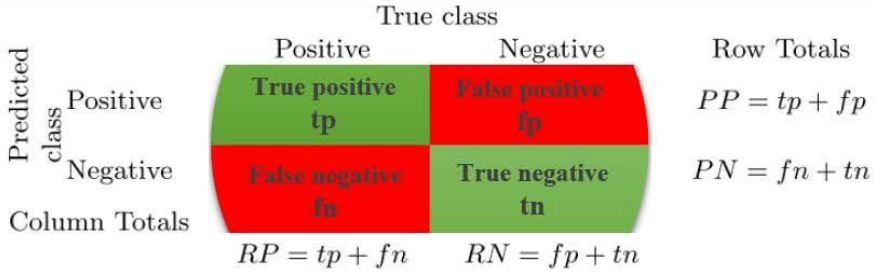

Fig. 1: Confusion matrix (Harrou, Zerrouki, Sun, & Houacine, 2019)

As the performance metrics, the proportion of accurately identified falls in all falls is called sensitivity or true positive rate (TPR), the proportion of successfully

identified falls in all detected falls is known as precision or positive predictive value (PPV), the percentage of accurately detected non-fall in all non-fall events is known as specificity or true negative rate (TNR), the fraction of accurately identified falls and non-fall activities is known as accuracy (ACC), and the harmonic mean of recall and precision is the F-score or F-measure. The performance metrics are described in Fig. 2.

$$TPR = \frac{tp}{RP} \qquad FPR = \frac{fp}{RN}$$

$$Accuracy = \frac{tp+tn}{RP+RN} \qquad Precision = \frac{tp}{PP}$$

$$F_1 Score = 2 * \frac{Precision * TPR}{Precision + TPR}$$

Fig. 2: Performance metrics (Harrou, Zerrouki, Sun, & Houacine, 2019)

3. Implementation Details

3.1. System Overview

3.1.1. Software Architecture

The software architecture is inspired by https://github.com/IBM/tfjs-web-app. The CNN model is built using two popular machine learning packages which are TensorFlow and Keras. The training and testing of the model are done in a Python environment. The proposed software architecture is illustrated in
Fig. 3.

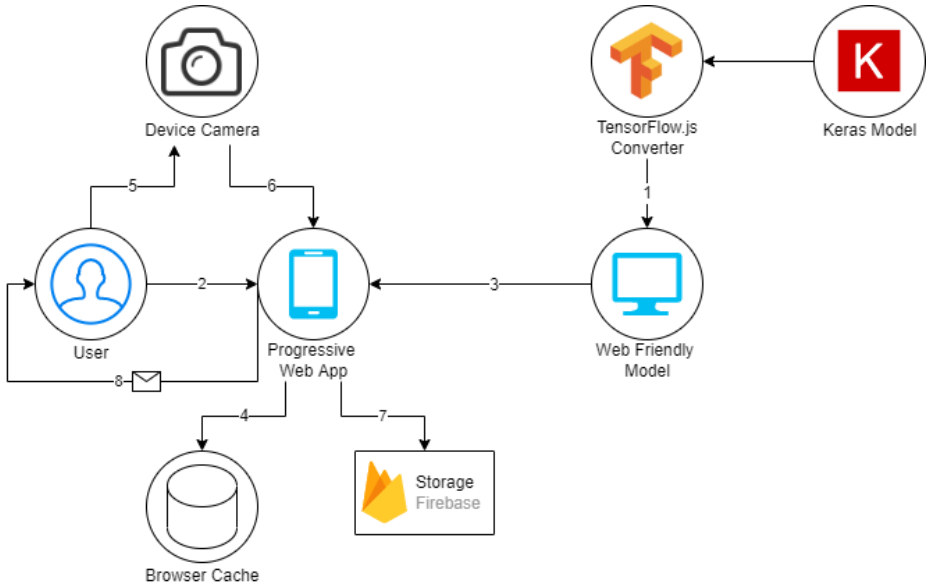

Fig. 3: Proposed software architecture

After the model is built and trained, the model will be converted to a web-friendly model (JSON) for the model to be executed in a web environment (process 1). After this, the JSON should be hosted online at a secure and highly available place so the front-end framework can retrieve the model easily.

As for the front-end part of the application, React is selected to build the user interfaces of the website. When the user launches the web application (process 2), the application assets and the model files are loaded from the web (process 3). The loaded assets and model are stored locally in the browser cache (process 4). The photo is taken from the video sequence (process 5) and fed into the model for fall prediction (process 6). When a fall is detected for a signed-in user, the current frame will be captured and uploaded to Firebase storage (process 7). An email will be sent to the user with the link to the image attached (process 8).

3.1.2. UML Sequence Diagrams

The proposed application is a simple fall detection application where limited configurations and settings are provided. The target user for the application is the family members of the person who is being monitored. The sequence diagram of the system setting is given in Fig. 4.

