symmetry

MDPI

Article

Symmetrically Stacked Long Short-Term Memory Networks for Fall Event Recognition Using Compact Convolutional Neural Networks-Based Tracker

Nur Ayuni Mohamed [1,2], Mohd Asyraf Zulkifley [1,*], Nor Azwan Mohamed Kamari [1] and Zulaikha Kadim [3]

[1] Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia; ayuni@siswa.ukm.edu.my (N.A.M.); azwank@ukm.edu.my (N.A.M.K.)
[2] Backend Industrial Engineering, Infineon Technologies (Malaysia) Sdn Bhd, Free Trade Zone, Batu Berendam 75710, Melaka, Malaysia
[3] Advanced Informatics Lab, MIMOS Berhad, Technology Park Malaysia, Kuala Lumpur 57000, Wilayah Persekutuan Kuala Lumpur, Malaysia; zulaikha.kadim@mimos.my
* Correspondence: asyraf.zulkifley@ukm.edu.my

Abstract: In recent years, the advancement of pattern recognition algorithms, specifically the deep learning-related techniques, have propelled a tremendous amount of researches in fall event recognition systems. It is important to detect a fall incident as early as possible, whereby a slight delay in providing immediate assistance can cause severe unrecoverable injuries. One of the main challenges in fall event recognition is the imbalanced training data between fall and no-fall events, where a real-life fall incident is a sporadic event that occurs infrequently. Most of the recent techniques produce a lot of false alarms, as it is hard to train them to cover a wide range of fall situations. Hence, this paper aims to detect the exact fall frame in a video sequence, as such it will not be dependent on the whole clip of the video sequence. Our proposed approach consists of a two-stage module where the first stage employs a compact convolutional neural network tracker to generate the object trajectory information. Features of interest will be sampled from the generated trajectory paths, which will be fed as the input to the second stage. The next stage network then models the temporal dependencies of the trajectory information using symmetrical Long Short-Term Memory (LSTM) architecture. This two-stage module is a novel approach as most of the techniques rely on the detection module rather than the tracking module. The simulation experiments were tested using Fall Detection Dataset (FDD). The proposed approach obtains an expected average overlap of 0.167, which is the best performance compared to Multi-Domain Network (MDNET) and Tree-structured Convolutional Neural Network (TCNN) trackers. Furthermore, the proposed 3-layers of stacked LSTM architecture also performs the best compared to the vanilla recurrent neural network and single-layer LSTM. This approach can be further improved if the tracker model is firstly pre-tuned in offline mode with respect to a specific type of object of interest, rather than a general object.

Keywords: fall event recognition; Compact Convolutional Neural Networks; Symmetrical Recurrent Neural Networks

Citation: Mohamed, N.A.; Zulkifley, M.A.; Kamari, N.A.M.; Kadim, Z. Symmetrically Stacked Long Short-Term Memory Networks for Fall Event Recognition Using Compact Convolutional Neural Networks-Based Tracker. *Symmetry* **2022**, *14*, 293. https://dx.doi.org/10.3390/sym14020293

Academic Editor: Jianfeng Ren

Received: 29 November 2021
Accepted: 28 January 2022
Published: 1 February 2022

1. Introduction

Nowadays, advancements in both machine learning and computer vision fields have propelled the number of studies in activity recognition applications. These studies are mostly geared towards intelligent systems that focus on monitoring and analysis of daily human activities. Furthermore, computational devices have become highly efficient and thus, contribute greatly to the development of many autonomous systems in activity recognition. In general, most of the studies have emphasized developing algorithms for classifying normal human activities, which is also known as Activities of Daily Living (ADL), such as sitting, walking, standing, and crouching down. Primarily, these algorithms

are also used in assessing the physical fitness and movement quality of an individual. Apart from that, the detection of abnormal activities is similarly important, specifically for safety and security-related reason. The inability to differentiate abnormal activities can cause many potential health risks, especially towards toddlers and elderly people. Therefore, early detection of abnormal activities is indispensable to prevent any bad consequence due to delay in detecting the event.

Fall event recognition is one of the most researched abnormal activities over the past few years [1,2]. The World Health Organization (WHO) has defined a fall event as a situation in which a person unintentionally lays down onto a lower surface [3]. It is usually unintentional, as such there will be a sudden change of body position from sitting or standing to a lower position [4,5]. Noury et al. [6] have categorized a fall event into four phases; pre-fall, critical, post-fall, and recovery, as illustrated in Figure 1. The pre-fall period is the phase in which a person is doing normal activities such as walking and sitting. The critical phase is the condition when there is a large change in body movements, directed towards a lower surface and ends with a vertical stop. Let us denote the start and end period of a fall event as T_0 and T_1. The inactivity period of a person after the fall is recognized as the post-fall phase. Lastly, the recovery phase is the condition when the person gets up again after the post-fall phase, where the timing is regarded as the response time, T_2. In general, the main cause for the occurrence of a fall event can be attributed to a loss of balance due to sudden trip or slip, or instability during movements [7,8]. Furthermore, WHO has also reported that the fall event is the second biggest death contributor globally with an estimate of 646,000 cases each year.

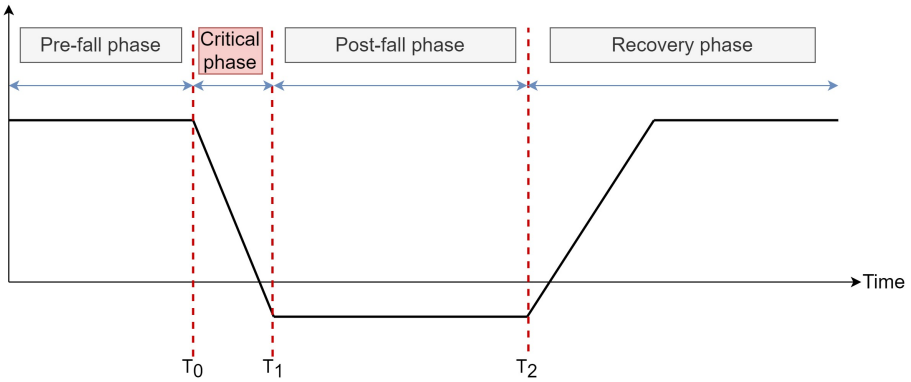

Figure 1. The four phases of a fall event.

Age and frailty level are the two most contributing factors for the fall event. Generally, elderly peoples tend to have weak muscles and they are prone to experience loss of balance easily. Thus, these factors will increase the occurrence probability of a falling-down situation. This statement is supported by statistical data from the United States National Institutes of Health (NIH), which has reported that about 1.6 million elderly peoples are involved in fall event injuries [9]. Moreover, individuals with low frailty levels, such as post-operative patients and people with mild disabilities, also belong to a high-risk group of fall event injuries. A possible consequence of a fall incident to these high-risk people is severe injury that leads to losing confidence, fear of falling, and loss of independence. Usually, a "long lie" situation, which is the total length of time of an individual to remain inactive lying on the floor after the incident, is a good indicator of the injury severity level. The long lie situation may cause loss of consciousness, hypothermia, internal bleeding, and dehydration [10]. In addition, the fall event can also cause death in some situations if early treatment is not administered [11]. Therefore, an early fall event recognition is very crucial and much needed to reduce the negative consequences and related injuries.

The state-of-the-art fall event recognition technology can be categorized into wearable device, ambient device, and vision-based system [12]. Generally, wearable devices utilize gyroscopes, barometric pressure sensors, and accelerometers to automatically detect a fall

event. It is known to have good accuracy, stability, and simplicity. Despite its efficacy, it requires an individual to wear the device for a prolonged time, which usually causes discomfort. Ambient devices make use of the infrared sensor, acoustic sensor, and piezoelectric sensor [13]. This method requires the installation of several ambient sensors at selected active regions to detect the occurrence of a fall event. However, ambient sensors are easily affected by noise, which then can trigger false fall event detection. Due to these circumstances, many studies have focused on vision-based systems to alleviate the aforementioned issues, as well as to allow fast and appropriate assistance. The vision-based systems rely on either one or more surveillance cameras to detect the occurrence of a fall event. The main advantages of a vision-based system are high flexibility, less intrusion, and it does not require other complementary sensors. In this paper, we only focus on a vision-based system for fall event recognition. Presently, most of the public facilities such as shopping malls, streets, as well as housing areas have been equipped with surveillance cameras like Closed Circuit Television (CCTV) or Internet Protocol (IP) cameras. This event detection has become indispensable in ensuring safety and providing security in public areas. More importantly, surveillance cameras can provide rich and useful information about the areas, not just solely for fall event detection. Moreover, recent advancements in camera technology and high-speed computer networks have made the system more affordable for commercial purposes.

