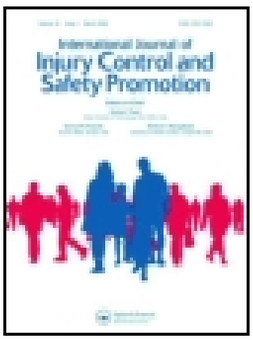

International Journal of Injury Control and Safety Promotion

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/nics20

Automatic fall detection using region-based convolutional neural network

Ghada Khaled Hader , Mohamed Maher Ben Ismail & Ouiem Bchir

To cite this article: Ghada Khaled Hader , Mohamed Maher Ben Ismail & Ouiem Bchir (2020): Automatic fall detection using region-based convolutional neural network, International Journal of Injury Control and Safety Promotion, DOI: 10.1080/17457300.2020.1819341

To link to this article: https://doi.org/10.1080/17457300.2020.1819341

Published online: 15 Sep 2020.

INTERNATIONAL JOURNAL OF INJURY CONTROL AND SAFETY PROMOTION
https://doi.org/10.1080/17457300.2020.1819341

Automatic fall detection using region-based convolutional neural network

Ghada Khaled Hader, Mohamed Maher Ben Ismail and Ouiem Bchir

Department of Computer Science, King Saud University, Riyadh, Saudi Arabia

ABSTRACT

The common classifiers usually used to detect fall incidents depend on building and maintaining complex feature extraction for accurate machine learning tasks. However, these efforts have not succeeded in determining an ideal classifier or feature extraction for fall detection. In this research, we address the feature extraction problem along with the choice of the most appropriate classifier by using deep learning where the most prominent features are learned over the numerous layers of the network. More specifically, a general framework that relies on a faster region-based convolutional neural network was designed and developped to recognize the fall incidents. In particular, we designed three custom architectures and exploited transfer learning by using pre-trained networks such as the VGG-16 and AlexNet to overcome the fall detection challenge. The performance of the proposed networks showed the advantage of the pre-trained networks, where VGG-16 scored highest in those measures followed by AlexNet, the custom networks showed impressive results that were close to the pre-trained networks.

ARTICLE HISTORY
Received 26 May 2020
Revised 31 August 2020
Accepted 1 September 2020

KEYWORDS
Machine learning; neural networks; transfer learning; digital image processing; fall detection

1. Introduction

Falls are considered as one of the common causes of fatal injuries. Statistically, falls are the second cause of death for unintentional injury-related deaths among all ages (Mubashir et al., 2013). Internationally, between 28% and 34% of older people experience a fall once a year. Furthermore, 40%–60% of these falls lead to injuries that vary from stroke, bone fractures and may lead to death (Alrashed, 2017; Lin et al., 2005). Such statistics are even higher in Saudi Arabia's local community. In fact, 49.9% of elderly Saudis, aged 60 and above, experience falling at least once a year where 83% of these falls happened indoors. Moreover, 74% of those who had fallen had post-fall injuries (Almegbel et al., 2018). Besides, the consequences of such falls surpass physical injuries to psychological problems where elders suffer from anxiety and fear of falling again and fear of not being able to walk and stand. These consequences affect their quality of life considerably and make them even more dependent.

Wearable devices, ambiance-based devices and vision-based devices have been introduced by the research community to overcome the fall detection challenge (Ben Ismail & Bchir, 2017; Mubashir et al., 2013). Wearable devices and ambiance-based devices are profoundly affected by noise and introduce a high level of intrusion. Usually, such sensors provide somewhat crude data that is difficult to interpret. Besides, the signals produced by the acceleration of body movement might detect safe and daily movements as falls and might face difficulties in detecting slow falls. Additionally, elders may forget to wear the devices, especially if they are struggling with memory problems (Fathima et al., 2014; Foroughi, Aski et al., 2008; Foroughi, Naseri et al., 2008). Vision-based devices are considered more usable and more appropriate since they represent a non-intrusive solution, require no wearable devices and are more robust to noise (Ben Ismail & Bchir, 2017). However, the use of videos/images for such applications triggers critical challenges such as the choice of the appropriate visual descriptor(s), learning technique(s) and algorithm(s). In other words, the performance of typical image-based fall detection systems is drastically affected by the feature extraction and machine learning (ML) components. Particularly, the high variability of fall movements makes the problem more challenging and limits the discriminatory information captured by the extracted features (Egmont-Petersen et al., 2002; Fathima et al., 2014).

Typically, various statistical and ML methods have been used to address both: the inference and the prediction problems. However, ML methods focus on prediction using general-purpose learning algorithms in order to mine hidden patterns in row data. In particular, such methods have been efficiently adopted for applications where the number of dimensions/variables/features exceeds the number of instances as the case of image and video datasets. In other words, ML puts 'light' assumptions on the data generation process. Besides, for some ML methods, the absence of an explicit model yields a challenging interpretability of the results. On the other hand, statistical methods concentrate on inference through the formulation and fitting of application-specific probability model. These models yield the computation of quantitative measure, such as the confidence, that a

CONTACT Mohamed Maher Ben Ismail mbenismail@ksu.edu.sa

discovered relationship describes a 'true' effect that is unlikely to result from noise. One should also note that ML and classical statistics exhibit different computational tractability when the number of features/attributes per instance increases. In fact, classical statistical modeling was intended to handle relatively small data and a few dozens of input features/attributes. In other words, it deals with less complex association problems. However, the boundary between statistical and ML gets hazier as the number of features/attributes as well the problem complexity increases as in the case of the image-based fall detection problem.

In this article, we investigate deep learning to overcome the feature extraction challenge for the image-based fall detection application. Specifically, we use a faster region-based convolutional neural network (R-CNN) to recognize the fall incident. Further, we exploit transfer learning and pre-trained neural networks such as the VGG-16 (Simonyan & Zisserman, 2015) and AlexNet (Krizhevsky et al., 2012) to design an efficient deep learning architecture in order to overcome the fall detection challenge. In particular, a training image collection is used to train the proposed network to learn a model able to detect fall incidents in an indoor environment.

2. Literature review

The earliest efforts aiming to design vision-based fall detection systems adopted supervised learning algorithms to classify the captured images/frames as 'Fall' or 'No-Fall'. Yu et al. (2012) used the support vector machine (SVM) (Cortes & Vapnik, 1995) to categorize four different types of posture. The issue with such supervised learning algorithm is the dependence of their performance on the extracted visual features. This led the researchers (Ben Ismail & Bchir, 2017) to map the visual descriptors to a more discriminative histogram feature using an unsupervised learning algorithm. The clustering algorithm they used generates possibility membership functions that are then fed into a K-Nearest Neighbors (KNN) (Stevens et al., 1967) classifier to categorize the captured frames and detect fall incidents.