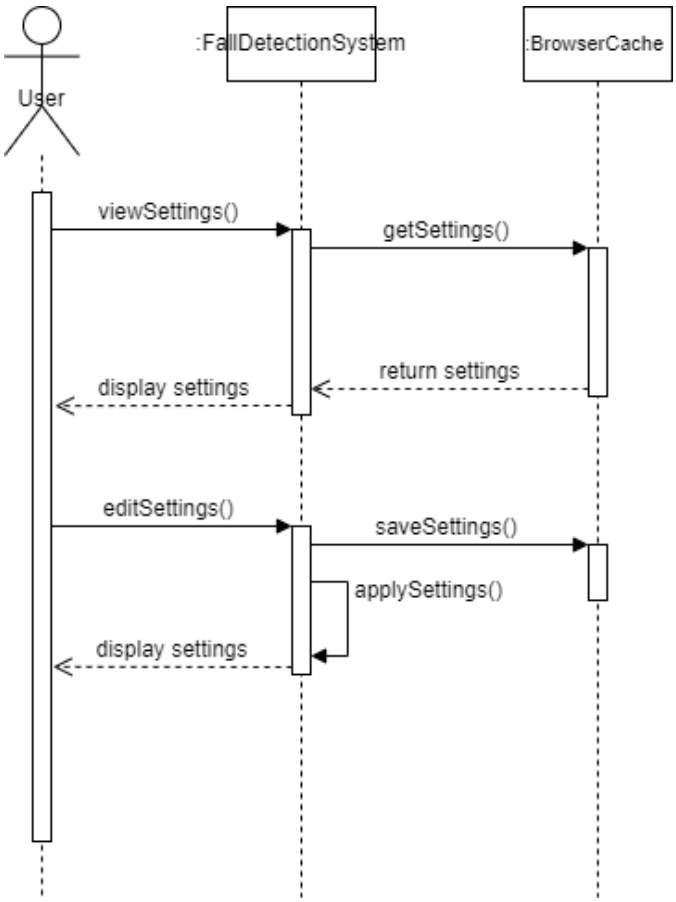

Fig. 4: Settings sequence diagram

Considering the main function of the application is to detect fall events from the camera feed, the ideal use case scenario is the webcam or smartphone is placed at a fixed position after the settings are confirmed. Hence, only a few parameters are provided for the user preference such as turning on or off the application's notification and signing in to receive email notifications when a fall occurs.

When the user selects to view the current settings, the system will request the current settings stored in the browser and display them on the user interface. When the user changed certain configurations, the system will then save the new settings in the browser and apply them. Finally, the latest configurations will be updated on the user interface.

Fig. 5 illustrates the fall detection sequence diagram. The application will retrieve the model and user settings from the browser and load them for this session. Then, the smartphone camera or webcam will send the video sequence to the application which will be interpreted as a series of frames or images. The image will be input into the loaded model and get the prediction result of either fall or no fall. Based on the prediction result, a fall will invoke an alternate flow where the current image will be uploaded to Firebase storage and returns an image URL, and send the push notification or email based on the user settings.

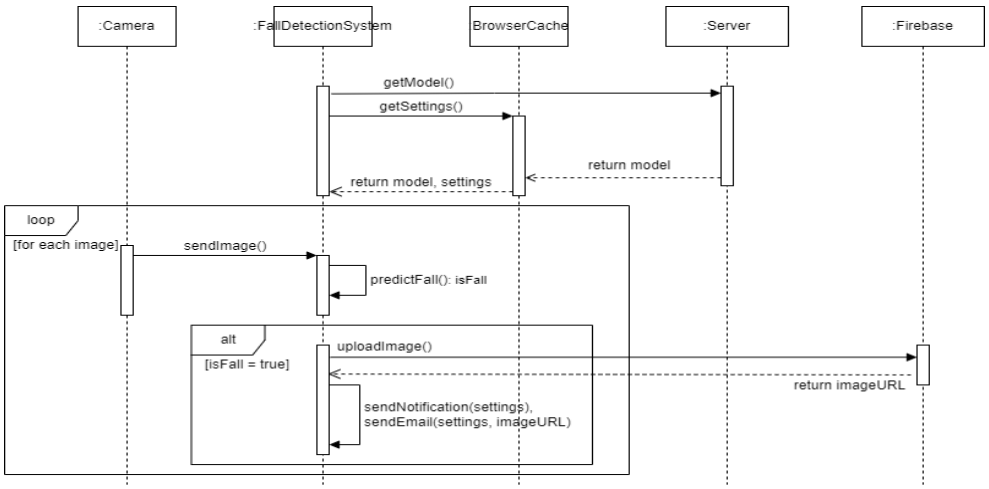

Fig. 5: Fall detection sequence diagram

3.2. Fall Detection

3.2.1. Dataset Used

There are limited public datasets available for the fall detection task and the URFD (Kwolek & Kepski, 2014) is selected as the dataset used for training and testing of the model. The authors provide an extra CSV file that contains the extracted features for each video sequence. The extracted features provide the labels of every frame in all videos that describes the human posture. The dataset provides 70 video sequences which consist of 30 fall events and 40 activities of daily living (ADL). Each sequence is captured from front-facing and ceiling view angles and RGB and depth images from the Microsoft Kinect camera are provided for each frame. Fig. 6 illustrates the 4 images of a frame extracted from a fall video sequence.

With the provided dataset, only RGB images from the front-facing camera are used to train the model. This is because the idea use case of the application is to place the smartphone or a computer webcam in a room pointed to the main activity space. Other than that, most rooms in Malaysia are installed with a ceiling fan, which could affect the position of the camera. Depth images are neglected as most of the camera lenses of a smartphone and webcam are not featured to capture depth images.

