Noname manuscript No.
(will be inserted by the editor)

DeepFall – Non-invasive Fall Detection with Deep Spatio-Temporal Convolutional Autoencoders

Jacob Nogas, Shehroz S. Khan, Alex Mihailidis

the date of receipt and acceptance should be inserted later

Abstract Human falls rarely occur; however, detecting falls is very important from the health and safety perspective. Due to the rarity of falls, it is difficult to employ supervised classification techniques to detect them. Moreover, in these highly skewed situations it is also difficult to extract domain specific features to identify falls. In this paper, we present a novel framework, *DeepFall*, which formulates the fall detection problem as an anomaly detection problem. The *DeepFall* framework presents the novel use of deep spatio-temporal convolutional autoencoders to learn spatial and temporal features from normal activities using non-invasive sensing modalities. We also present a new anomaly scoring method that combines the reconstruction score of frames across a video sequences to detect unseen falls. We tested the *DeepFall* framework on three publicly available datasets collected through non-invasive sensing modalities, thermal camera and depth cameras and show superior results in comparison to traditional autoencoder and convolutional autoencoder methods to identify unseen falls.

Keywords fall detection · convolutional autoencoders · spatio-temporal · anomaly detection

1 Introduction

Each year, one out of five falls that older adults incur, causes a serious injury to them such as broken bones or a head injury [4]. Among nursing home residents, on an average, 2.6 falls per person per year occur [25]. Detecting falls is important from the perspective of health and safety; however, due to the infrequent occurrence of falls, it is difficult to collect sufficient training data for them [15]. Given this lack of training data, there may be no or

Jacob Nogas
University of Toronto, Canada
E-mail: jacob.nogas@mail.utoronto.ca

Shehroz S. Khan
Toronto Rehabilitation Institute, University Health Network, Canada
E-mail: shehroz.khan@uhn.ca

Alex Mihailidis
University of Toronto, Canada
E-mail: alex.mihailidis@utoronto.ca

little training data for falls, and it will hard to employ supervised classification techniques. There are also concerns that the sensing devices for the task of fall detection may be invasive and breach the privacy of the person [19,32]. Some non-invasive sensing devices, such as a thermal or depth camera, may not fully reveal the identity of a person, but it is difficult to extract discriminative features to identify unseen falls, especially in a highly skewed data scenario [27].

The research work discussed in this paper deals with identifying falls in video sequences captured from non-invasive sensing modalities, such as thermal and depth cameras, which may fully or partially protect the privacy of a person. We propose to address the problem of fall detection as an anomaly detection problem by considering the lack of fall data and the abundance of normal activities of daily living (ADL). The general idea explored in the paper is to train an autoencoder on normal activities and use their reconstruction error to identify unseen falls during testing. A deep autoencoder (DAE) can be used to learn features from normal ADL; however, it ignores the 2D structure of images, and may force each feature to be global to span the entire visual field [18]. In visual recognition tasks, convolutional autoencoders (CAE) perform better because they can discover localized spatial features that repeat themselves over the input [18]. A video sequence embeds information in both space and time; therefore, spatio-temporal convolutional autoencoders are more relevant as they can learn a representation of local spatio-temporal patterns of frames in a video [1].

In this paper, we present a novel fall detection framework, *DeepFall*, which comprises of (i) formulating fall detection as an anomaly detection problem, (ii) designing a deep spatio-temporal convolutional autoencoder (DSTCAE) and training it on *only* the normal ADL, and (iii) proposing a new anomaly score to detect unseen falls. The DSTCAE first encodes stacks of contiguous frames in a time window by performing 3D convolution/3D pooling. During the decoding phase, the DSTCAE uses 3D UpSampling, or 3D deconvolution to reconstruct the input window of frames. Reconstruction error for each frame within a window is then calculated. We then present a new method to compute anomaly score, termed as within-context score, which considers the reconstruction error of frames within a window and gives an anomaly score of a given window. The anomaly score can be used to identify an unseen fall during the testing phase of DSTCAE. To the best of our knowledge, this is the first time a DSTCAE has been used for fall detection problem. The present paper is an extension of our previous work [22], and differs from it in the following manner:

- In the previous work, we employed Convolutional LSTM based Autoencoder (CLSTMAE). Whereas, in this paper we use a different 3D Convolutional autoencoder, DSTCAE, for learning spatio-temporal features from the normal ADL. Our results show that DSTCAE gives better results than CLSTM for all the datasets.
- The previous paper only tested CLSTM on thermal dataset, whereas in this paper we extend our experiments on two more depth camera based fall detection datasets.
- The previous paper presented a frame based anomaly score. However, in this paper we present a new anomaly scoring method that takes the decision at the level of a window (comprising of contiguous frames) and not individual frames, which resulted in similar or higher area under the curve (AUC) across all the datasets.
- In this paper, we did an additional experiment to understand the impact on AUC of designating a window as a fall by increasing the number of fall frames in it from 1 to the maximum size of the window. This experiment provided insights on choosing appropriate fall frames in a window before identifying it as a fall window using a given anomaly score.