Some state of the art works (Jackson et al., 2005; Miaou et al., 2006) used shape features to surround the individual body and improve the fall detection overall accuracy. Others used object bounding boxes or ellipses to track individuals and support the fall detection task (Foroughi, Aski et al., 2008; Foroughi, Naseri et al., 2008; Nait-Charif & McKenna, 2004). In particular, Qian et al. (2008) used two bounding boxes or more to detect falls. They divided the human body into several parts based on its anatomy and assigned a weight to each part before they applied the SVM (Cortes & Vapnik, 1995) classifier. Similarly, the dynamics of the movement of the human body inside of the box was the main focus of the work introduced by Yun and Gu (2016). In fact, since the velocity and motion change significantly when falling, the rate of change in motion and velocity are used to extract more discriminative features. Then, the authors considered the motion inside of the box as connected points, which made their work less sensitive to the angle of the camera (Yun & Gu, 2016). Unlike other works that are restricted to some movement patterns, Foroughi, Aski et al. (2008) and Foroughi, Naseri et al. (2008) considered a more extensive range of movements that vary from regular and daily activities such as walking and running to lying down, bending and sitting down. They also included a more abnormal behavior such as limping or stumbling and falling. The authors combined the integrated time motion images (ITMI) with Eigen-space techniques then fed the resulting features into a multilayer perceptron (MLP) neural network (Foroughi, Naseri et al., 2008) to classify the motion into Fall or No-Fall event precisely. This approach is able to detect a fall after inactivity in places that are not considered inactivity zone like chairs, sofas. Since it is the storage of the whole video and its browsing for motion information is resource consuming, the authors used motion-history images (MHI) and motion-energy images (MEI) (Bobick & Society, 2001) to combine the whole motion sequence into one image to be used for comparison with the stored movements. As in the work by Foroughi, Aski et al. (2008), the authors used the head position because it's usually easily visible, and it has a very large movement when the individual falls. More specifically, the head position is first detected in each frame (Foroughi, Aski et al., 2008). Cireşan et al. (2011) pre-processed the image content, extracted the visual features and fed the resulting low-level feature into a convolutional neural network (CNN) (LeCun, 1997). These networks are trained with online gradient descent and use max-pooling instead of using subsampling layers, which results in faster convergence time. Nevertheless, these improvements are time-consuming and computationally expensive.

Rougier et al. (2011) used edge points obtained from a person's silhouette by comparing two consecutive person silhouettes using the edge points. A Canny edge detector was used to generate these edge points. The resulting edges are then matched throughout the video sequence to track the deformation silhouette. Consequently, they classify the movement as Fall or No-Fall based on the shape deformation during the fall, followed by a lack of significant movement after the fall. In other research, the center-of-mass for people's silhouette and its distance from the floor plane was used by Diraco et al. (2010). The authors used a 3D camera that provides a depth map, gray-level images and 3D coordinates of the points representing the scene. It's necessary to calculate the distance between the person's silhouette's centroid and the floor.

Other contributions used the geometrical and spatial orientation of the silhouette, and the speed of the change in its center to detect the fall incident (Anderson et al., 2010), where they used the silhouettes captured using different cameras, to build a set of voxels and combine the centroid, height and significant orientation of the body. They used the Fuzzy logic with an inference engine that operates on rules in an IF–THEN format. The output of the system consists of fuzzy membership sets where the goal for each image is to calculate the membership degree of each

person's voxel with respect to the pre-defined classes. The outputs are processed temporally to generate temporal linguistic summarizations that are input to another fuzzy inference system to be able to classify the activity. Other features, such as the ratio of height and weight of the human body were used by Miaou et al. (2006) as a feature for better characterization of the fall incident. The images were obtained using an Omni-camera that captures a 360° view. The background is cleaned by randomly selecting a number of frames and then calculating their average to reduce the noise in the images. Then, all the objects' height and width are labeled to form the training set. Noise removal is also an essential step (Liu et al., 2010) where the ratio and difference of the bounding box width and height of a human body's silhouette are used as a feature for better discrimination between fall and fall-free categories. Note that the upper limb influence is reduced by using the vertical projection histogram of the silhouette image. Whenever the person stretches his/her arm, a peak in the distribution curve is recorded by taking the individual mean and standard deviation. Then this extreme value is replaced to eliminate the effect of upper limb activities. Hazelhoff et al. (2008) used the orientation of a person's main axis and the ratio of the variances in horizontal and vertical directions to express the subject position. Two cameras equipped with a tracker were used to track the detected objects. In addition, the objects were marked as human or non-human based on the motion and the head region; if the object is motionless and doesn't have a head region, this object is marked as non-human. In the case of a fall, at least one main axis angle would be large in absolute value, and both variance ratios would be small, indicating that both angles can be coupled together. In the work by Vishwakarma et al. (2007), the aspect ratio and the gradient distortion of an object in a two-dimensional plane were used as features. When a fall happens, a drastic change occurs with respect to either x or y axes. If a fall is detected, a confirmation step is applied to calculate the fall angle to confirm the incident. Note that the next seven frames are taken into consideration to analyze the fall pattern.

The selection of the relevant features for fall detection is a challenging task since those features experience a large variation depending on the individuals themselves when it comes to shapes, sizes and habits. An alternative to overcome this issue is the use of an automated approach that is able to learn the features and capture the details of the human motion. This can be accomplished by using deep learning techniques where it can learn the most effective features over the several layers it has (Jokanovic et al., 2016). This led different researchers to use deep learning for detecting falls, either based on vision or non-vision approaches. Lu et al. (2018) used three-dimensional CNN that uses video kinematic data and employs a temporal sequence to extract motion features without specifying the features. Also, to increase the accuracy, the region of interest was located using a spatial visual attention scheme that is based on long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997). Another work (Jokanovic & Amin, 2018) introduced the range-Doppler radar, also known as frequency modulated continuous wave (FMCW). It implemented deep learning in the form of two stacks sparse autoencoder (neural network) for feature extraction and preform classification using Softmax (logistic regression). The feedback from the radar in the frequency domain was used as an input signal. In the work by Doulamis (2016), the human body was first extracted from the background regardless of the visual environment, then tracked using a geometric feature. Specifically, a 3D measurement of the human body was extracted from the 2D images and fed into a time delay neural network (TDNN) (Waibel et al., 1990) to model fall patterns.

3. Proposed approach

In this work, we use deep CNN (LeCun, 1997) to overcome the critical choice of the most appropriate visual descriptor on which depends the performance of the fall detection system. The CNN is a sub-type of deep neural networks that are used to process two-dimensional data. This made them mostly used for image-based learning tasks as they can learn the optimal features to be used for classification and/or detection applications. The topology of CNN would leverage the spatial relationship and would enhance the performance by reducing the number of parameters in the network; this gives CNN better results than usual backpropagation algorithms (Liu et al., 2017). Furthermore, CNN is one of the most used approaches for deep learning for feature extraction (Cireşan et al., 2011; He et al., 2015; Jokanovic & Amin, 2018). However, CNN is considered computationally expensive because the training phase consists of multiple stages and exhibits large space and time requirements that yield a slow detection process (Druzhkov & Kustikova, 2016). Particularly, we adapt the Faster R-CNN that shares convolutions and addresses this CNN limitation. An overview of the Faster R-CNN architecture is shown in Figure 1. As can be seen, the proposed architecture is composed of two fully integrated main components. The first one is a deep fully convolutional network that is intended to propose candidate regions called region proposal network (RPN). This first part is fed with static video frames, and the output consists of a set of candidates bounding boxes. The second component is a Fast R-CNN detector which processes the video frames along with regions of interest (RoI). As can be seen, these two components are aggregated into one unified network. More specifically, the convolutional layers are shared by both networks, which allow one single learning/training task. Finally, the Fast R-CNN detector part of the network ends with fully connected layers ending with two output layers: The first one generates the softmax probability for the class 'Fall' or 'Non-Fall' while the second layer produces the position of the bounding box surrounding the falling person.