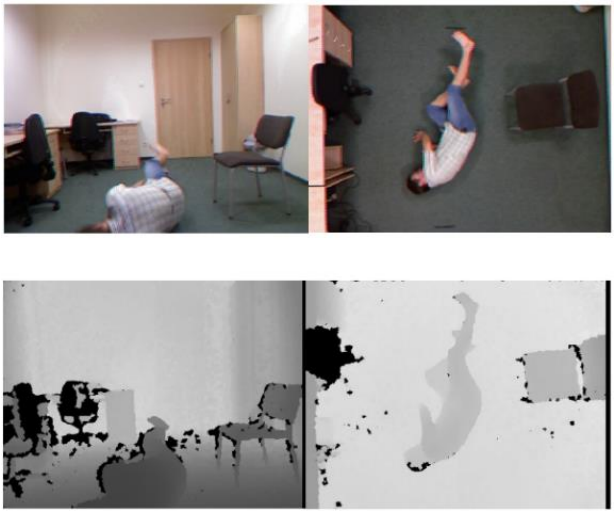

Fig. 6: Sample fall frame from URFD

3.2.2. Data Preprocessing

The labels provided in the CSV file have three values, "-1" is for not lying, "1" for lying, and "0" is the temporary pose during a fall (from not lying to lying). Since the designed deep learning network only supports binary classification, the not lying value (-1) is replaced with a value 0 so the network understands the fall as 1, no fall as 0. The dataset is split into two subsets, one for training and one for testing with a ratio of 8:2. The image distribution for training and testing is given in Table 1.

Table 1: Dataset distribution

Dataset	Total image	Fall image	No fall image
URFD	2995	903	2092
Training set	2396	722	1674
Testing set	599	181	418

3.2.3. FallNet Architecture

The brain of the fall detection application is the model or deep learning network that can predict a fall event in a given image. The FallNet proposed by (Reddy & Geetha, 2010) is used in this application as the backbone structure of the deep learning

network (Fig. 7). The author proposed a CNN that takes an input image and outputs a probability array with 6 classes of different falls which are abnormal fall, back fall, fall from the chair, front fall, normal fall, and side fall. Since the proposed application does not need to differentiate the type of falls, the network is modified to output a binary decision of fall or no fall.

This network is chosen for its simplicity and ideal performance metrics with 95.7% of precision and 97.6% of F1-score. Also, the image preprocessing process such as human background segmentation is not considered and applied in the application for real-time fall detection. In this work, the output layer of FallNet is modified to produce a single class to indicate a fall or not in the image, not different types of falls.

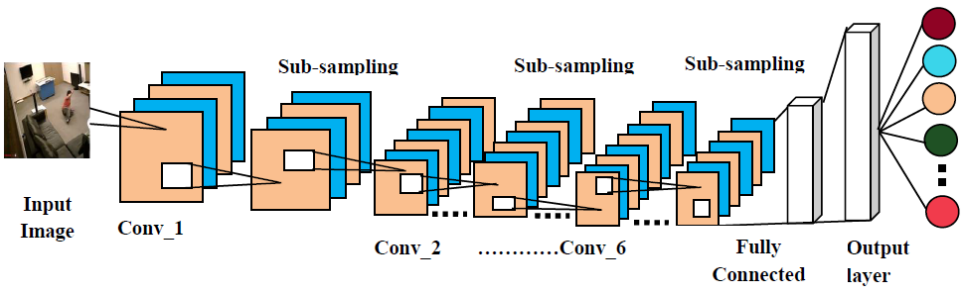

Fig. 7: Architecture of FallNet (Reddy & Geetha, 2010)

3.2.4. Model Training

The total number of images uses to perform model training is 2995. This number is then distributed with a ratio of 80% for training and 20% for testing. In the end, the size of the stacked training and testing images are (2396, 480, 640, 3) and (599, 480, 640, 3) respectively where 480 and 640 are the height and width of the image, and 3 represent the 3 channels of RGB color.

FallNet architecture is reproduced using the Keras package by constructing the CNN layer by layer. This model is then compiled and trained with the parameters as given in Table 2.

Table 2: Training parameters

Parameter	**Value**
Optimizer	Adam (lr=0.005)
Total params	215, 505
Total trainable params	215, 505
Loss	Binary crossentropy
Metrics	Accuracy
Epochs	6

3.2.5. Model Evaluation

The model stops its training with loss and validation losses of 0.0452 and 0.0643, accuracy and validation accuracy of 0.9858 and 0.9783 which means the convergence of the model is achieved. The sensitivity, precision, and f-measure are shown in Table 3. Fig. 8 and Fig. 9 depict the overall accuracy and loss of the training process, respectively.

Table 3: Performance matrix

Accuracy	Specificity	Sensitivity	Precision	F-measure
0.9783	0.9837	0.9647	0.9591	0.9619

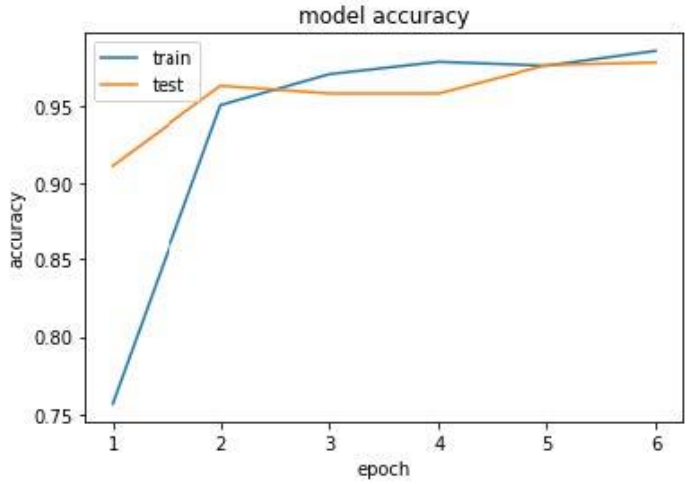

Fig. 8: Graph of accuracy against the number of epoch

The model is evaluated with the metrics from the predecessor of the fall detection model. Table 4 shows the confusion matrix and the performance of the trained CNN model.