In real-life situations, a fall event is classified as a sporadic event, in which it occurs infrequently, and thus the number of training data collected for a supervised system may not be optimal [14]. The imbalance in training data between fall and normal activities makes the development of a fall event detection system more challenging. Recently, Convolutional Neural Networks (CNNs/ConvNet) have gained tremendous achievements and attention, and have been widely used in various computer applications such as object detection [15,16], object tracking [17,18], image classification [19,20], image segmentation [21,22], machine translation [23,24], natural language processing [25,26], and physiotherapy monitoring [27]. This is because CNNs have strong capabilities in learning the object's features using multi-layer nonlinear transformations. Apart from that, a transfer learning method can be adopted to overcome the lack of training data by transferring a set of trained parameters from one model to a related task [28]. Besides, the top trackers of Visual Object Tracking (VOT) Challenge in 2015 and 2016, namely MDNET [29] and TCNN [30], have derived CNNs to represent and train the object appearance model, which has also achieved good results in tracking performance. Thus, this outstanding capability has motivated us to employ CNN-based tracker as a feature extractor that is robust to the challenges in fall event detection.

To the best of our knowledge, most of the existing fall event recognition methods only focus on classifying the presence of the fall event from normal activities throughout video sequences. However, our goal is to not only detect the fall event, but we aim to determine the exact instantaneous fall frame, $\text{fall}_{\text{detect}}$ in which the fall event occurs. Since time management is a vital issue in fall event detection, hence reducing the time between the occurrence of a fall event and the response time will reduce the negative consequences of fall-related injuries. Therefore, this paper introduces a two-stage fall event detection system. The first stage aims to detect the object and obtain the object's trajectories throughout the contiguous video frames by employing a fully CNN-based tracker. The instantaneous fall frame detection will be determined in the second stage by modeling the temporal dependencies between contiguous spatial coordinates using the symmetrically stacked Recurrent Neural Networks (RNN) with underlying Long Short-Term Memory (LSTM) networks. The symmetrical stacking of the networks has managed to create a more accurate deep network that relates the possibility of a fall event to the movement trajectories. The selection of the symmetrically stacked LSTM network over Vanilla RNN is due to its better capabilities in capturing time-series relationships and its ability to overcome the issue of vanishing gradients.

2. Related Works

Over the past few years, there is a large volume of published studies on the development of machine learning approaches to fall event recognition. The previous studies have reported that machine learning approaches can provide a non-intrusive approach and be less susceptible to noise. There are various different types of features extracted from video sequences, which are then used to train a classifier to detect the fall event. Generally, machine learning can be categorized into traditional and deep learning approaches. The traditional machine learning approaches include basic neural networks, support vector machine (SVM), and hidden Markov model (HMM), which all follow a shallow learning paradigm. Meanwhile, a deep learning approach exploits many numbers of hidden layers that are capable to learn object features and representations directly with little to no prior knowledge.

The earliest work of fall event recognition using neural networks has been introduced by Alhimale et al. [31]. Silhouette information is utilized by implementing median filter background subtraction to separate the foreground from background. Object's binary map is then fed to the neural networks and the bounding box aspect ratio is computed for fall event identification. Utilizing a similar approach, Hsu et al. [32] have introduced a Gaussian mixture model (GMM) background subtraction to obtain the foreground information. The bounding box aspect ratio, ellipse orientation, and vertical velocity of the object's center point are combined to train the neural networks for fall event detection. Fall event recognition based on SVM classification has been proposed in [33–36]. In [33], SVM is used for final fall event classification, which was trained by integrating Hu-moment and body posture information. The final fall event classification is determined if and only if there is a fall event, which is characterized by the velocity and changes in the bounding box aspect ratio. Similarly, authors in [34] have also extracted body posture information by considering acceleration as an additional feature to cater for the situation in which velocity changes are not able to provide clear speed differential during fall event movements. In [35], fall event detection is classified through SVM by taking the object features from the top-view depth images. They argued that the top-view depth images have less occlusion compared to the frontal-view images. Iazzi et al. [36] has implemented multi-class SVM, which is trained using vertical and horizontal histogram representations to classify a fall event from the confounding events such as bending, sitting, and lying. Unlike others, Zerrouki et al. [37] claimed that HMM can solve the classification problem better for sequential data. Therefore, they implemented HMM to discriminate fall events from other normal activities, which has resulted in reliable fall detection results. Meanwhile, Thuc et al. [38] have introduced two types of HMMs to model two different scenarios. The first HMM is implemented to differentiate fall-risk event from walking scenarios, while the second HMM is employed to determine the exact fall event based on the object shape and motion features.

Nowadays, many researchers have exploited the deep learning approach in fall event recognition to handle the limitations of traditional machine learning approaches. Li et al. [39] presented a fall event recognition method by applying CNNs to learn human shape deformation features in each video frame sequence. A similar approach has also been explored by works in [40,41]. However, silhouette information is obtained through background subtraction first, which is later used as the input to the CNNs architecture despite taking the whole frame information. In [42], human pose information obtained from the OpenPose algorithm is used to train CNNs to detect the fall event. The paper [43] has introduced two-stage training with the implementation of Principal Component Analysis Network (PCANet) as a feature extractor on colored image input. During the first-stage training, they have applied SVM to predict frame labels from each sub-video that contains walking, falling, and lying activities, while an SVM is re-applied as an action model to predict each sub-video label in the second-stage training. Conceptually, similar work has also been carried out by Wang et al. [44]. However, they extract silhouette information using Caffe framework and combined it with histograms of oriented gradients (HOG) and local binary pattern (LBP) to increase the fall event detection performances.

Apart from that, Marcos et al. [45] has incorporated CNNs with motion information, extracted from optical flow to detect a fall event. While, Haraldsson [46] has employed motion history images (MHI) to learn the temporal features, which is used to classify fall events via depthwise CNNs. Additionally, the same idea is used by Kong et al. [47], in which they have used three-stream CNNs in taking full benefit of object motion and appearance representations. The work by Abobakr et al. [48] has developed an end-to-end deep learning framework comprising of convolutional and RNN networks. They have applied ConvNet to analyze human body features for each frame sequence using depth images and then model the temporal information to recognize fall events using LSTM architecture. The paper by Anishchenko [49] has used a fixed second fully-connected (FC) with only two hidden nodes instead of following the exact AlexNet architecture. Interestingly, the modification has contributed to better Cohen's kappa measurement for fall and no-fall classification. Shojaei-Hashemi et al. [50] has introduced fall event recognition using the LSTM networks by employing the transfer learning approach to compensate for a small training data size of fall event. Firstly, the multi-class LSTM network is trained using a large number of normal human activities. Then, the learned weights are transferred so that it can be retrained for two-class LSTM for fall event recognition. Feng et al. [51] has introduced a fall event recognition in complex scenes using object detection approach. Firstly, the You Only Look Once (YOLO) v3 architecture is used to detect the object and the tracking process is performed by implementing a Deep-Sort tracking method. Next, a CNN architecture is used to extract the object features for each trajectory, which is later fed to the LSTM network to classify the fall event.

Nonetheless, the implementation of CNNs in object tracking application is less exploited, despite fruitful advantages in other computer vision domains. The difficulty in fitting the CNNs with limited training data is the major challenge to implement CNNs in object tracking purpose. This is due to the only information such as position and size of tracked object can be extracted in the first frame for a model-free tracker setting. Another drawback of CNNs is a longer training time which makes it impractical to be used in an online learning-based model update. However, the top trackers of 2015 VOT Challenge, namely TCNN and MDNET trackers have designed compact CNNs in its tracker architecture. These trackers were based on model-free tracker setting which trained and updated its object appearance model by using information that was only supplied during the first frame. Generally, these trackers shared a similar tracking fundamental, but differed a lot in handling the FC models. The TCNN tracker proposed a tree-structured to handle the FC nodes by retaining the parent node and deleting the oldest node once a new child node was spawned out. The TCNN tracker updated its FC models using the positive training data that were selected from the top matched object appearance model. However, MDNET tracker updated its FC layers by dividing them into several domains that were correspond to the most recent training samples. Each domain was trained separately and then categorized into either tracked object or background.

3. Methods

In this paper, we propose an instantaneous fall event recognition method which consists of two main modules; (1) object detection and (2) fall frame event recognition as outlined in Figure 2. The main purpose of the first module is to detect and track the object of interest throughout the video sequence, in which a fully CNN-based tracker is employed. The tracker base follows a similar network architecture as used in the Multiple Model Convolutional Neural Network (MMCNN) tracker [52]. Whereas, the second module will determine the exact instantaneous fall frame, $\text{fall}_{\text{detect}}$ by utilizing the LSTM network, which has been trained using the trajectories information, obtained from the first module.