In this research, we designed five deep neural networks to be integrated in the convolutional layers' component of the Faster R-CNN architecture. Three were custom networks while we exploited transfer learning and pre-trained two

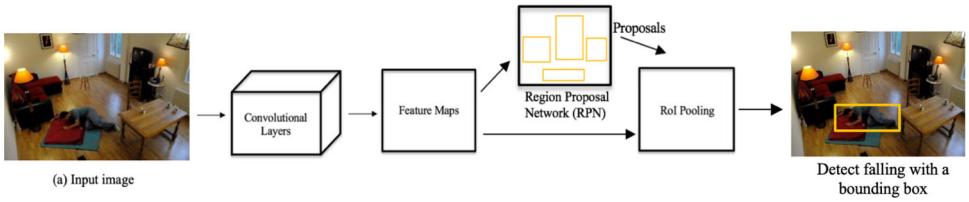

Figure 1. The Faster R-CNN architecture.

neural networks using VGG-16 (Simonyan & Zisserman, 2015) and AlexNet (Krizhevsky et al., 2012). The custom network and the pretrained networks consist of multiple layers as shown in Figures 2 and 3, respectively. The first layer is an image input layer with a specified input size $p \times p$ where p refers to the pixel size and with three channels corresponding to the RGB colors.

Typically, for classification tasks this input is the size of the image. However, for detection CNN needs to work on smaller parts of the image so we set it to the smallest bounding box size of a fall in the dataset. The next middle layers consist of a two-dimensional convolution layer with $p \times p$ filter size and a number of filters with a set stride and padding. Afterwards a rectified linear unit (ReLu) layer follows which is an activation layer that removes the negative part of a function (Glorot & Bordes, 2011):

$$f(x) = \begin{cases} 0 & x<0 \\ x & x \geq 0 \end{cases} \quad (1)$$

Following the ReLu layer, a $p \times p$ max pooling layer with a predefined stride are set to determine the step of the pooling function. Those layers (convolution, ReLu, max pooling) may be repeated multiple times.

The final layers are composed of a particular number of neurons in a fully connected layer, followed by ReLu layer, then another fully connected layer with size two for the two labels 'Fall' and 'No-Fall'. Finally, a Softmax layer which gives the probability distribution for each class, then a classification output layer which is used to calculate the cross-entropy loss for mutually exclusive classes; in our case 'Fall', 'No-Fall'. The classification layer follows the softmax layer and takes the output and assigns each input using a multi-class cross-entropy error function for a *1*-of-*K* coding scheme where K indicates the class (Glorot & Bordes, 2011):

$$\text{loss} = \sum_{i=1}^{N} \sum_{j=1}^{K} t_{ij} \ln y_{ij} \quad (2)$$

where N denotes the number of training samples, K is the number of classes, t is the target vector where t_{ij} represents the training sample i with the j th class, y_{ij} indicates the value of the Softmax function which is the probability of sample i belonging to class j.

The training algorithm can be summarized using the following four steps given an initial training rate:

1. The RPN is trained using a stochastic gradient descent with momentum (SGDM) (Murphy, 2012). The update function is:

$$\theta_{\updownarrow+1} = \theta_{\updownarrow} - \alpha \nabla E(\theta_{\updownarrow}) + \gamma(\theta_{\updownarrow} - \theta_{\updownarrow-1}) \quad (3)$$

where the iteration number is denoted as $\updownarrow$. The learning rate is $\alpha > 0$, while θ represents the parameter vector. $\nabla E(\theta_{\updownarrow})$ denotes the gradient of the loss function. Finally, γ represents the contribution of the previous gradient step to the current iteration. All new layers are initialized with weighting values from a zero-mean Gaussian distribution with standard deviation 0.01.

1. Since Faster R-CNN is composed of a deep convolutional network of RPN and a Fast R-CNN the features are shared by training a Fast R-CNN using the proposal generated by RPN from the first step.
2. Fix convolutional layer and fine-tune the unique layers of RPN initialized by the Fast R-CNN in step 2.
3. Fix convolutional layer and fine-tune the fully connected layer of the Fast R-CNN.

3.1. Custom network architectures

Initially, various setting combinations of the network parameters were investigated in this research. This was intended to determine the best custom convolutional neural network to couple with the Faster R-CNN. This shoed that setting a max-pooling layer for each set of convolutional and ReLu layers has no effect, and sometimes it actually decreases the model performance. The most promising results were achieved using a max-pooling layer every other convolutional and ReLu set of layers. Moreover, it has been found that increasing the size of the fully connected layer has a positive impact on the performance while having two fully connected layers without a dropout layer yields the best results. Besides, adding a dropout layer either after each fully connected layer or after some of them degrade the results. Thus, the best three architectures that yielded the best results and performance share the same first 11 layers as shown in Figure 4. On the other hand, their last four layers are reported in Figure 5. More specifically, the first common layers in Figure 4 consist of a size of 64×64 for the image input layer, and four sets of convolutional and ReLu layers with '1' padding for the convolutional layers, a 2×2 max pooling with stride two after every two sets of convolutional and ReLu layers. The first two convolutional layers have 64 filters, and the last has 128 filters.

On the other hand, the last four common layers, shown in Figure 5, are composed of a ReLu layer, fully connected layers, a softmax layer and a classification output layer.

The custom networks investigated in this research differ starting from the 12th layer, as detailed in Figure 6. As it can be seen, the first architecture (Network A) consists of