Overall, the performance of the CNN model is very acceptable with every evaluation score acceding 0.95. With this result, the model is ready to be converted to a web-friendly model that can be used and executed in a browser. TensoFlow.js provides a convenient way to convert the Keras model into a type that is runnable in a JavaScript environment. The function returns a folder consisting of a JSON file and bin file which store the model's structure and its weights. Until this stage, the machine learning model is ready and marks the end of the model implementation.

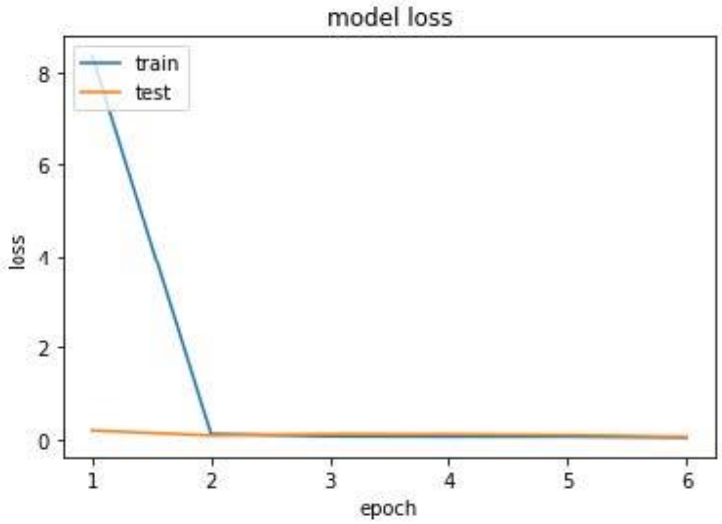

Fig. 9: Graph of loss against the number of epoch

Table 4: Confusion matrix

		Predicted Label	
		No Fall	Fall
Actual Label	No Fall	TN = 422	FP = 7
	Fall	FN = 6	TP = 164

3.3. Software Development

The software can be divided into two parts, which are the front-end and back-end. Technologies used for each part are React.js and Firebase respectively. React.js is chosen as the front-end JavaScript library because of its ease of use and powerful community support.

The software includes Visual Studio Code as the integrated development environment (IDE), PowerShell as the terminal, Node.js and NPM as the server environment and package installer, Git and Sourcetree as the version control system and Git GUI, and OBS Studio as a virtual webcam, that can play local images or videos as webcam input for testing.

Moreover, Git and GitHub are incorporated to track the progress and manage multiple branches throughout the software development. Benefits are taken from using the issues tracker to monitor the pending features and bugs, reviewing and confirming the merging of several branches, and creating a project Kanban as a issues tracker.

The email service used in this application is EmailJS. This service allows the developer to send emails solely with client-side technology. There is no need for a server; simply connect EmailJS to one of the supported email services, Gmail is used here, build an email template, and send an email using their Javascript library. This platform provides a free plan for testing that comes with 200 monthly requests and 2 email templates.

One email service has been created on the EmailJS platform to send notification emails upon the fall is detected using the assigned template. The template will receive several parameters from the client-side and render custom email content accordingly. The variables include user email, user name, current date-time, and an image URL that links to the fallen image captured.

For user authentication and image storage, Firebase is integrated into the application. Sign-in method via Google as the provider is enabled to allow the user to sign in using their Google account. The storage function is also utilized to store the fall images captured from the webcam for the signed-in users. Firebase also provides a free plan that allows 5GB of total storage and 1GB of bandwidth per day. The authenticated user will be recorded in the user list, along with their email, date created, signed-in date, and user UID.

When a fall is detected for a signed-in user, the application will capture and upload the image of current webcam footage to the Firebase storage. In the Firebase storage, a hierarchy structure of folders is created to store all the images captured from the application.

4. Discussion

4.1. Results

4.1.1. Main Page

Fig. 10 shows the main page of the application.

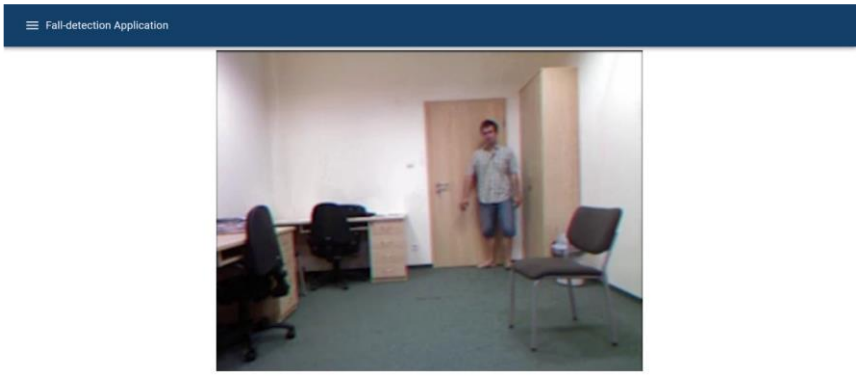

Fig. 10: Main page

The main page is made up of two primary components which are the menu bar, and the body that shows the current stream of the webcam. A blank screen will be shown if the application has no permission to access the webcam or the webcam is not found on the device.

4.1.2. Setting Menu

Fig. 11 shows the setting menu screen. Upon clicking the menu button on the app bar, a slide-out drawer will be displayed on the left side of the page, showing the current adjustable settings. A mask is also applied to the main body at the same time. Users can choose to sign in with a Google account and toggle the push notifications in the settings menu.