– In the present paper, we compare CLSTM with three variants of DSTCAE for all the three datasets, and also added new results from one-class nearest neighbours on the features learned through the DAE.

2 Related Work

To the best of our knowledge, fall detection has not been addressed as a anomaly detection problem using unsupervised deep learning methods. Therefore, in this section, we present a review of literature on video anomaly detection, which is closer in concept to the work presented in this paper.

Ribeiro et a. [24] propose to use CAE for detecting anomalies in videos. They extract appearance and motion features from video frames, and combine them to present different scenarios to the CAE. They train the CAE on only normal frames and used a regularized reconstruction error as a score to identify normal and anomalous frames. They extend their work by using reconstruction error from the CAE trained on normal video frames as an input to a one-class SVM and showed similar results [10]. Tran and Hogg [28] use CAE and one-class SVM for video anomaly detection. They first extract foreground optical flow patches using an empirical threshold, then train a convolutional Winner-Takes-All autoencoder on these patches, which learn hierarchical sparse representations in an unsupervised manner. After training the model, sparse features are extracted and fed to a one-class SVM to learn the concept of normality. Their method shows competitive performance on two datasets in comparison to existing state-of-the-art methods. Chalapathy et al. [5] present a robust CAE that considers anomalies in the data and modifies the objective function for training the model. They use the reconstruction error as a score and comment that their method is not over-sensitive to anomalies; can discover subtle anomalies and can be potentially deployed in live settings. They show their results on a video activity detection dataset and other image datasets. Hasan et al. [11] use both hand-crafted spatio-temporal features and fully convolutional feed-forward autoencoder to learn regular motion patterns in videos. They introduce a regularity score that scales the reconstruction error of a frame between 0 and 1. They show competitive performance of their method to other state-of-the-art anomaly detection methods. Munawar et al. [20] present a method to generate unbiased features by unsupervised learning for finding anomalies in industrial robots surveillance task. They cluster the input image data based on different image cues (such as color and gradient) to generate pseudo-class labels. Then, they simultaneously train the network on different pseudo-class labels to learn neutral feature representation using convolution neural networks. Finally, they use a deep long short term memory based recurrent neural network to predict the next video frame in the learned feature space. If it deviates significantly from the observed frame, it is identified as an anomaly in time and space. A fusion of appearance as well as motion features is used by [31] to detect anomalies in video. More precisely, stacked denoising DAE networks are applied separately to optical flow, and raw image patches. Anomalies in video are then detected in a one-class SVM framework.

Zhao et al. [33] present a spatio-temporal autoencoder that uses 3D convolutions to extract spatio-temporal features from videos. They further introduce a a weight-decreasing prediction loss for generating future frames to improve the learning of motion feature in videos. They tested their method on a dataset comprising a set of real-world traffic surveillance videos along with other standard anomaly detection datasets and show superior performance on state-of-the-art approaches. Penttilä [23] propose to use 3D CAE to learn spatial and temporal features to find anomalies in hyperspectral data. After training the network, they extract feature maps from the data and feed to standard anomaly detection algorithms. A two stage cascade of autoencoders is used by [26] to detect anomalies in video. In particular, spatio-temporal patches are fed to a 3D autoencoder, where regions of interest are detected. The proposed regions are then fed to a 3D convolutional neural network. The layers of the cascaded deep networks are designed as single-class Gaussian classifiers, which are used to detect anomalies in the video.