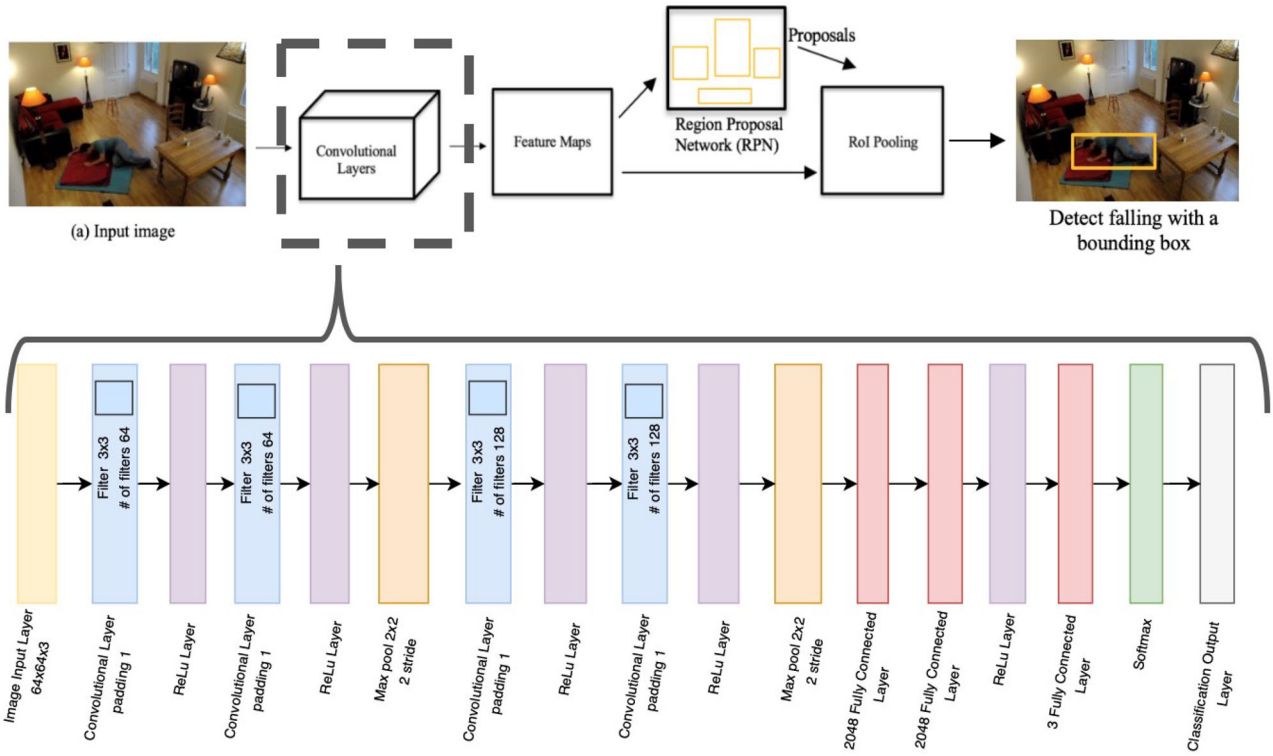

Figure 2. Illustrative architecture of the proposed network.

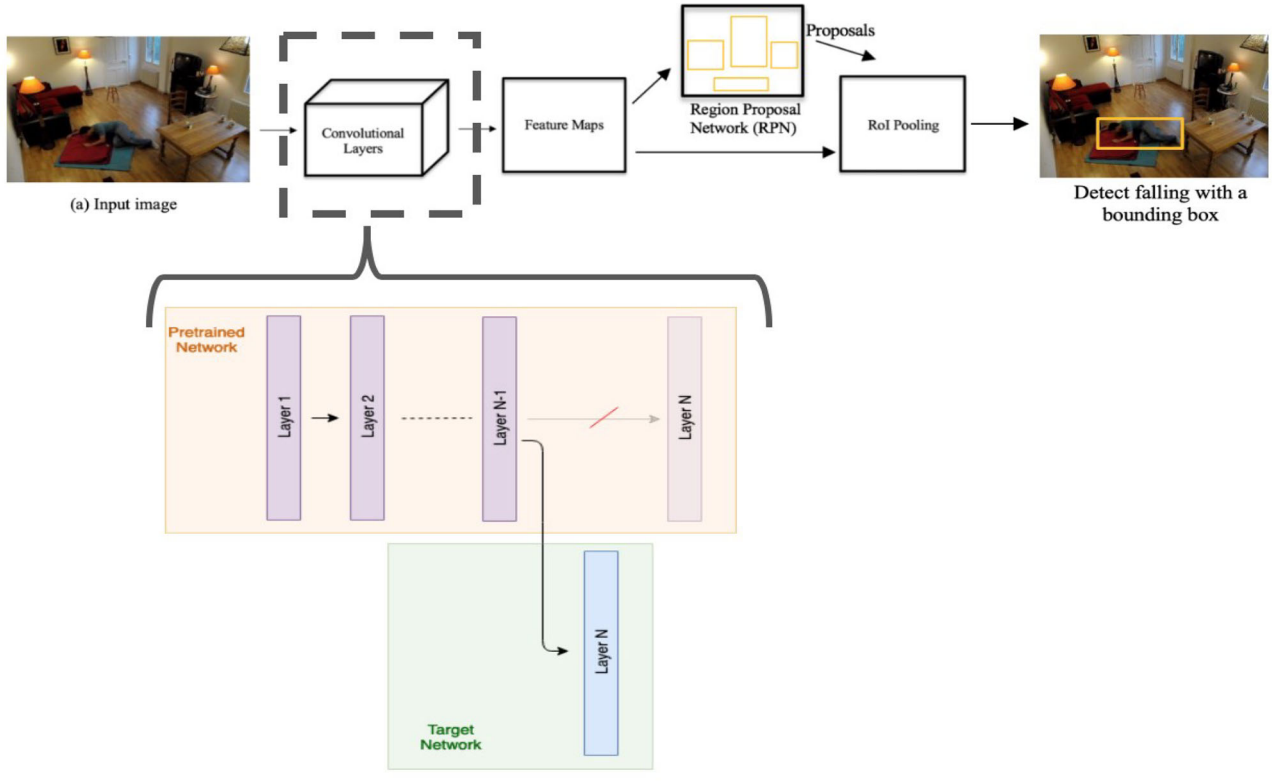

Figure 3. Illustrative architecture of the pretrained network.

two fully connected layers of 2048 with a dropout layer after the second fully connected layer, while the second architecture (Network B) consists of two fully connected layers of 4096 without dropout layer. Finally, the third architecture (Network C) consists of two fully connected layers of 4096 with a dropout layer between the fully connected layers.

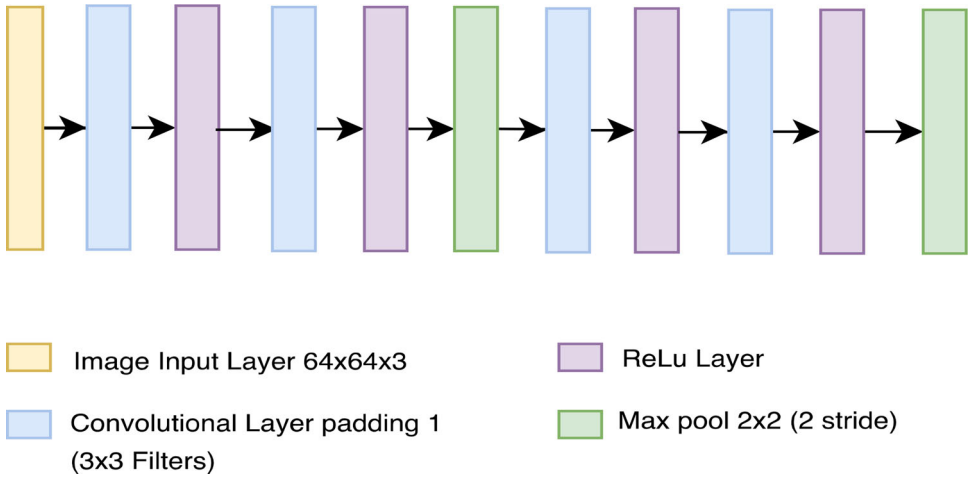

Figure 4. First 11 common layers of the custom networks.