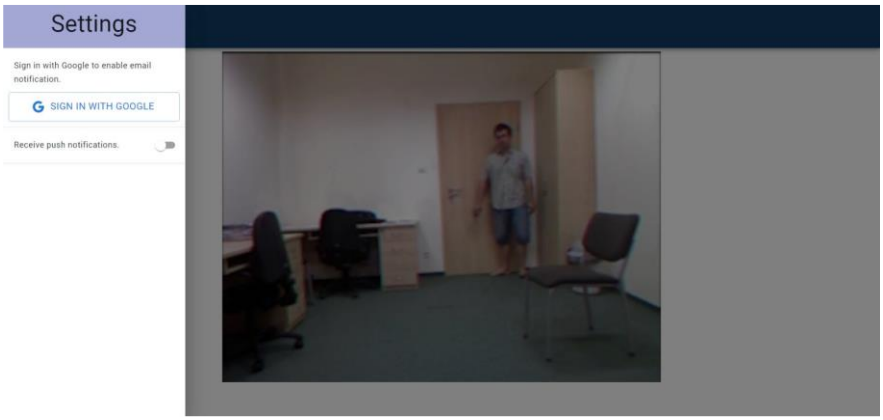

Fig. 11: Setting menu

4.1.2.1. Account

Fig. 12 shows the Sign-in pop-up window.

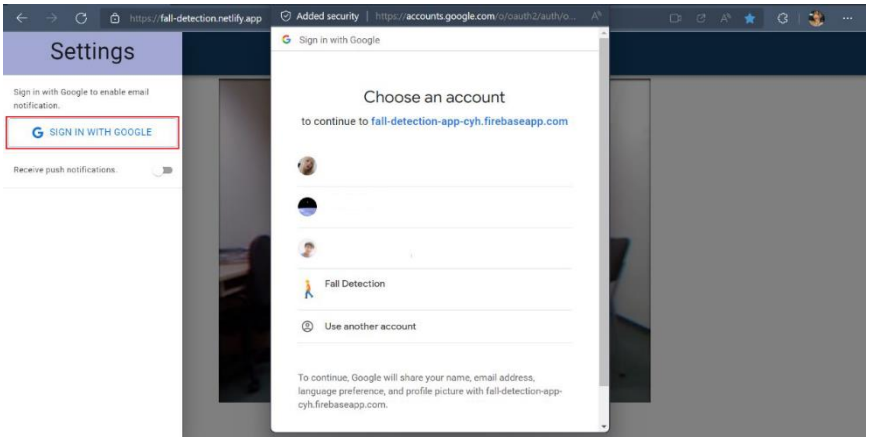

Fig. 12: Sign-in pop-up

It will be shown when the user clicks the “SIGN IN WITH GOOGLE” button. Users can select a Google account that is currently signed in inside the browser or add another Google account.

After the user has been signed in successfully, the user details including screen name, avatar, and email address will be shown in the setting menu, indicating the currently signed-in user. This is shown in Fig. 13. Sign out option is provided by clicking the “SIGN OUT” button in red. An email will be sent to the signed-in user when a fall is detected.

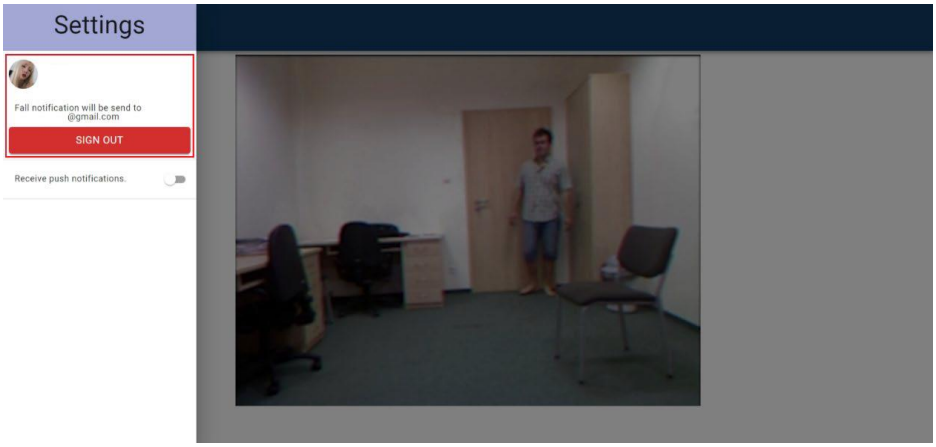

Fig. 13: User details

4.1.2.2. Push Notification

Other than that, users can toggle the push notification from the settings menu. The notification is shown in Fig. 14. The push notification will be sent to the device as an alert of the fall.