Chong and Tay [7] present a method to detect anomalies in videos that consists of a spatial feature extractor and a temporal encoder-decoder framework. The spatial feature extractor comprises of convolutional and deconvolutional layers, whereas the temporal encoder-decoder is a three-layer convolutional long short term memory model. This model will be referred to as CLSTMAE. Their model is trained only on the videos with normal scenes. They use a regularity score [11] to identify anomalies. They show comparable performance in comparison to other standard methods; however, it may produce more false alarms. Nogas et a. [22] apply the CLSTMAE model presented in [7], in conjunction with a novel scoring method, to the task of fall detection. The CLSTM is trained on videos of ADL only , and falls are detected as an anomaly. They show superior performance of CLSTMAE in comparison to DAE and CAE on a dataset captured using thermal camera.

The literature review suggests that spatio-temporal CAE are a good candidate to learn spatial and temporal features from normal activities and identify anomalies in videos. Most of the reviewed methods either set a threshold on the reconstruction error or feed the reconstruction error to other classifiers to identify anomalies. None of the reviewed methods are used for fall detection problem. In *DeepFall*, we train a DSTCAE only on normal ADL, and use a new anomaly scoring methods to identify a fall as anomaly from non-invasive sensors with superior performance.

In the next section, we introduce the different components of the *DeepFall* framework, which comprises of a DSTCAE and anomaly scoring methods (after formulating fall detection in an anomaly detection setting).

3 Deep Spatio-Temporal Convolutional Autoencoders

Traditional autoencoders consist of fully connected layers [16]. When passing an image to such a network, it is flattened into a 1D vector of pixels, and each pixel is connected to the units of the next layer. This results in a large number of network parameters, and fails to make use of the 2D local structure in an image [18]. A CAE exploits this 2D spatial structure, by sharing a set of weights across the image, resulting in far fewer parameters than a traditional autoencoder, and allowing the extraction of local features from neighbouring pixels [18]. A CAE may be a good choice to find spatial structures in images, but it cannot capture temporal information present in the contiguous frames of a video [33]. 3D-convolutions can be used to extract both temporal and spatial features by convolving a 3D kernel with the cube formed by stacking temporally contiguous frames of a video (we refer it as a window). This allows information across these contiguous frames to be connected to form feature maps, thereby capturing spatio-temporal information encoded in these adjacent frames [13]. This idea can be used to construct a 3D convolutional autoencoder (DSTCAE), where the encoding phase consists of dimensionality reduction both in space and time, of a window of images that are joined contiguously. In the decoding phase, the network attempts to reconstruct the whole window.

3.1 Sliding Window Reconstruction

The inputs to DSTCAE are windows of contiguous video frames. These windows are generated by applying a temporal sliding window to video frames, with window length T, padding (or not) and stride B (which represents the amount of frames shifted from one window to the next). If a video contains V frames, and padding is not used, then the number of windows (D) generated is

$$D = \lfloor \frac{V - T}{B} \rfloor + 1 \tag{1}$$

In our implementation of DSTCAE, we chose $T = 8$ (this choice is explained in Section 5.2), no padding and $B = 1$, which ensures that all the frames of the video are selected in forming windows. We re-size all the input frames to 64×64. The resulting network input (I) thus have dimensions $8 \times 64 \times 64$. DSTCAE encodes I into spatio-temporal features, and then attempts to reconstruct it from the encoded representation. For training DSTCAE, we use mean squared error loss between I and reconstructed output window O (same dimensions as I), optimized on a per batch basis, giving the following cost function:

$$C(\theta) = \frac{1}{N} \sum_{i=1}^{N} \|I_i - O_i\|_2^2 \tag{2}$$

where N is the number of training samples in a batch, θ denotes the network parameters, and $\| \cdot \|_2$ denotes the Euclidean norm.

3.2 3D Convolutions

The main component of a DSTCAE is the 3D layer, which is defined as follows: the value v at position (x, y, z) of the j^{th} feature map in the i^{th} 3D convolution layer, with bias b_{ij}, is given by the equation [13]

$$v_{ij}^{xyz} = f(\sum_{m} \sum_{p=0}^{P_i-1} \sum_{q=0}^{Q_i-1} \sum_{s=0}^{S_i-1} w_{ijm}^{pqs} v_{(i-1)m}^{(x+p)(y+q)(z+s)} + b_{ij}) \tag{3}$$

where P_i, Q_i, S_i are the vertical (spatial), horizontal (spatial), and temporal extent of the filter cube w_i in the i^{th} layer. The set of feature maps from the $(i-1)^{th}$ layer are indexed by m, and w_{ijm}^{pqs} is the value of the filter cube at position pqs connected to the m^{th} feature map in the previous layer. Multiple filter cubes will output multiple feature maps.