Figure 5. Last four common layers of the custom networks.

3.2. Pre-trained network architectures

For the pre-trained AlexNet (Krizhevsky et al., 2012) and VGG-16 (Simonyan & Zisserman, 2015) networks, the first ten layers' weights were unchanged, while the last fully connected layer was replaced with a three outputs layer. We replaced the last layer (classification output layer) to fit the updated fully connected layer. Specifically, we have frozen the initial weights for the original 10 first layers of AlexNet (Krizhevsky et al., 2012) and VGG-16 (Simonyan & Zisserman, 2015) networks, respectively.

4. Experiments

In this paper, we use the LE2I DIJON UMR6306 dataset (Charfi et al., 2013). This dataset encloses videos captured in four different locations (home, coffee room, lecture room and office). The videos include 'Fall' and 'No-Fall' scenarios with various lighting conditions and recording angles. For this research, we have broken down those videos into frames of size 320×240 pixels using a function in MatLAB ver. R2018a which generated those frames. Among those frames, a collection of 3957 frames was collected and manually tagged with a bounding box and a label of Fall or No-Fall around the person in the image. This manual labeling process showed that the dataset includes different falling scenarios: A fall can happen from a standing or walking position, standing on ladders, sleeping or laying on a bed or sitting position. It is also important to note that falls have similar characteristics with natural postures and movements such as crouching. Thus, 1829 frames have been labeled as 'Fall', while 2128 others belong to the 'No-Fall' class.

4.1. Settings

For both custom and pre-trained networks, there were common settings that had the same effect on the custom neural network and on the pre-trained networks. Specifically, these settings include the number of epochs, the initial learning rate and the threshold for the intersection over union (IoU). The number of epochs has been utilized in this experiment to balance between the time of execution and the quality of the result. For our experiments, we found that 10 epochs have given us good result with a reasonable amount of training time. As for learning rate, we reach the most optimal rate of 0.001 after experimenting with rates of 0.0001 and 0.0005 to 0.005 and 0.001. Lastly, increasing the threshold for the IoU to be 0.7 has a negative impact on the precision of the detection.

We calculated the general accuracy for both Fall and No-Fall, and we computed the precision, recall, miss rate and F1 measures separately for Fall and No-Fall classes in addition to the average performance. These measures were assessed for the average of the folds in the validation stage then measured it again at the testing stage.

4.2. Validation results

The results for VGG-16 (Simonyan & Zisserman, 2015) and AlexNet (Krizhevsky et al., 2012) had extremely close results in the average for Fall and No-Fall for the different performance measures, as seen in Table 1. Accuracy is highest for the pre-trained networks; with VGG-16 (Simonyan & Zisserman, 2015) having the highest accuracy of 0.995 compared to 0.991 scored by AlexNet (Krizhevsky et al., 2012). Network A comes after, leading the custom networks with an accuracy of 0.9199, leaving Network B and Network C with almost the same accuracy of 0.89232 and 0.8926. As for the other performance measures, the pre-trained networks are still in the lead where VGG-16 (Simonyan & Zisserman, 2015) scores 0.99466, 0.99526 and 0.99494 for recall, precision, and F1 and 0.00534, 99.46672 and 0.53328 in miss rate, TP rate, and FP rate. AlexNet (Krizhevsky et al., 2012) achieves almost similar scores with a slight

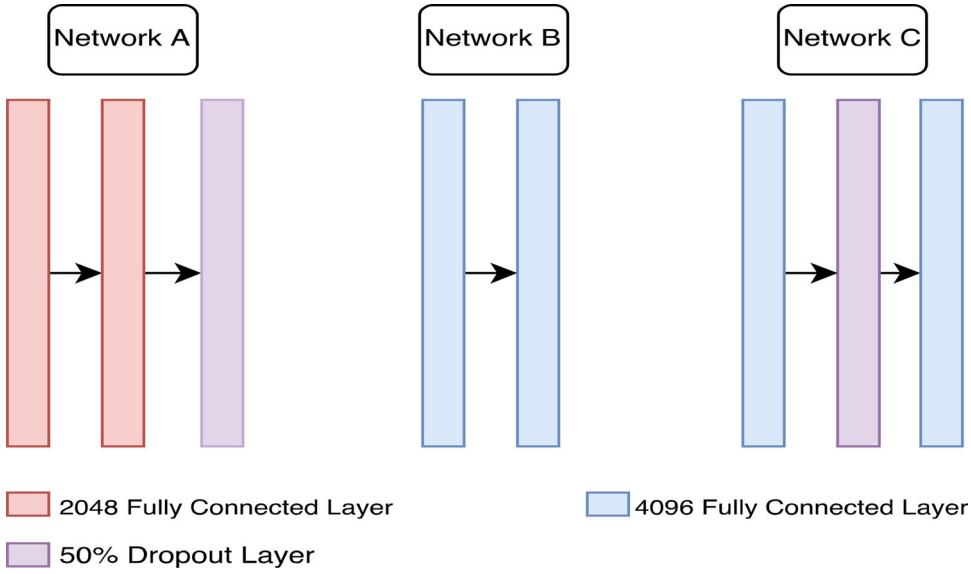

Figure 6. Different middle layers of the different custom networks

decrease in each measure. After that, for custom networks, Network A shows the best results across the different performance measures where Network B and C fall behind with extremely close results to each other.

It can be clearly seen that pre-trained VGG-16 (Simonyan & Zisserman, 2015) and AlexNet (Krizhevsky et al., 2012) yielded similar performance across the different performance measures, with a slight advantage for VGG-16 (Simonyan & Zisserman, 2015). This advantage in performance holds for precision, recall, and F1 for both Fall and No-Fall. In Falls, as seen in Table 2, VGG-16 (Simonyan & Zisserman, 2015) scored 0.99174, 0.9974 and 0.99454 as recall, precision and F1 respectively, leaving AlexNet (Krizhevsky et al., 2012) in an extremely close proximity of 0.98992 for recall, 0.99046 for precision and 0.99018 for F1. As for custom networks in Falls, Network *C* scores best in recall, TP rate and miss rate with corresponding scores of 0.92602, 92.60196 and 0.07398. However, the FP rate is extremely high, reaching 13.3838; such a high number indicates that the network would give bad results due to the high number of false positives. The lowest FP rate of 7.42696 is seen in Network B, where it is showing a better performance in FP and precision equalling 0.90642. But it scores the worst in TP rate. This makes Network A as the optimal result with the advantages in the accuracy of 0.9199, and F1 of 0.91118. Network A comes as second best in all other performance measures where it scores very close to the best scores of custom networks, achieving results of 0.91816, 0.90522 for recall and precision.