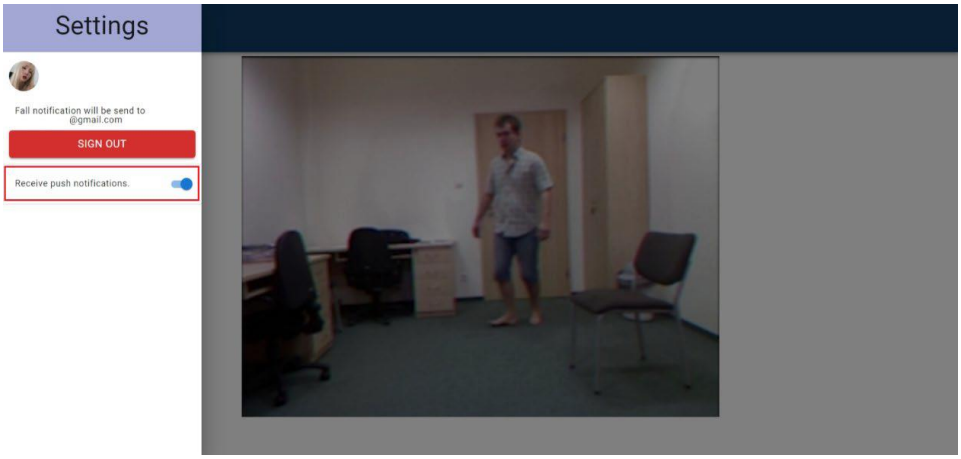

Fig. 14: Push notification toggle

4.1.3. Fall Detection

As shown in Fig. 15, the application will send an email and push notification since the user exists and the toggle of push notification setting is set to on. The push notification will be shown on the screen of the device whether the user stays on the web page or not. Clicking on this notification will redirect the user to the webpage. The green color toast on the bottom left will be shown on the web page indicating the status of sending the email. Fig. 16 depicts that this email has been sent to the signed-in user, indicating the fall occurred on 26 March 2022 at 12:32 pm local time. The user can decide the following action based on the provided link in the email. An example of the link's content is shown in Fig. 17.

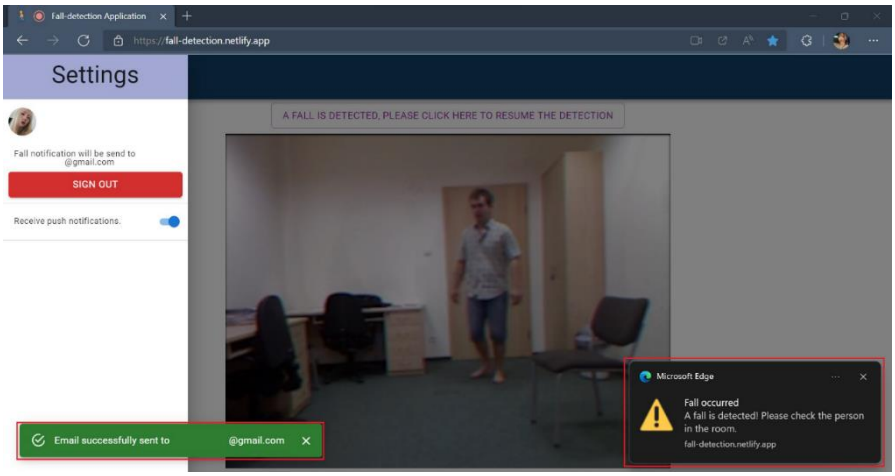

Fig. 15: Notifications and email status

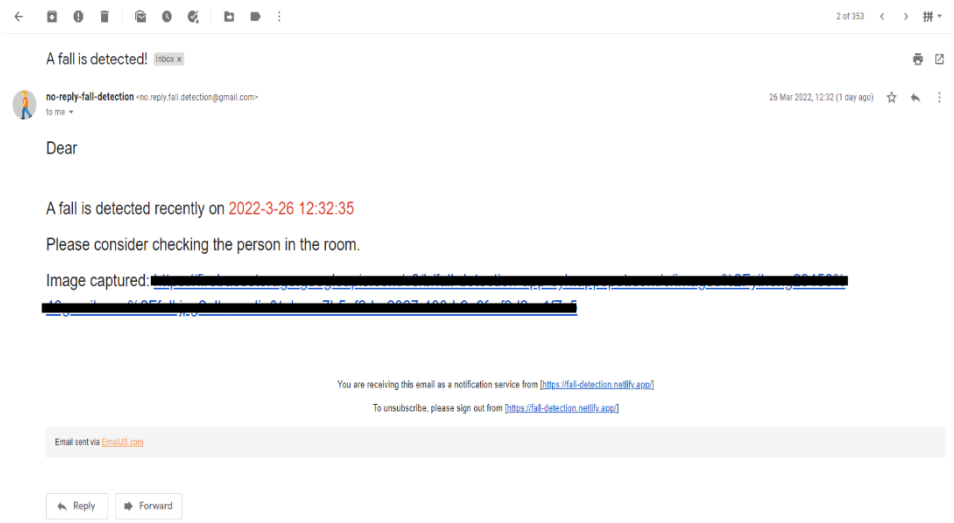

Fig. 16: Email received

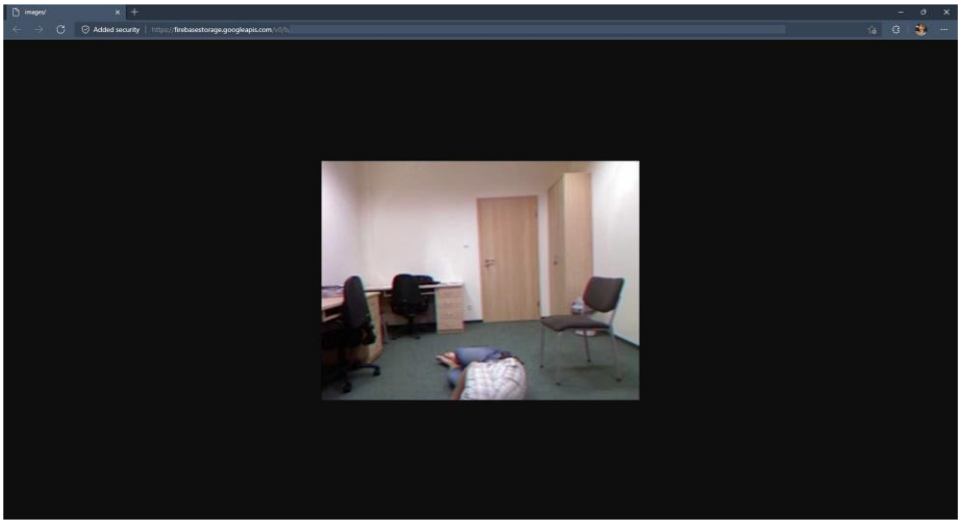

Fig. 17: Image captured

4.1.4. PWA Features

The PWA technology is implemented into the final application. This feature will allow faster and easier access to the application for both desktop (Fig. 18) and mobile devices (Fig. 19). A special icon appears on the right end of the address bar, indicating that this web application is installable on the device. The screen of Installed application on home page is shown in Fig. 20.