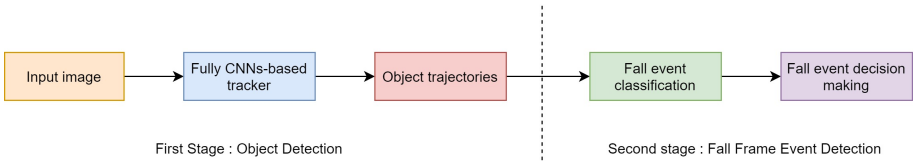

Figure 2. Pipeline of the proposed approach.

3.1. Dataset

The Fall Detection Dataset (FDD) [53] is used to conduct the experiments. The dataset consists of 124 annotated RGB videos comprising 94 falls and 30 ADLs videos in two different simulated situations, termed as the "Coffee Room" and "Home" situation as shown in Figure 3. This dataset contains acting events, in which the actor has purposely fall under various fall angles; forward, backward, and lateral. It also includes fall events from different starting posture positions such as standing and sitting. Nonetheless, the ADLs videos are excluded in this experiment since our main goal is only to determine the fall event occurrence. Each video is recorded using a single camera with a frame size of 320×240 pixels and a frame rate of 25 frames/s.

Figure 3. Sample of images from FDD dataset with "Coffee Room" scene shown at the top row and "Home" scene shown at the bottom row for three different types of fall event.

3.2. Object Detection

3.2.1. Tracker Architecture

In this section, the head region is selected as the region of interest that will be tracked, instead of the whole body. Figure 4 depicts eight consecutive samples of the head region during two different fall event situations. The base MMCNN tracker employed in this paper consists of three convolutional layers and three FC layers as illustrated in Figure 5. However, the FC layers' configurations are modified to produce a compact tracker architecture to be implemented for object tracking purposes. The Table 1 summarizes the full configurations of MMCNN tracker architecture. In this work, pre-trained VGG-M weights and biases that have been trained on ImageNet database [54] are imported as feature extractor and they will not be retrained due to the limited number of training data. Meanwhile, the weights and biases for FC layers are randomly initialized and trained using softmax cross-entropy loss function. All RGB input images will be resized to 75×75 pixels before they are fed to the first CNN layer. The first, second, and third CNN layers will have filter sizes of 96, 256, and 512, respectively. Local response normalization and maximum down-pooling operations are applied to the first and second layers of CNN. The down-pooling kernel size is 3 by 3 with a stride size of 2. The output of the third CNN layer will be flattened out into a vector $\mathcal{R}^{512}$ before being passed to the dense classification layers. The number of hidden nodes for the first and second FC layers are fixed to 512, while the last layer will be classified into two output classes, either the tracked object or background information.

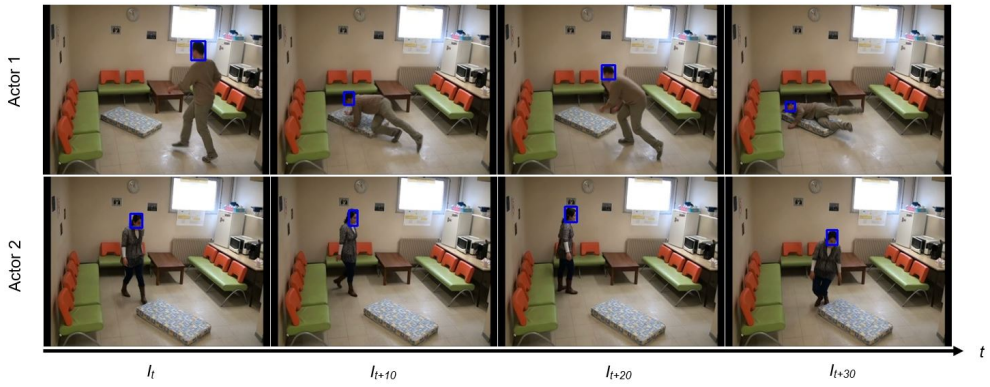

Figure 4. Head regions during two different fall event situations.

Table 1. The full configurations of the proposed tracker architecture.

Layer	Filter Size	Stride	Padding	Output	Activation Function
Conv1	7×7	2	0	$96 \times 35 \times 35$	ReLU
Pool1	3×3	2	0	$96 \times 17 \times 17$	-
Conv2	5×5	2	0	$256 \times 7 \times 7$	ReLU
Pool2	3×3	2	0	$256 \times 3 \times 3$	-
Conv3	3×3	1	0	$512 \times 1 \times 1$	ReLU
FC1	-	-	-	512	ReLU
FC2	-	-	-	512	ReLU
FC3	-	-	-	2	Softmax

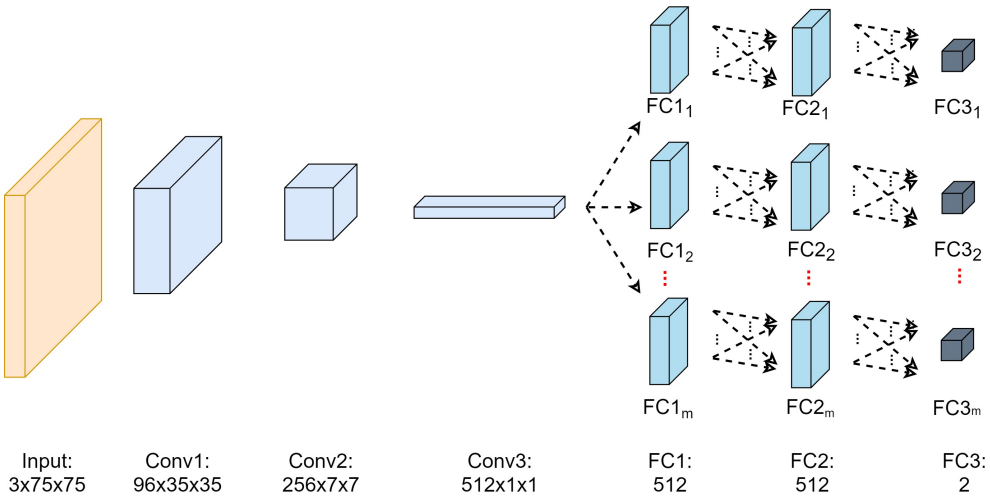

Figure 5. Architecture of the MMCNN tracker.

Additionally, the MMCNN tracker retains a compact number of object appearance models that are symmetrically aligned for the FC layers' configurations. The illustration of symmetrical multiple object appearance models is depicted as in Figure 5, with m refers as the total number of object appearance models. The MMCNN maintains m models for the FC layers with the direction to cater for occlusion problems and help to re-detect the tracked object if it has been occluded for a short period of time. Maintaining several m models is necessary since the object appearance model will be different before and during the occlusion. For example, if only a single object appearance is maintained, the model will have higher probability to be updated with the occlusion information. Contrarily, if several object appearance models are maintained, the first model might be updated with occlusion information while, the rest of the models will remain unchanged. Then, the rest of the models will be a good reference point for the model updates after the occlusion is over. Simultaneously, this approach helps to reduce model drifting during tracking process if multiple object appearance models are retained.

Generally, the handling of multiple object appearance models for the FC layers of MMCNN shares similar fundamental with TCNN and MDNET trackers. However, there are some improvements that have been proposed. The TCNN handles the FC nodes in a tree-structure approach by retaining the parent node and deleting the oldest FC nodes once the new child node is born. The tree-structure approach helps to update the FC nodes using the top matched appearance model. Similarly, the MMCCN tracker adopts the tree-structure approach in handling the FC nodes. However, the MMCNN tracker will retain a set of diverse FC nodes instead of just deleting the oldest model. Figure 6 represents the comparison of the tree-structure approach for TCNN and MMCNN tracker in updating the object appearance model with n represents the FC nodes. For a new model update in MMCNN tracker, a parent node will be replaced by a newly spawned child node according to the similarity score that is calculated during the model update. Hence, the most similar node will be replaced and updated with the new information.

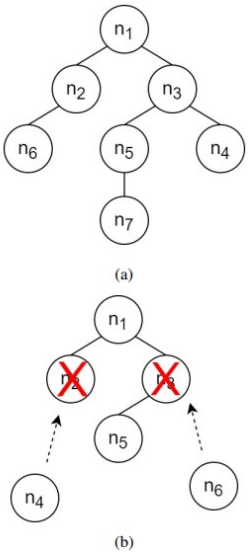

Figure 6. Comparison of tree-structure approach for (**a**) TCNN and (**b**) MMCNN trackers.