For the results of detecting No-Fall, Table 3 shows that VGG-16 (Simonyan & Zisserman, 2015) has the best scores again across all the performance measures where it scored 0.99758, 0.99312 and 0.99534 in recall, precision and F1 compared to 0.992, 0.99136 and 0.99168 for AlexNet (Krizhevsky et al., 2012) in the same measures. After those two networks come Network A, where it has the highest F1 among custom network reaching 0.92676. Network B scores best in recall, miss rate and TP rate having scores 0.92572, 0.07428 and 92.57304. Similarly, to Fall detection, Network B scores extremely high in FP rate, which disqualifies it from being chosen as the best custom network. Network C scores the best FP rate of 7.39804 and the best precision of 0.9369. However, it has the worst recall, F1, TP rate, and miss rate.

4.3. Testing results

This advantage in the performance of pre-trained networks holds in the test data across the performance measures as seen in Table 4 for Fall. VGG-16 (Simonyan & Zisserman, 2015) is in the lead for recall, precision and F1, scoring 0.9948, 0.9846 and 0.9897. AlexNet (Krizhevsky et al., 2012) scored very similar results with a slight decrease, where it got 0.9871 for recall, 0.9796 for precision and 0.9834 for F1. Custom networks on test data shows a clear disadvantage for Network C on detecting falls, while Network A shows a more profound advantage on test data since it achieved the best result in most performance measures, specifically on recall having 0.9284 and F1 with 0.9153, leaving Network B to have a minor advantage of 0.0053 on precision scoring 0.9083 against 0.9026 for Network A precision.

Additionally, as it can be seen, Table 4 reports the performances of two relevant related works. Namely, the method by Qian et al. (2008) which uses shallow ML models and visual descriptors to classify video frames into Fall or Non-Fall classes, and the method by Doulamis (2016) which uses an end-to-end deep neural network to extract the features and map them into the pre-defined classes were overtaken by the proposed models with respect to all performance measures.

For test data in No-Fall, Table 5 shows pre-trained networks excel again with the lead of VGG-16 (Simonyan & Zisserman, 2015), achieving the best results in 0.9855 recall, 0.9951 in precision and 0.9903 F1 where it is plotted with the other networks results. AlexNet (Krizhevsky et al., 2012) follows VGG-16 (Simonyan & Zisserman, 2015) as in before

Table 1. Validation performance average of the 'Fall' and 'No-Fall' classes achieved using different implementations of Faster R-CNN.

Network	Accuracy	Recall	Precision	F1	MR	TP rate	FP rate
Network A	0.9199	0.91946	0.91932	0.91897	0.08055	91.9452	8.0548
Network B	0.89232	0.89021	0.89448	0.89096	0.10979	89.02058	10.97942
Network C	0.8926	0.8961	0.89176	0.89188	0.1039	89.60908	10.39092
VGG-16	**0.995**	**0.99466**	**0.99526**	**0.99494**	**0.00534**	**99.46672**	**0.53328**
AlexNet	0.991	0.99096	0.99091	0.99093	0.00904	99.09389	0.90611

Table 2. Validation performance of the 'Fall' class achieved using different implementations of Faster R-CNN.

Network	Recall	Precision	F1	MR	TP rate	FP rate
Network A	0.91816	0.90522	0.91118	0.08186	91.81496	7.92456
Network B	0.8547	0.90642	0.8789	0.1453	85.46812	7.42696
Network C	0.92602	0.84662	0.88404	0.07398	92.60196	13.3838
VGG-16	**0.99174**	**0.9974**	**0.99454**	**0.00826**	**99.17478**	**0.24134**
AlexNet	0.98992	0.99046	0.99018	0.01008	98.98938	0.8016

Table 3. Validation performance of the 'No-Fall' class achieved using different implementations of Faster R-CNN.

Network	Recall	Precision	F1	MR	TP rate	FP rate
Network A	0.92076	0.93342	0.92676	0.07924	92.07544	8.18504
Network B	0.92572	0.88254	0.90302	0.07428	92.57304	14.53188
Network C	0.86618	0.9369	0.89972	0.13382	86.6162	7.39804
VGG-16	**0.99758**	**0.99312**	**0.99534**	**0.00242**	**99.75866**	**0.82522**
AlexNet	0.992	0.99136	0.99168	0.008	99.1984	1.01062

Table 4. Fall results using different implementations of Faster R-CNN on testing data.

Network	Accuracy	Recall	Precision	F1	MR	TP rate	FP rate
Network A	0.9184	0.9284	0.9026	0.9153	0.0716	92.8401	9.0713
Network B	0.8325	0.7416	0.9083	0.8165	0.2584	74.1573	7.5614
Network C	0.8536	0.8796	0.8255	0.8517	0.1204	87.965	17.0341
VGG-16	**0.99**	**0.9948**	**0.9846**	**0.9897**	**0.0052**	**99.4819**	**1.4458**
AlexNet	0.984	0.9871	0.9796	0.9834	0.0129	98.7147	1.8913
Method in Doulamis (2016)	0.8511	0.8496	0.8066	0.8271	0.0961	85.965	17.233
Method in Qian et al. (2008)	0.8321	0.8675	0.8341	0.8503	0.1193	87.255	16.910

Table 5. No-Fall results using different implementations of Faster R-CNN on testing data.

Network	Accuracy	Recall	Precision	F1	MR	TP rate	FP rate
Network A	0.9184	0.9093	0.9335	0.9212	0.0907	90.9287	7.1599
Network B	0.8325	0.9244	0.7799	0.846	0.0756	92.4386	25.8427
Network C	0.8536	0.8297	0.8827	0.8554	0.1703	82.9659	12.035
VGG-16	**0.99**	**0.9855**	**0.9951**	**0.9903**	**0.0145**	**98.5542**	**0.5181**
AlexNet	0.984	0.9811	0.9881	0.9846	0.0189	98.1087	1.2853
Method in Doulamis (2016)	0.8511	0.8416	0.8171	0.8288	0.1204	86. 5	19.0341
Method in Qian et al. (2008)	0.8321	0.8511	0.8889	0.8517	0.1193	87.908	19.825

scoring a close 0.9811, 0.9881 and 0.9846 in recall, precision and F1. As for custom networks, Network A scored the best results in precision with 0.9335 and F1 with 0.9212 compared to 0.7799 precision and 0.846 F1 for Network B and 0.8827 in precision and 0.8554 in recall for Network C. Network B shows a very small advantage of 0.0151 over Network A in recall. Also, as one can see in Table 5, the proposed models outperforms the methods by Qian et al. (2008) and Doulamis (2016) with respect to the considered performance measures.

These results proved that the representation depth is beneficial for the fall detection accuracy, and that state-of-the-art performance can be achieved using pre-trained models which the architecture exhibits a substantially increased depth. Namely, VGG-16- and AlexNet-based models generalize well to the fall detection task and dataset. Moreover, they outperform more complex recognition pipelines built around less deep scene representations. This yet again confirms the importance of depth in visual representations. Besides, the dropout layers included in these architectures yield a structured model regularization that overcomes the over-fitting issue. Moreover, these dropout layers help learning more relevant and independent features by preventing the co-adaptations among the neurons that are randomly chosen.