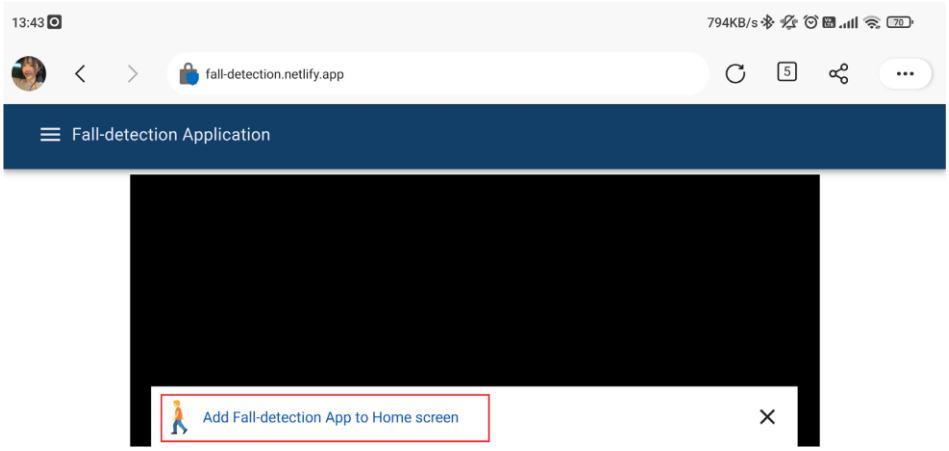

Fig. 18: Installed application on desktop

4.2. Limitations & Challenges

The fall detection application is not flawless, it contains some limitations for now. Firstly, the application does not detect falls in real life with very high accuracy. One of the reasons for this is that the model is trained on experimented images. Although

the URFD dataset provides multiple actors and scenarios, the machine learning model cannot adapt to complex backgrounds and viewing angles which differ for every user. The fall detection also doesn't perform well under various illumination conditions.

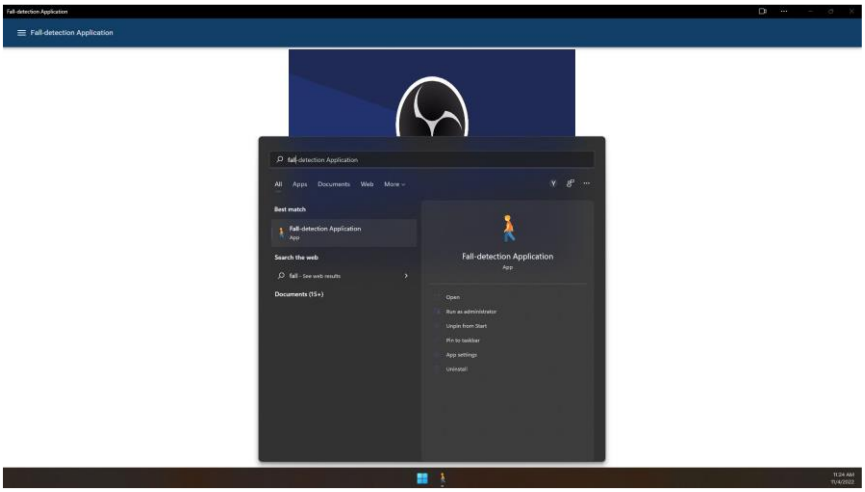

Fig. 19: PWA installation prompt on mobile

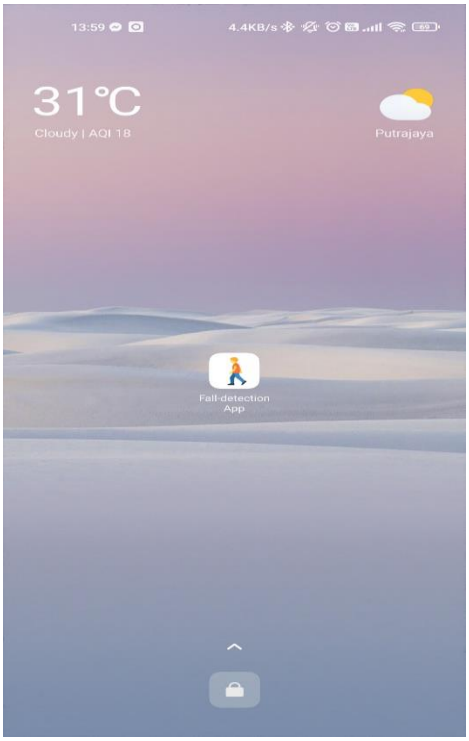

Fig. 20: Installed application on home page

Other than that, the biggest limitation of the application is the services it uses. Two online services are used to send an email and to store captured images. EmailJS has a monthly quota of 200 emails for the free plan while Firebase provides 5GB of free storage for the free plan. These limitations force the current application to limit the number of emails per fall to one and each user can only store an image in the database, which means the user could not retrieve older images.