Even though we have employed MMCNN tracker in this manuscript, however, there are some different approaches that have been utilized. The approaches are listed as follow:

1. Implementation using RGB images instead of thermal infrared (TIR) images.
2. The head is used as the region of interest (ROI) instead of the whole body.
3. The sampling candidates' scheme for the training data.

Generally, the MMCNN is developed for the object tracking purposes using the TIR images. However, the remarkable improvement of the MMCNN tracker with regards to the handling of multiple object appearance models have motivated us to adopt the MMCNN tracker in our experiment. Moreover, the head region is used as the ROI instead of the whole body as implemented in MMCNN's experiments. The noticeable drawback of the FDD dataset is that there are situations in which the tracked object is near to the camera placement that result in large size of ROI if the whole body is used as the ROI. Hence, to alleviate this situation, we have selected the head as the ROI with the assumption that the head size will not differ much as the person moves across the video. Additionally, the MMCNN tracker have extracted the sampling candidates based on the Gaussian distribution using the last known tracked object's position. This situation can cause the background information is sampled far from the last position. However, we have proposed a different sampling candidates' scheme in which the sampling candidates are constrained by a specific rule where we employed a translation strategy in both foreground and background sampling candidates' extraction. This is to ensure that all sampling candidates are sampled at and nearby the last known position as well as to ensure that all sampling candidates bounding

boxes are generated to a fixed size similar to the first frame information. This is crucial to mitigate the issue of different ROI regions that exists in the FDD dataset.

3.2.2. Sampling Generation

The tracker is trained according to binary classifier problem with the object of interest is considered as one class, and the background is considered to be the other class. A set of sparse sampling data, $\boldsymbol{B}_{train} = \boldsymbol{B}_{train,+ve} \bigcup \boldsymbol{B}_{train,-ve}$, is generated, which consists of positive, $\boldsymbol{B}_{train,+ve}$ and negative, $\boldsymbol{B}_{train,-ve}$ training bounding boxes to represent the foreground and background regions, respectively, as shown in Figure 7. For initialization, only the first frame information will be used to train the FC model. While for model update, which is training a new FC model, the samples are collected from a fixed frame interval. The number of samples is also lesser compared to the initialization stage where the samples are generated based on the tracked output, which does not represent the object appearance perfectly. Hence, it becomes a weak supervision method, where it is normal for the model to experience model drift after a certain period of tracking. To reduce uncertainty in sampling generation, each frame will contribute equally in case of tracking drift in some of the frames.

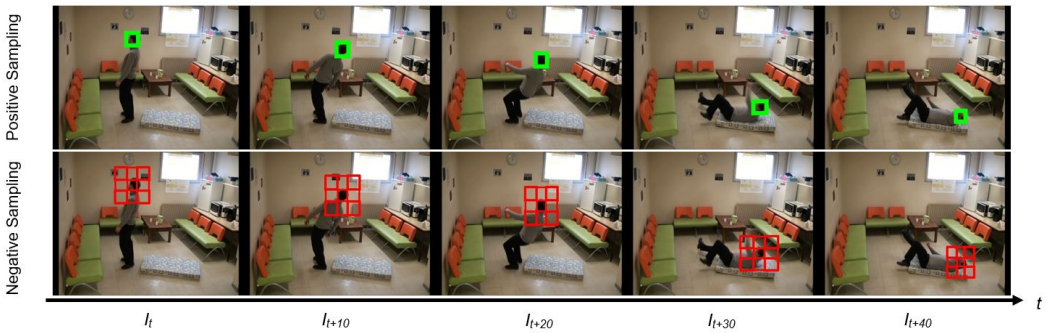

Figure 7. Sampling candidates with positive and negative samples are shown at the top and bottom rows, respectively.

Let t represents the current frame, a set of patches of $\boldsymbol{B}^t_{train,+ve}$ and $\boldsymbol{B}^t_{train,-ve}$ are sampled with regards to the given first frame ground truth bounding box, $\boldsymbol{B}^t_{gt} = [x^t, y^t, w^t, h^t] \in \mathbb{R}^4$ as in Equations (1) and (2), where s_{+ve} and s_{-ve} represent the total number of positive and negative patches, respectively.

$$\boldsymbol{B}^t_{train,+ve} = \left\{ b^1_{+ve}, \ldots, b^{s_{+ve}}_{+ve} \right\} \tag{1}$$

$$\boldsymbol{B}^t_{train,-ve} = \left\{ b^1_{-ve}, \ldots, b^{s_{-ve}}_{-ve} \right\} \tag{2}$$

The $\boldsymbol{B}^t_{train,+ve}$ bounding boxes are generated by sampling a set of foreground regions that are similar to $\boldsymbol{B}^t_{gt}$ with a translation stride threshold, $\rho = 15$. On the other hand, $\boldsymbol{B}^t_{train,-ve}$ bounding boxes are generated by sampling a set of background regions that are closed to the $\boldsymbol{B}^t_{gt}$ with each patch, b_{-ve} will follow the rule in Equation (3), where x_i and y_i represent the top-left coordinates, w, and h represent the width and height of each patch, respectively. Then, both $\boldsymbol{B}^t_{train,+ve}$ and $\boldsymbol{B}^t_{train,-ve}$ will be resized to 75×75 pixels to fit the required input size of the first CNN layer.

$$\begin{aligned} & b_{-ve,i} = [x_i, y_i, w, h], \quad b_{i,-ve} \notin B^i_{gt} \\ & \text{with} \quad i = \{1, \ldots, s_{-ve}\} \end{aligned} \tag{3}$$

With the assumption that a head size will not differ much as the person moves across the video, all generated samples will have a fixed size similar to the first frame information. The candidate bounding boxes will be generated according to the last known previous

location, which is the tracker output of the previous frame. For $\boldsymbol{B}^t_{train,+ve}$, the samples generated need to fulfill the condition of intersection over union (IoU) rule with regards to the previous tracker output. The IoU needs to be at least 0.85, which means that the object moves between the two frames will not be too large. While, for $\boldsymbol{B}^t_{train,-ve}$, the IoU must be less than 0.15, so that they represent the background image that surrounds the tracked object. When the minimum IoU rule is enforced, the generated $\boldsymbol{B}^t_{train,-ve}$ will not be too far from the tracked object, which will help in training the new FC model.

3.2.3. Tracker Model Update

Since the object's appearance will change as it moves, the tracker must be updated periodically to avoid model drift issue. A set of $\boldsymbol{B}_{train,+ve}$ and $\boldsymbol{B}_{train,-ve}$ are collected using Gaussian distribution with an intersection over union (IoU) limiting factor, c_{+ve} and c_{-ve}, which are set to 0.7 and 0.3, respectively, with $\sigma = 3$. These limits are enforced so that both training bounding boxes are still sampled around the last updated position given the fact that it is a model-free tracker with no consequence frames $\boldsymbol{B}_{gt}$ information. Both $\boldsymbol{B}_{train,+ve}$ and $\boldsymbol{B}_{train,-ve}$ will be accumulated throughout r recent frames.

The FC model with the highest score will be selected among the m models, which will be retrained using similar $\boldsymbol{B}_{train,+ve}$ and $\boldsymbol{B}_{train,-ve}$. However, the FC models with the highest similarity score will be replaced by a newly trained FC model. The FC model will be trained for a maximum number of 50 epochs using softmax cross-entropy loss function and Adam optimizer with a fixed learning rate of 0.001. Contrary to the previous methods [29,30], an old model is not deleted as the object might revisit the scene once again or the object might strike the same pose again. However, the most similar model will be updated and spawned as a new node. This scheme allows the tracker to maintain a diverse set of appearance models, as such any similar model will have limited weights towards the final decision.

$$\begin{aligned} \boldsymbol{B}^t_{train,+ve} \sim \quad & \mathcal{N}(\boldsymbol{B}^t_{gt}, \sigma) \\ \text{s.t.} \quad & \text{IoU}_i > c_{+ve}, \quad i = \{1, \ldots, s_{+ve}\} \end{aligned} \tag{4}$$

$$\begin{aligned} \boldsymbol{B}^t_{train,-ve} \sim \quad & \mathcal{N}(\boldsymbol{B}^t_{gt}, \sigma) \\ \text{s.t.} \quad & \text{IoU}_i < c_{-ve}, \quad i = \{1, \ldots, s_{-ve}\} \end{aligned} \tag{5}$$

3.3. Fall Frame Event Recognition

The implementation of fall frame event detection using the LSTM network is explained in this section. The LSTM network is employed to incorporate temporal information in detecting a fall event. The tracked object trajectories from the first stage are used to train the LSTM network in distinguishing between fall and no-fall features. Then, the final features are smoothed out by a fall event decision-making module to identify the exact frame, in which the fall event has occurred.