The slightly better performance of VGG-16 compared to AlexNet can be attributed to the fact that VGG-16 exhibits a better feature learning ability than AlexNet because it is deeper than AlexNet. Specifically, it can capture sparser features compared to AlexNet.

4.4. Sample results

A sample of the detected frames is shown in Figure 7. In Figure 7, one can see the Fall detected frames for the five implementations of Faster R-CNN, the falls have been

Figure 7. Sample results of detected 'Fall' frames using different implementations of Faster R-CNN.

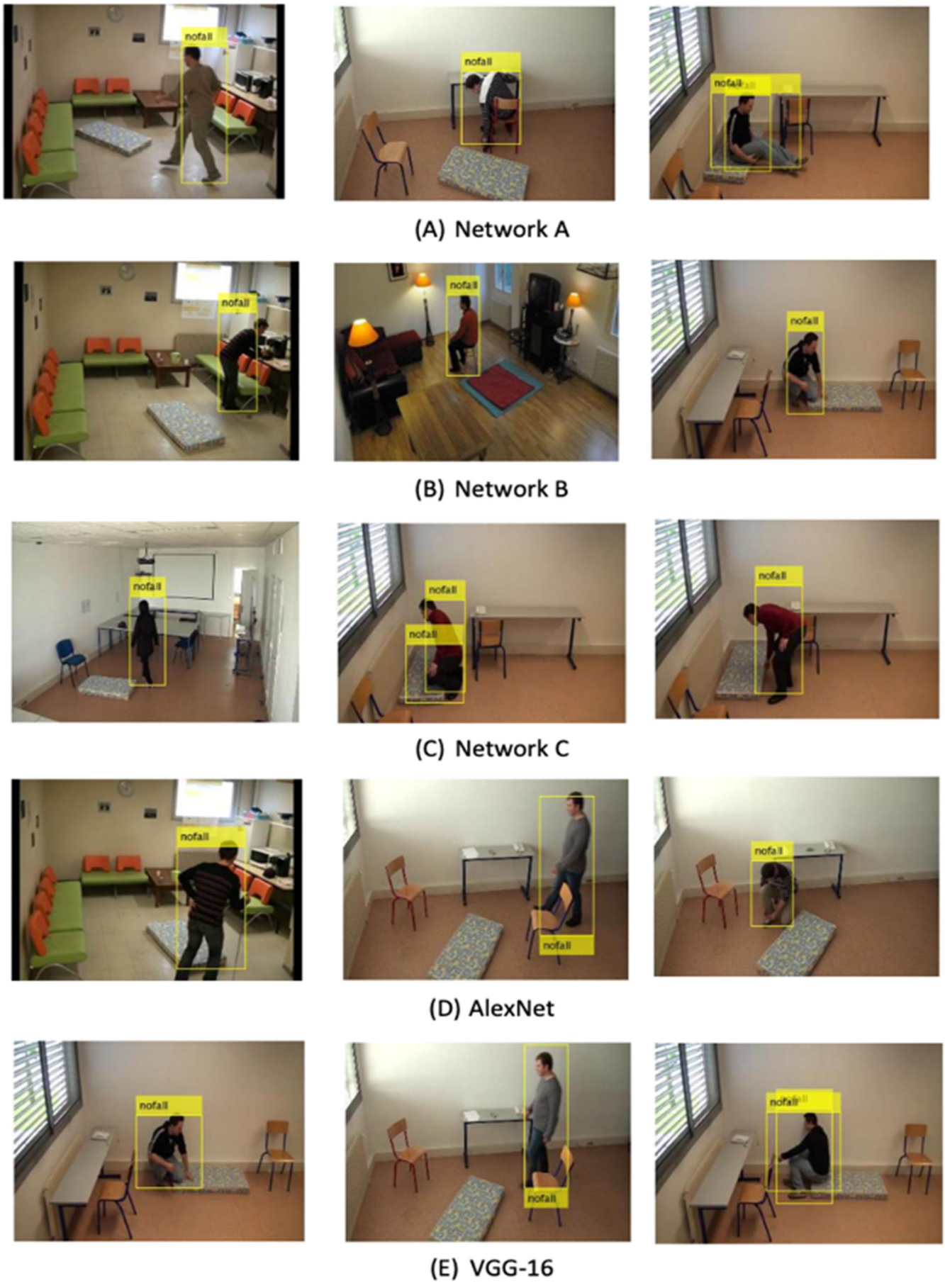

Figure 8. Sample results of detected 'No-Fall' frames using different implementations of Faster R-CNN.

detected in different stages of the fall, not only when the person is lying on the floor.

Moreover, in Figure 8, it can be seen that the detector was able to distinguish daily movements such as (sitting, bending, kneeling and walking) from fall and was able to label those movements as No-Fall movements.

5. Conclusions

In this work, we addressed the feature extraction problem along with the choice of the most appropriate classifier using CNN for a Fall detection problem. In a CNN, the most prominent features are learned over the several layers the network has which eliminate the difficulty of choosing the best feature for fall detection. We designed and develop a general framework in which video/image processing techniques are used along with deep learning to model the scenes containing fall incidents. In particular, we utilized a Faster R-CNN to recognize the Fall and No-Fall incidents. We achieved that by creating five implementations of CNN component in the Faster R-CNN; three of which were custom network architectures and two by using transfer learning with pre-trained VGG-16 (Simonyan & Zisserman, 2015) and AlexNet (Krizhevsky et al., 2012). The developed networks were able to differentiate between everyday movements that are similar to fall (such as sitting, bending, kneeling and walking), and actual falls. Using deep learning most used performance measures we were able to compare the performance of the different architectures we have implemented, where the pre-trained networks show superior performance over the custom networks achieving an accuracy of 0.99 for VGG-16 (Simonyan & Zisserman, 2015) and 0.984 for AlexNet (Krizhevsky et al., 2012) on test data. As for the custom networks, Network A shows the best performance in custom networks receiving 0.9184 in the accuracy of test data.

Acknowledgement

This work was supported by the Research Center of the College of Computer and Information Sciences at King Saud University. The authors are grateful for this support.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Research Center of the College of Computer and Information Sciences at King Saud University. The authors are grateful for this support.

References

Almegbel, F. Y., Alotaibi, I. M., Alhusain, F. A., Masuadi, E. M., Al Sulami, S. L., Aloushan, A. F., & Almuqbil, B. I. (2018). Period prevalence, risk factors and consequent injuries of falling among the Saudi elderly living in Riyadh, Saudi Arabia: A cross-sectional study. *BMJ Open*, *8*(1), e019063. https://doi.org/10.1136/bmjopen-2017-019063

Alrashed, A. M. (2017). Illustration of informal caregiving within Saudi society: demography, scope of care and enabling arrangements. *Scandinavian Journal of Caring Sciences*, *31*(2), 263–272. https://doi.org/10.1111/scs.12339

Anderson, D., Luke, R. H., Keller, J. M., Skubic, M., Rantz, M., & Aud, M. (2010). Linguistic summarization of video for fall detection using voxel person and fuzzy logic. *Computer Vision and Image Understanding*, *113*(1), 80–89. https://doi.org/10.1016/j.cviu.2008.07.006

Ben Ismail, M. M., & Bchir, O. (2017). Automatic fall detection using membership based histogram descriptors. *Journal of Computer Science and Technology*, *32*(2), 356–367. https://doi.org/10.1007/s11390-017-1725-z

Bobick, A. F., & Society, I. C. (2001). The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(3), 257–267.