Moreover, the current machine learning model performs predictions with certain delays. This is because the model is designed to receive the entire image as the input which leads to a higher amount of time needed to process the image.

There are a few issues encountered throughout the development and implementation process. Firstly, testing the application is a challenging task as the application reads video footage from the current webcam to perform fall detection, which means the footage must be modified to a video input for testing purposes. The solution for this is to use OBS as a virtual webcam to stream local video files as webcam input that makes testing out the application possible.

5. Conclusion

5.1. Achievements

The in-room fall detection application is a web application that can detect fall events from the video stream of the webcam using a custom CNN machine learning model. Functions such as signing in to receive email notifications and toggling push notifications are implemented to provide multiple options to the user for desired use cases.

During the initial stage of planning, the background of the area of interest has been studied for a deeper understanding of the problems. This also provides possible solutions for the problem aimed to be solved. Other than that, the requirements of the project were analyzed and presented with the combination of the suitable machine learning technique and the software requirements. The detailed explanations of each component are listed and defined with the dataset used, data preprocessing, the architecture of the selected machine learning technique, and the software designs which are represented with several technical drawings.

In the development phase, the model is trained using the URFD image data using Python and Tensorflow package. Evaluation is performed on the testing data, which is separated from the main dataset initially. With the decent readings on all the evaluation matric, the model is saved and converted to a web-friendly format for future use. The software is built using a JavaScript frontend library called React and Firebase as the backend storage that provides authentication service as well. Finally, the production build application is wrapped up and hosted on Netlify for public access.

This paper demonstrates the integration of a custom machine learning model that is built and trained to classify falls from an image, which is then deployed and utilized

by the web front-end library.

5.2. Future Works

In the fullness of time, improvement of the application should be done to enhance its functionalities and performance. One of the features that can be added to the application is supporting multiple cameras. With multiple video streams, the application can be transformed into a control panel to report any falls in different rooms or spaces.

Apart from that, a more in-depth research is required to improve the generalization of the machine learning model. Multiple approaches can be done to minimize or eliminate the effect of different backgrounds such as extracting the human silhouette, adding a bounding box to exclude background, and performing training on multiple datasets with different environments and illuminations.

Lastly, a custom email server can be made and deployed to send more emails and a scalable database that provides larger storage, allowing the user to retrieve fall images that are being captured previously.

Authors' Contributions

Author 1: System development and testing, formal analysis, data curation, report writing; Author 2: Supervision, review & editing.

References

Cai, X., Li, S., Liu, X., & Han, G. (2020). Vision-Based Fall Detection With Multi-Task Hourglass Convolutional Auto-Encoder. IEEE Access, 8, 44493-44502. doi:10.1109/ACCESS.2020.2978249

Chen, Y., Li, W., Wang, L., Hu, J., & Ye, M. (2020). Vision-Based Fall Event Detection in Complex Background Using Attention Guided Bi-Directional LSTM. IEEE Access, 8, 161337-161348. doi:10.1109/ACCESS.2020.3021795

Cumming, R., Salkeld, G., Thomas, M., & Szonyi, G. (2000). Prospective study of the impact of fear of falling on activities of daily living, SF-36 scores, and nursing home admission. The journals of gerontology. Series A, Biological sciences and medical sciences, M299-305. doi:10.1093/gerona/55.5.M299

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248-255. doi:10.1109/CVPR.2009.5206848

Girshick, R. (2015). Fast R-CNN. Retrieved from https://github.com/rbgirshick/

Harrou, F., Zerrouki, N., Sun, Y., & Houacine, A. (2017). Vision-based fall detection system for improving safety of elderly people. IEEE Instrumentation & Measurement Magazine, 20(6), 49-55. doi:10.1109/MIM.2017.8121952

Harrou, F., Zerrouki, N., Sun, Y., & Houacine, A. (2019). An Integrated Vision-Based Approach for Efficient Human Fall Detection in a Home Environment. IEEE Access, 7, 114966-114974. doi:10.1109/ACCESS.2019.2936320

Kwolek, B., & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer Methods and Programs in Biomedicine, 117(3), 489-501. doi:10.1016/j.cmpb.2014.09.005

Lin, C. -Y., Wang, S. -M., Hong, J. -W., Kang, L. -W., & Huang, C. -L. (2016). Vision-Based Fall Detection through Shape Features. 2016 IEEE Second International Conference on Multimedia Big Data (BigMM), (pp. 237-240). doi:10.1109/BigMM.2016.22

Núñez-Marcos, A., Azkune, G., & Arganda-Carreras, I. (2017). Vision-Based Fall Detection with Convolutional Neural Networks. Wireless Communications and Mobile Computing. doi:10.1155/2017/9474806

Panahi, L., & Ghodsb, V. (2018). Human fall detection using machine vision techniques on RGB–D images. Biomedical Signal Processing and Control, 44, 146-153. doi:10.1016/J.BSPC.2018.04.014

Reddy, G. P., & Geetha, M. K. (2010). VIDEO BASED FALL DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORK. European Journal of Molecular & Clinical Medicine, 7(11), 739-748. Retrieved from https://ejmcm.com/article_5270.html

Santos, G. L., Endo, P. T., Monteiro, K. H., Rocha, E. d., Silva, I., & Lynn, T. (2019). Accelerometer-Based Human Fall Detection Using Convolutional Neural Networks. Sensors, 19(7), 1644. doi:10.3390/s19071644

Soomro, K., Zamir, A. R., & Shah, M. (2012). A Dataset of 101 Human Actions Classes From Videos in The Wild. doi:10.48550/ARXIV.1212.0402