3.3.1. Fall Event Classification

The proposed LSTM network will follow a standard k stacked LSTM layers as depicted in Figure 8 with $h = 32$ number of hidden layers. Tracked object trajectories that were obtained in the first stage are fed to the LSTM network to classify the fall from no-fall event features with $s_n = [x_n, y_n] \in \mathbb{R}^{n \times d \times 1}$ denotes the spatial coordinates at each time steps, n, where d is the total number of input. s_n will follow weighted components approach with weight, $W_s \in \mathbb{R}^{h \times d}$, which will be randomly initialized. The LSTM is used to relate the outputs from several previous time steps $(n-1, n-2, \ldots, 0)$. Hence, at the current time step, the input s_n is mapped to the current output, y_n based on the previous hidden states, $(h_{n-1}, h_{n-2}, \ldots, 0)$. h_n, which is the current hidden states will be forwarded to the next s_{n+1} to estimate the new output features y_{n+1}. A softmax cross-entropy loss function is applied so that the last output becomes a probability distribution $P(h_n)$ with $d \in \mathbb{R}^2 = \{\text{fall}, \text{no-fall}\}$ classes. A five-fold cross-validation procedure is applied to test performance of the LSTM network. The training data will be divided into five-fold, in

which four out of the five folds will be used to train the LSTM network, while the remaining fold is used for testing purpose. These procedures are repeated five times, whereby each fold will be tested independently.

Four different features are used as the input, s_n to train the networks as summarized in Table 2. The F1 features include only the (x_n, y_n) trajectory coordinates whereas, the F2 features are also based on the trajectory coordinates but with the additional dropout layer with channel probability of 0.5. Dropout layers are included with the intention of reducing the over-fitting problem during the training phase. The F3 features are $(x_n, y_{normalize,n})$ trajectory coordinates, where $y_{normalize,n}$ is obtained by normalizing each y_n with regards to the first frame input, y_0 as in Equation (6). The motivation behind this normalization layer is to figure out the significant features, where a fall event should results in a large change of y-coordinates during the critical phase. Figure 9 depicts the comparison of the pixel position between y_n and $y_{normalize,n}$. Similarly, the F4 features implement the same features as in F3, but with the addition of dropout layers.

$$y_{n,i} = y_i - y_0, \quad i = \{1, \ldots, V\} \tag{6}$$

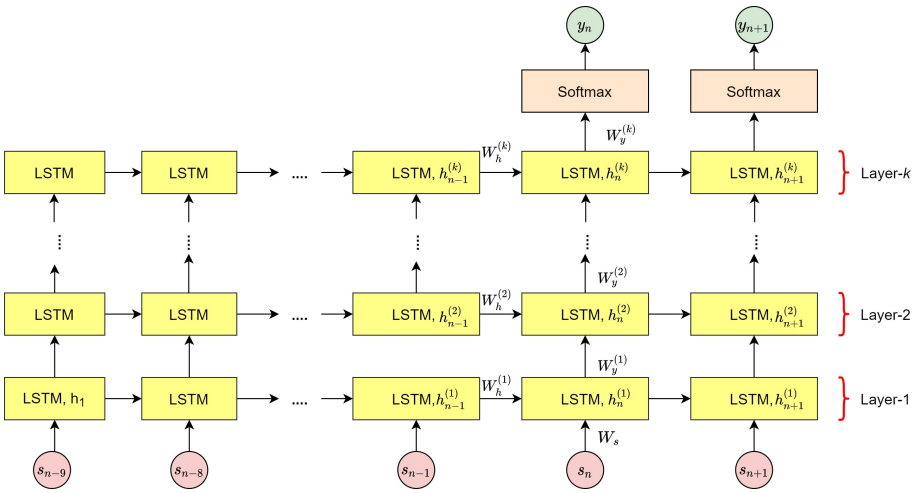

Figure 8. Stacked LSTM architecture.

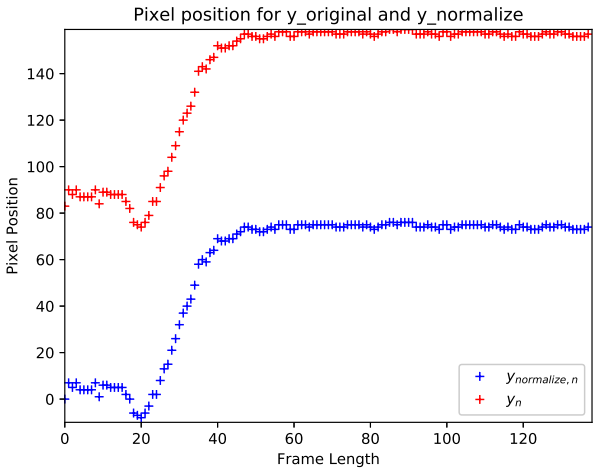

Figure 9. Comparison graph of y_n and $y_{normalize,n}$.

Table 2. Description of features used to train RNN architecture.

No.	Feature	Description
1	F1	x_n,y_n
2	F2	x_n,y_n + dropout
3	F3	x_n,$y_{normalize,n}$
4	F4	x_n,$y_{normalize,n}$ + dropout

3.3.2. Fall Event Decision-Making

The goal of this subsection is to make a final decision on locating the exact fall frame using previous fall event classification features. Let i represents the current frame, $\Delta y^{best} \in \mathbb{R}^1$ is calculated based on Equation (8), where y_i is the current and y_{i-1} is the previous output scores in a video sequence of V length. Δy_i is measured since the output scores of fall event classification is in d classes, in which the fall frame is detected when the change in output scores is the biggest. The exact fall frame, $\text{fall}_{\text{detect}}$ is finally calculated as in Equation (9).

$$\Delta y_i = y_i - y_{i-1}, \quad i = \{1, \ldots, V - n\} \tag{7}$$

$$\Delta y^{best} = arg\max(|\Delta y_i|) \tag{8}$$

$$fall_{detect} = \Delta y^{best} + n \tag{9}$$

3.4. Performance Evaluation

Four Visual Object Tracking (VOT) evaluation metrics are used to quantify the performance of the trackers that include accuracy (Ac), robustness (Ro), reliability (Re), and expected area overlap (EAO). The VOT protocol follows a reset-based procedure, in which a tracking failure, F is triggered whenever a tracker predicts no overlapping areas ($IoU = 0$) between the tracker bounding box, B^t and ground truth bounding box, B^t_{gt} in a video sequence of V length as ruled in Equation (11). Figure 10 illustrates an example of the IoU for both bounding boxes. The VOT protocol also requires the tracker to re-initialize five frames after each failure. Ac measures the average IoU between the B^t and B^t_{gt} bounding boxes, while Ro captures the number of F. Re measures the likelihood of successful tracking after Q frames which have been fixed to 100. Lastly, EAO averages the IoU over a range of frames between upper limit, lim_{up} and lower limit, lim_{low}. The EAO is introduced to rank the tracker by considering a trade-off between Ac and Ro. Meanwhile mean error, (*Mean Err*) is used to quantify the performance of the proposed fall frame event detection. *Mean Err* measures the mean error between the detected fall frame, $\text{fall}_{\text{detect}}$ and the provided ground truth frame, fall_{gt}.

$$Ac = \frac{B^t \bigcap B^t_{gt}}{B^t \bigcup B^t_{gt}} \tag{10}$$

$$Ro = \sum_{i=1}^{V} F_i, \quad F_i = \begin{cases} 1, & \text{if } IoU_i = 0 \\ 0, & \text{if } IoU_i > 0 \end{cases} \tag{11}$$

$$Re = e^{-Q\frac{Ro}{V}} \tag{12}$$

$$EAO = \frac{1}{lim^{up} - lim^{low}} \sum_{i=lim^{low}}^{lim_{up}} IoU_i \tag{13}$$

$$Mean\ Err = \left|fall_{detect} - fall_{gt}\right| \tag{14}$$

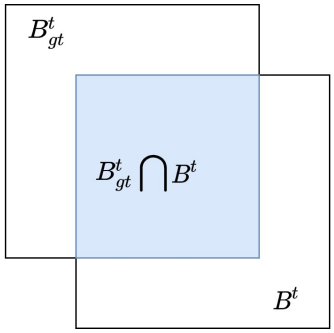

Figure 10. Illustration of IoU for both B^t and B^t_{gt} bounding boxes.