Charfi, I., Miteran, J., Dubois, J., Atri, M., & Tourki, R. (2013). Optimised spatio-temporal descriptors for real-time fall detection: Comparison of SVM and Adaboost based classification. *Journal of Electronic Imaging*, *22*(4), 041106. https://doi.org/10.1117/1.JEI.22.4.041106

Cireşan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). *Flexible, high performance convolutional neural networks for image classification* [Paper presentation]. IJCAI International Joint Conference on Artificial Intelligence (pp. 1237–1242).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

Diraco, G., Leone, A., & Siciliano, P. (2010). *An active vision system for fall detection and posture recognition in elderly healthcare* [Paper presentation]. 2010 Design, Automation & Test in Europe Conference & Exhibition. (DATE 2010) (pp. 1536–1541).

Doulamis, N. (2016). *Vision based fall detector exploiting deep learning* [Paper presentation]. Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments (pp. 1–8).

Druzhkov, P. N., & Kustikova, V. D. (2016). A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, *26*(1), 9–15.

Egmont-Petersen, M., Ridder, D. D., & Handels, H. (2002). Image processing with neural networks – A review. *Pattern Recognition*, *35*(10), 2279–2301.

Fathima, A. A., Vaidehi, V., & Selvaraj, K. (2014). Fall detection with part-based approach for indoor environment. International *Journal of Intelligent Information Technologies*, *10*(4), 51–69.

Foroughi, H., Aski, B. S., & Pourreza, H. (2008). *Intelligent video surveillance for monitoring fall detection of elderly in home environments* [Paper presentation]. 2008 11th International Conference on Computer and Information Technology, no. ICCIT (pp. 219–224).

Foroughi, H., Naseri, a., Saberi, a., & Yazdi, H. S. (2008). *An eigen-space-based approach for human fall detection using integrated time motion image and neural network* [Paper presentation]. 2008 9th International Conference on Signal Processing (pp. 1499–1503).

Glorot, X., & Bordes, A. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, *15*, 315–323.

Hazelhoff, L., Han, J., & de With, P. H. N. (2008). *Video-based fall detection in the home using principal component analysis BT* [Paper presentation]. Proceedings of the 10th International Conference on Advanced concepts for intelligent vision systems (pp. 298–309).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Jackson, M., Nelson, D., & Stirk, S. (2005). Computer *vision in human–computer interaction* [Paper presentation]. ICCV 2005 Workshop on HCI Proceedings, Beijing, China.

Jokanovic, B., & Amin, M. (2018). Fall detection using deep learning in range-Doppler radars. *IEEE Transactions on Aerospace and Electronic Systems*, *54*(1), 180–189. https://doi.org/10.1109/TAES.2017.2740098

Jokanovic, B., Amin, M., & Ahmad, F. (2016). Radar fall motion detection using deep learning [Paper presentation]. 2016 IEEE Radar Conference (pp. 1–6).

Krizhevsky, B. A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems,25*(2),1–9.

LeCun, Y. (1997). Convolutional networks for images, speech, and time-series. The Handbook of brain theory and neural networks. MIT Press.

Lin, C. W., Ling, Z. H., Chang, Y. C., & Kuo, C. J. (2005). Compressed-domain fall incident detection for intelligent home surveillance [Paper presentation]. Proceedings of IEEE International Symposium on Circuits and Systems (pp. 3781–3784).

Liu, C.-L., Lee, C.-H., & Lin, P.-M. (2010). A fall detection system using K-Nearest Neighbor classifier. *Expert Systems with Applications*, *37*(10), 7174–7181. https://doi.org/10.1016/j.eswa.2010.04.014

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. Neurocomputing, *234*, 11–26. https://doi.org/10.1016/j.neucom.2016.12.038

Lu, N., Ren, X., Song, J., & Wu, Y. (2018). *Visual guided deep learning scheme for fall detection* [Paper presentation]. IEEE International Conference on Automation Science and Engineering (pp. 801–806).

Miaou, S. G., Sung, P. H., & Huang, C. Y. (2006). *A customized human fall detection system using omni-camera images and personal information* [Paper presentation]. Conference Proceedings – 1st Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare D2H2 2006 (pp. 39–42).

Mubashir, M., Shao, L., & Seed, L. (2013). A survey on fall detection: Principles and approaches. Neurocomputing, *100*, 144–152. https://doi.org/10.1016/j.neucom.2011.09.037

Murphy, K. (2012). *Machine learning: A probabilistic perspective*. The MIT Press.

Nait-Charif, H., & McKenna, S. J. (2004). *Activity summarisation and fall detection in a supportive home environment* [Paper presentation]. Proceedings of the 17th International Conference on Pattern Recognition (Vol. 4, pp. 323–326).

Qian, H., Mao, Y., Xiang, W., & Wang, Z. (2008). *Home environment fall detection system based on a cascaded multi-SVM classifier* [Paper presentation]. Proceedings of 10th International Conference on Control, Automation, Robototics and Vision (pp. 1567–1572).

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

Rougier, C., Meunier, J., St-Arnaud, A., & Rousseau, J. (2011). Robust video surveillance for fall detection based on human shape deformation. *IEEE Transactions on Circuits and Systems for Video Technology*, *21*(5), 611–622. https://doi.org/10.1109/TCSVT.2011.2129370

Simonyan, K., & Zisserman, A. (2015). *Very deep convolutional networks for large-scale image recognition* [Paper presentation]. ICLR.

Stevens, K. N., Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *I*, 21–27.

Vishwakarma, V., Mandal, C., & Sural, S. (2007). *Automatic detection of human fall in video BT* [Paper presentation]. Second international conference on Pattern Recognition and Machine Intelligence (pp. 616–623).

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1990). Phoneme recognition using time-delay neural networks. In A. Waibel & K.-F. Lee (Eds.), *Readings in speech recognition* (pp. 393–404). Morgan Kaufmann.

Yu, M., Rhuma, A., Naqvi, S. M., Wang, L., Member, S., & Chambers, J. (2012). A Posture recognition based fall detection system for monitoring an elderly person in a smart home environment . *IEEE Transactions on Infomation Technology in Biomedicine*, *16*(6), 1274–1286. https://doi.org/10.1109/TITB.2012.2214786

Yun, Y., & Gu, I. Y. H. (2016). Human fall detection in videos by fusing statistical features of shape and motion dynamics on Riemannian manifolds. Neurocomputing, *207*, 726–734. https://doi.org/10.1016/j.neucom.2016.05.058