4. Results and Discussion

4.1. Tracker Initialization

The MMCNN, TCNN, and MDNET trackers have been configured as model-free tracker as such the object appearance model will be learned using the first frame ground truth bounding box, $\boldsymbol{B}_{gt}$. The FC layers will be trained using softmax cross-entropy loss function and Adam optimizer with a learning rate of 0.0005. The training cycle is set to a maximum of 50 epochs with a mini-batch sampling size of 128. An equal number of positive, $\boldsymbol{B}_{train,+ve}$ and negative, $\boldsymbol{B}_{train,-ve}$ training samples are used to train the trackers. A similar set of $\boldsymbol{B}_{train,+ve}$ and $\boldsymbol{B}_{train,-ve}$ training samples are also used to initialize all m object appearance models for the MMCNN tracker, in case of occlusion that happens in the early part of the video. Both $\boldsymbol{B}_{train,+ve}$ and $\boldsymbol{B}_{train,-ve}$ will then be sampled with the limiting factor of IoU in the model update process for all the trackers. Let c_{+ve} and c_{-ve} represent the IoU threshold for the positive and negative training data, while N_{+ve} and N_{-ve} indicates the positive and negative sampling data for the model update process. Similarly, the LSTM network is trained using softmax cross-entropy loss function and Adam optimizer, but with the learning rate of $1e^{-6}$ and mini-batch sampling size of 32 for a maximum number of 1000 epochs. A ratio of 1:2 of positive, $\boldsymbol{S}_{train,+ve}$ versus negative, $\boldsymbol{S}_{train,-ve}$ training data is used to train the LSTM network. Table 3 summarizes the list of parameters involved.

Table 3. List of parameters.

Parameter	Value
$\boldsymbol{B}_{train,+ve}$	400
$\boldsymbol{B}_{train,+ve}$	400
m	3
c_{+ve}	0.7
c_{ve}	0.3
N_{+ve}	50
N_{-ve}	200
$\boldsymbol{S}_{train,+ve}$	6200
$\boldsymbol{S}_{train,-ve}$	9400

The selection of all parameters has been meticulously performed using greedy optimization technique. Three significant parameters have been tested that includes the optimization technique, learning rate values, and the size of mini-batch for both MMCNN and LSTM networks. These parameters are denoted as the essential parameters that need to be accentuated during the training process. The first greedy optimization approach tests various types of optimizers: Adam, Stochastic Gradient Descent (SGD), RMSProp, and AdaDelta, in which Adam optimization has been selected as the best optimization technique. The next greedy optimization covers the selection of learning rate values. Through the testing, MMCNN tracker and LSTM networks work well with 0.0005 and $1e^{-6}$, respectively. Lastly, the greedy optimization is used to find the suitable size of mini-batch with

mini-batch sets to 128 and 32 sampling data are the best values for MMCNN and LSTM networks, respectively.

4.2. Tracking Performance Comparison of the Fully CNN-Based trackers

This subsection aims to compare and measure the performance of three CNN-based trackers, which are MMCNN, TCNN and MDNET. Furthermore, each tracker will also be tested by considering two difficulty attributes, which are background clutter and occlusion to represent real-life challenges. Firstly, Table 4 shows the full results of each tracker performance without considering any attribute-based video. The MMCNN tracker performs the best with the highest *EAO* of 0.167, followed by MDNET and TCNN with 0.158 and 0.145, respectively. Even though the MMCNN tracker has the lowest average accuracy, which is 0.442, it is more reliable with performance values of 0.855. The tracker is able to track the object with the fewest number of track failures. Undeniably, average accuracy will drop over a period of time due to model appearance drift and thus, the trade-off between accuracy and track failure needs to be observed. Contrarily, TCNN has the highest robustness with 1.000, followed by MDNET with 0.812. Their higher accuracy values are caused by frequent re-initialization procedures, but the *EAO* performance will be lower, as it neglects the accuracy after the first track failure. Figure 11 shows some output samples for all tested trackers.

Table 4. Performance comparison of *Ac*, *Ro*, *Re*, and *EAO* results of all three tested trackers without considering any attribute-based video.

Method	*EAO*	*Ac*	*Ro*	*Re*
MMCNN	0.167	0.442	0.449	0.855
MDNET	0.158	0.508	0.812	0.738
TCNN	0.145	0.494	1.000	0.692

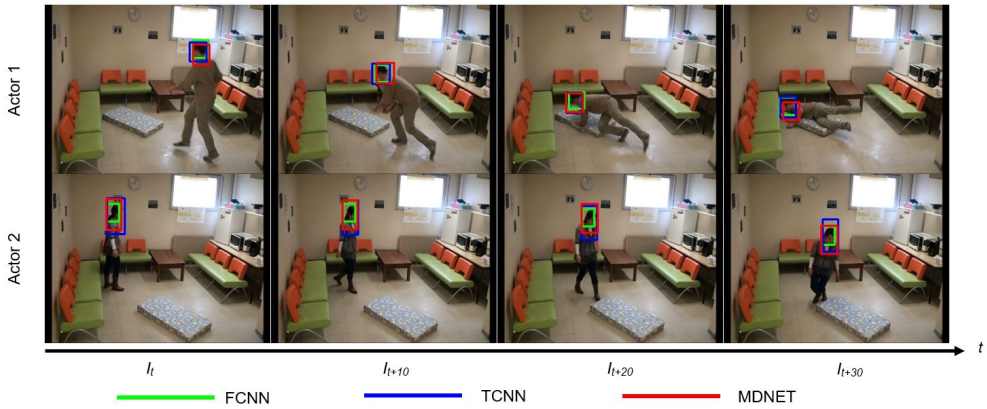

Figure 11. Samples of tracking output of fully CNN-based trackers.

Background clutter is a situation, in which the background that surrounds the object of interest has similar appearance information to the foreground as shown in Figure 12a. Table 5 shows the tracking results of all three trackers with MMCNN still achieves the highest *EAO* of 0.274. Surprisingly, TCNN achieves a higher *EAO* of 0.229 compared to MDNET for this test case, which is the opposite of the normal test case. MMCNN and TCNN performances, which are better in this cluttered case can be attributed to training data that considers the parent node information. In MDNET, the new FC node will be updated solely based on recent accumulated samples and thus if the scene is cluttered, the samples will be degenerated in value. However, the failure rates for TCNN and MDNET are still high with 1.200 and 1.800, respectively. This indicates that both trackers cannot locate the object as good as MMCNN.

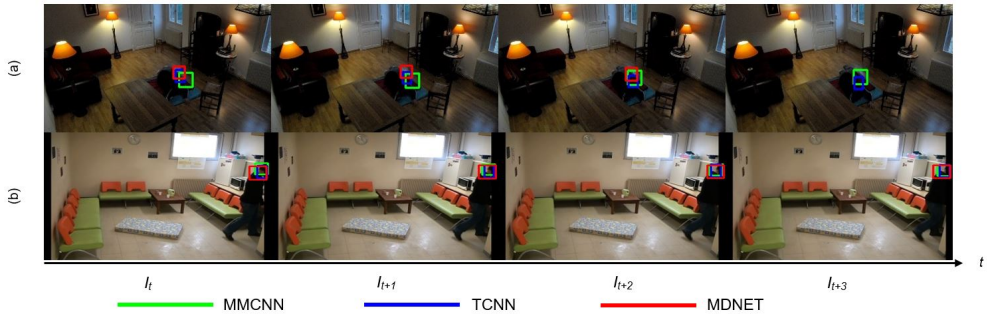

Figure 12. Samples of tracking output according to the attribute-based (**a**) background clutter (**b**) occlusion.

Table 5. Performance comparison of *Ac*, *Ro*, *Re*, and *EAO* results of the three tested trackers for the background clutter scenario.

Method	*EAO*	*Ac*	*Ro*	*Re*
MMCNN	0.274	0.431	1.000	0.626
TCNN	0.229	0.395	1.200	0.557
MDNET	0.214	0.471	1.800	0.449

Occlusion is one the challenging aspect in visual object tracking, as such the object of interest is either partially or fully occluded as illustrated in Figure 13b. Occlusion affects the hardest for a tracker which follows track-by-detection philosophy as appearance information is severely limited. Table 6 shows that MMCNN is the top among all trackers with the highest *EAO* of 0.204 and robustness of 0.583. The *EAO* of MDNET is slightly lower than MMCNN with 0.202 followed by TCNN with 0.175. Furthermore, MMCNN has the smallest total number of tracking failure which is triggered when IOU is less than zero as shown in Table 7. Both MDNET and TCNN trackers have a total of 13 tracking failures, which contribute to the high robustness value of 1.083. On the other hand, the number of tracking failures for MMCNN is around half of the MDNET and TCNN and thus, it is more suitable for the implementation in autonomous fall event detection.

Table 6. Performance comparison of *Ac*, *Ro*, *Re*, and *EAO* results of three trackers for the occlusion case.

Method	*EAO*	*Ac*	*Ro*	*Re*
MMCNN	0.204	0.392	0.583	0.865
MDNET	0.202	0.473	1.083	0.706
TCNN	0.175	0.459	1.083	0.783

Table 7. Comparison of the total number of failure ($Iou < 0$) of three tested trackers for the occlusion scenario.

Method	MMCNN	MDNET	TCNN
Failure	7	13	13

4.3. Fall Frame Event Recognition Comparison

This subsection is dedicated to validate our proposed approach to recognize the exact fall frame event, $\text{fall}_{\text{detect}}$. The validation process will be done by comparing two different Recurrent Neural Networks (RNN) models, which are Vanilla RNN and LSTM. The $\text{fall}_{\text{detect}}$ is determined according to Δy^{best}, extracted from the sequences of Δy_i. Figure 13 depicts an example of how Δy^{best} and mean error are obtained. In short, the mean error is the average difference between the detected fall frame and its ground truth value. Meanwhile, Table 8 shows the comparison of the mean error performance of three network models. For

Vanilla RNN, it achieves 36 mean error frames for both F1 and F2 features, followed by F4 and F3 with 37 frames and 43 frames, respectively. Note that, F3 features do not have a large impact and significant in training all network models for fall event detection. For a single-layer LSTM case, F1 features produce the lowest mean error of 32 frames, followed by F4 features. However, there is only a slight difference between the mean error of F2 and F3 features. It is interesting to note that a single-layer LSTM model performs better with the addition of dropout layers compared to the Vanilla RNN.

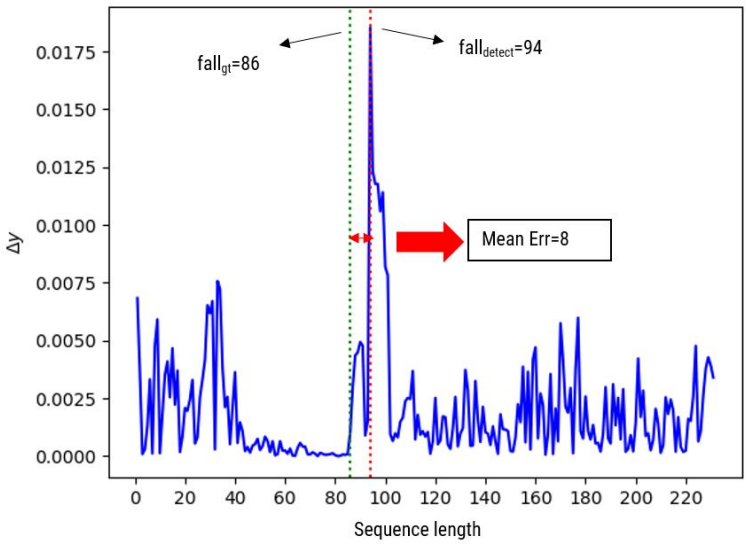

Figure 13. Illustration of the Δy^{best} and mean error.

Furthermore, additional experiments have also been performed using various layer configurations of the Stacked LSTM. F1 features work well on 2 layers of stacked LSTM with a mean error of 29 frames. However, a further increment in the number of layers will also increase the mean error value. For F2 features, it performs well with both 3 and 4 layers of stacked LSTM and manages to obtain a mean error of 28 frames. Similarly, the number of mean error also increases accordingly to the increasing number of layers for the F3 features. Surprisingly, F4 features provide the most prominent representation with the lowest number of mean error of 22 frames for 3 layers of stacked LSTM. However, the same pattern is observed when further testing is done with an additional number of layers, where the mean error is also increasing. The 22 frames of mean error are still tolerable as it produces ≈1 s delay time for a real-time application, given 30 frames per second video data. Hence, Stacked LSTM performs the best compared to the Vanilla RNN and Single LSTM networks. This is because the hierarchy of the hidden layers enables the network to represent more complex representations to capture information at different scales.

Table 8. Mean error value for Vanilla RNN, Single-layer LSTM and Stacked LSTM.

No.	Features	Vanilla RNN	Single LSTM	Stacked LSTM					
				2-Layer	3-Layer	4-Layer	5-Layer	6-Layer	7-Layer
1	F1	36	32	29	33	32	35	30	44
2	F2	36	35	33	28	28	31	41	43
3	F3	43	36	26	29	30	32	36	37
4	F4	37	33	26	22	32	37	37	39

5. Conclusions

In this work, we have presented a method to detect instantaneously the frame, in which the fall event has occurred. Our core approach relies on deep learning representations,

where a fully CNNs tracker is employed to detect and track the object of interest in a video sequence. Furthermore, we have presented a symmetrically stacked LSTM approach to identify the fall from no-fall features. The model learns the temporal information and able to exploit the significant features, specifically the rich amount of information from human motion. A larger movement difference towards the ground level in the vertical axis indicates a higher possibility of fall event occurrence. Simulation experiments, tested on the FDD dataset, have shown that the proposed approach outperforms the state-of-the-art method in terms of frame mean error and suitable for real-time application. In future work, depth image will be exploited to further relate the temporal information for a more robust fall event detection system.

Author Contributions: Conceptualization, N.A.M. and M.A.Z.; methodology, N.A.M.; software, N.A.M. and M.A.Z.; validation, N.A.M.; formal analysis, N.A.M. and M.A.Z.; investigation, N.A.M. and M.A.Z.; writing—original draft preparation, N.A.M., M.A.Z., N.A.M.K. and Z.K.; writing—review and editing, N.A.M., M.A.Z., N.A.M.K. and Z.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by Ministry of Higher Education Malaysia under Grant FRGS/1/2019/ICT02/UKM/02/1 and in part by Universiti Kebangsaan Malaysia under Grant GUP-2019-008.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The orginal dataset can be downloaded from http://le2i.cnrs.fr/Fall-detection-Dataset?lang=en (accessed on 1 November 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADL	Activities of Daily Living
CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Networks

References

1. Khan, S.S.; Hoey, J. Review of fall detection techniques: A data availability perspective. *Med. Eng. Phys.* **2017**, *39*, 12–22. [CrossRef] [PubMed]
2. Khan, S. Classification and Decision-Theoretic Framework for Detecting and Reporting Unseen Falls. Ph.D. Thesis, University of Waterloo, Waterloo, ON, USA, 2016.
3. Ahmed, M.; Mehmood, N.; Nadeem, A.; Mehmood, A.; Rizwan, K. Fall detection system for the elderly based on the classification of shimmer sensor prototype data. *Healthc. Inform. Res.* **2017**, *23*, 147–158. [CrossRef] [PubMed]
4. Makhlouf, A.; Boudouane, I.; Saadia, N.; Cherif, A.R. Ambient assistance service for fall and heart problem detection. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 1527–1546. [CrossRef]
5. Fortino, G.; Gravina, R. Fall-MobileGuard: A smart real-time fall detection system. In Proceedings of the International Conference on Body Area Networks, Sydney, Australia, 28–30 September 2015; pp. 44–50.
6. Noury, N.; Rumeau, P.; Bourke, A.; ÓLaighin, G.; Lundy, J. A proposal for the classification and evaluation of fall detectors. *Innov. Res. BioMed. Eng.* **2008**, *29*, 340–349. [CrossRef]
7. Igual, R.; Medrano, C.; Plaza, I. Challenges, issues and trends in fall detection systems. *Biomed. Eng. Online* **2013**, *12*, 1–24. [CrossRef]
8. Mohamed, N.A.; Zulkifley, M.A. Moving object detection via TV-L1 optical flow in fall-down videos. *Bull. Electr. Eng. Inform.* **2019**, *8*, 839–846. [CrossRef]
9. Yang, L.; Ren, Y.; Hu, H.; Tian, B. New fast fall detection method based on spatio-temporal context tracking of head by using depth images. *Sensors* **2015**, *15*, 23004–23019. [CrossRef]
10. Bhandari, S.; Babar, N.; Gupta, P.; Shah, N.; Pujari, S. A novel approach for fall detection in home environment. In Proceedings of the Global Conference on Consumer Electronics (GCCE), Nagoya, Japan, 24–27 October 2017; pp. 1–5.

11. Khel, M.A.B.; Ali, M. Technical Analysis of Fall Detection Techniques. In Proceedings of the International Conference on Advancements in Computational Sciences (ICACS), Lahore, Pakistan, 18–20 February 2019; pp. 1–8.
12. Ge, C.; Gu, I. Y. H.; Yang, J. Human fall detection using segment-level CNN features and sparse dictionary learning. In Proceedings of the International Workshop on Machine Learning for Signal Processing (MLSP), Tokyo, Japan, 25–28 September 2017; pp. 1–6.
13. Stone, E.E.; Skubic, M. Fall detection in homes of older adults using the Microsoft Kinect. *IEEE J. Biomed. Health Inform.* **2014**, *19*, 290–301. [CrossRef]
14. Mohamed, N.A.; Zulkifley, M.A.; Ibrahim, A.A.; Aouache, M. Optimal Training Configurations of a CNN-LSTM-Based Tracker for a Fall Frame Detection System. *Sensors* **2021**, *21*, 6485. [CrossRef]
15. Ouyang, W.; Wang, X.; Zeng, X.; Qiu, S.; Luo, P.; Tian, Y.; Li, H.; Yang, S.; Wang, Z.; Loy, C.-C.; et al. Deepid-net: Deformable deep convolutional neural networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2403–2412.
16. Ouyang, W.; Wang, X. Joint deep learning for pedestrian detection. In Proceedings of the IEEE International Conference on Computer Vision, Portland, OR, USA, 23–28 June 2013; pp. 2056–2063.
17. Zulkifley, M.A. Two streams multiple-model object tracker for thermal infrared video. *IEEE Access* **2019**, *7*, 32383–32392. [CrossRef]
18. Mohamed, N.A.; Zulkifley, M.A.; Kamari, N.A.M. Convolutional Neural Networks Tracker with Deterministic Sampling for Sudden Fall Detection. In Proceedings of the International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 7 October 2019; pp. 1–5.
19. Shaikh, S.H.; Saeed, K.; Chaki, N. *Moving Object Detection Approaches, Challenges and Object Tracking*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 5–14.
20. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
21. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*; Springer: Cham, Switzerland, 2015; pp. 234–241.
22. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
23. Zoph, B.; Yuret, D.; May, J.; Knight, K. Transfer learning for low-resource neural machine translation. *arXiv* **2016**, arXiv:1604.02201.
24. Zhou, J.; Cao, Y.; Wang, X.; Li, P.; Xu, W. Deep recurrent models with fast-forward connections for neural machine translation. *Trans. Assoc. Comput. Linguist.* **2016**, *4*, 371–383. [CrossRef]
25. Conneau, A.; Schwenk, H.; Barrault, L.; Lecun, Y. Very deep convolutional networks for natural language processing. *arXiv* **2016**, arXiv:1606.01781.
26. Zhang, J.; Zong, C. Deep learning for natural language processing. In *Deep Learning: Fundamentals, Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2019 ; pp. 111–138.
27. Zulkifley, M.A.; Mohamed, N.A.; Zulkifley, N.H. Squat angle assessment through tracking body movements. *IEEE Access* **2019**, *7*, 48635–48644. [CrossRef]
28. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
29. Nam, H.; Han, B. Learning multi-domain convolutional neural networks for visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NE, USA, 27–30 June 2016; pp. 4293–4302.
30. Nam, H.; Baek, M.; Han, B. Modeling and propagating cnns in a tree structure for visual tracking. *arXiv* **2016**, arXiv:1608.07242.
31. Alhimale, L.; Zedan, H.; Al-Bayatti, A. The implementation of an intelligent and video-based fall detection system using a neural network. *Appl. Soft Comput.* **2014**, *18*, 59–69. [CrossRef]
32. Hsu, Y.-W.; Perng, J.-W.; Liu, H.-L. Development of a vision based pedestrian fall detection system with back propagation neural network. In Proceedings of the International Symposium on System Integration (SII), Nagoya, Japan, 11–13 December 2015; pp. 433–437.
33. Wang, R.-D.; Zhang, Y.-L.; Dong, L.-P.; Lu, J.-W.; Zhang, Z.-Q.; He, X. Fall detection algorithm for the elderly based on human characteristic matrix and SVM. In Proceedings of the International Conference on Control, Automation and Systems (ICCAS), Busan, Korea, 13–16 October 2015; pp. 1190–1195.
34. Mohd, M.N.H.; Nizam, Y.; Suhaila, S.; Jamil, M.M.A. An optimized low computational algorithm for human fall detection from depth images based on Support Vector Machine classification. In Proceedings of the IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuching, Malaysia, 12–14 September 2017; pp. 407–412.
35. Kasturi, S.; Jo, K.-H. Classification of human fall in top Viewed kinect depth images using binary support vector machine. In Proceedings of the International Conference on Human System Interactions (HSI), Ulsan, Korea, 17–19 July 2017; pp. 144–147.
36. Iazzi, A.; Rziza, M.; Thami, R.O.H. Fall detection based on posture analysis and support vector machine. In Proceedings of the International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, 21–24 March 2018; pp. 1–6.
37. Zerrouki, N.; Houacine, A. Combined curvelets and hidden Markov models for human fall detection. *Multimed. Tools Appl.* **2018**, *77*, 6405–6424. [CrossRef]
38. Thuc, H.L.U.; Van Tuan, P.; Hwang, J.-N. An effective video-based model for fall monitoring of the elderly. In Proceedings of the International Conference on System Science and Engineering (ICSSE), Ho Chi Minh City, Vietnam, 21–23 July 2017; pp. 48–52.

39. Li, X.; Pang, T.; Liu, W.; Wang, T. Fall detection for elderly person care using convolutional neural networks. In Proceedings of the International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Shanghai, China, 14–16 October 2017; pp. 1–6.
40. Adhikari, K.; Bouchachia, H.; Nait-Charif, H. Activity recognition for indoor fall detection using convolutional neural network. In Proceedings of the International Conference on Machine Vision Applications (MVA), Nagoya, Japan, 8–12 May 2017; pp. 81–84.
41. Yu, M.; Gong, L.; Kollias, S. Computer vision based fall detection by a convolutional neural network. In Proceedings of the International Conference on Multimodal Interaction, Glasgow, UK, 13–17 November 2017; pp. 416–420.
42. Huang, Z.; Liu, Y.; Fang, Y.; Horn, B.K. Video-based fall detection for seniors with human pose estimation. In Proceedings of the International Conference on Universal Village (UV), Boston, MA, USA, 21–24 October 2018; pp. 1–4.
43. Wang, S.; Chen, L.; Zhou, Z.; Sun, X.; Dong, J. Human fall detection in surveillance video based on PCANet. *Multimed. Tools Appl.* **2016**, *75*, 11603–11613. [CrossRef]
44. Wang, K.; Cao, G.; Meng, D.; Chen, W.; Cao, W. Automatic fall detection of human in video using combination of features. In Proceedings of the International Conference on Bioinformatics and Biomedicine (BIBM), Shenzhen, China, 15–18 December 2016; pp. 1228–1233.
45. Núñez-Marcos, A.; Azkune, G.; Arganda-Carreras, I. Vision-based fall detection with convolutional neural networks. In *Wireless Communications and Mobile Computing*; Hindawi: London, UK, 2017 ; pp. 1–16.
46. Haraldsson, T. Real-time Vision-based Fall Detection: With Motion History Images and Convolutional Neural Networks. Master's Thesis, Luleå University of Technology, Luleå, Sweden, 2018.
47. Kong, Y.; Huang, J.; Huang, S.; Wei, Z.; Wang, S. Learning spatiotemporal representations for human fall detection in surveillance video. *J. Vis. Commun. Image Represent.* **2019**, *59*, 215–230. [CrossRef]
48. Abobakr, A.; Hossny, M.; Abdelkader, H.; Nahavandi, S. Rgb-d fall detection via deep residual convolutional lstm networks. In Proceedings of the Digital Image Computing: Techniques and Applications (DICTA), Canberra, Australia, 10–13 December 2018; pp. 1–7.
49. Anishchenko, L. Machine learning in video surveillance for fall detection. In Proceedings of the Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT), Yekaterinburg, Russia, 7–8 May 2018; pp. 99–102.
50. Shojaei-Hashemi, A.; Nasiopoulos, P.; Little, J.J.; Pourazad, M.T. Video-based human fall detection in smart homes using deep learning. In Proceedings of the International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018; pp. 1–5.
51. Feng, Q.; Gao, C.; Wang, L.; Zhao, Y.; Song, T.; Li, Q. Spatio-temporal fall event detection in complex scenes using attention guided LSTM. *Pattern Recognit. Lett.* **2020**, *130*, 242–249. [CrossRef]
52. Zulkifley, M.A.; Trigoni, N. Multiple-model fully convolutional neural networks for single object tracking on thermal infrared video. *IEEE Access* **2018**, *6*, 42790–42799. [CrossRef]
53. Charfi, I.; Miteran, J.; Dubois, J.; Atri, M.; Tourki, R. Optimized spatio-temporal descriptors for real-time fall detection: Comparison of support vector machine and Adaboost-based classification. *J. Electron. Imaging Int. Soc. Opt. Photonics* **2013**, *22*, 041106. [CrossRef]
54. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]