3.3 3D Encoding and Decoding

Encoding : The input I introduced in 3 in a DSTCAE is encoded by a sequence of 3D convolution and 3D-max-pooling layers. 3D convolutions operate as described in equation 3, with stride of $1 \times 1 \times 1$, and padding. The max-pooling layers use padding, with stride and kernel dimensions $2 \times 2 \times 2$. This means that each dimension (temporal depth, height, and width) is reduced by a factor of 2 with every max-pooling layer. This process is repeated for 2 level of depth. For hidden layers (in both encoding and decoding), the activation function f in equation 3 is set to $\mathbf{ReLU}$. We use $P_i = Q_i = 3$, and $S_i = 5$, for all convolutional and deconvolutional layers, as these values were found to produce the best results across all data sets. The specification of the encoding and decoding process is shown in Table 1.

DSTCAE-UpSampling	DSTCAE-Deconv	DSTCAE-C3D
Input - (8, 64, 64, 1)	Input - (8, 64, 64, 1)	Input - (8, 64, 64, 1)
3D Convolution - (8, 64, 64, 16)	3D Convolution - (8, 64, 64, 16)	3D Convolution - (8, 64, 64, 16)
3D Max-pooling - (4, 32, 32, 16)	3D Max-pooling - (4, 32, 32, 16)	3D Max-pooling - (8, 32, 32, 16)
3D Convolution - (4, 32, 32, 8)	3D Convolution - (4, 32, 32, 8)	3D Convolution - (8, 32, 32, 8)
3D Max-pooling - (2, 16, 16, 8)	3D Max-pooling - (2, 16, 16, 8)	3D Max-pooling - (4, 16, 16, 8)
3D Convolution - (2, 16, 16, 8)	3D Deconvolution - (4, 32, 32, 8)	3D Convolution - (4, 16, 16, 8)
3D UpSampling - (4, 32, 32, 8)	3D Deconvolution - (8, 64, 64, 16)	3D Max-pooling - (2, 8, 8, 8)
3D Convolution - (4, 32, 32, 8)	3D Deconvolution - (8, 64, 64, 1)	3D Convolution - (2, 8, 8, 8)
3D UpSampling - (8, 64, 64, 16)		3D UpSampling - (4, 16, 16, 8)
3D Convolution - (8, 64, 64, 1)		3D Convolution - (4, 16, 16, 8)
		3D UpSampling - (8, 32, 32, 8)
		3D Convolution - (8, 32, 32, 8)
		3D UpSampling - (8, 64, 64, 16)
		3D Convolution - (8, 64, 64, 1)

Table 1 Configuration of the encoding and decoding of DSTCAE-UpSampling, DSTCAE-Deconv, and DSTCAE-C3D. Encoding is in blue, and decoding in yellow

Decoding : For the decoding of DSTCAE, we explore two variants. The first method (DSTCAE-UpSampling) uses padded 3D convolutions with stride $2 \times 2 \times 2$, followed by a fixed UpSampling operation for increasing dimensions (as seen in Figure 1). In particular, we use 3D UpSampling layers as defined in Keras [6], with UpSampling factors $2 \times 2 \times 2$. That is, matrix elements are repeated across each dimension, such that the extent of all dimensions is doubled. The second method (DSTCAE-Deconv) uses 3D deconvolutions [6], with stride $2 \times 2 \times 2$, and padding [33] instead of UpSampling. This results in an increase in each dimension by a factor of 2, thus undoing a max-pooling operation. For both the decoding methods, the output layer activation function f is set to $\mathbf{tanh}$; computed element-wise in both cases. We use $\mathbf{tanh}$ for the output layer to limit the reconstructed pixel values in the range $[-1, 1]$, so that they are comparable to the input (see Section 5.2). In both decoding methods, outputs of the network (O) have the same dimensions as I, and are generated by convolving a single filter cube with a final layer of feature maps. By proposing two methods for the decoding procedure in DSTCAE, we created two variants of the spatio-temporal autoencoder, whose performance are compared for detecting unseen falls in Section 5.4.

A diagram illustrating the DSTCAE-UpSampling structure is shown in Figure 1. The first layer generates 16 feature maps (depicted in figure by vertical stacks), by convolving 16 different filter cubes with the input window of frames. This layer is followed by a 3D max-pooling layer. Convolution and pooling layers are repeated in this manner once more, resulting in an encoded dimension of $2 \times 16 \times 16 \times 8$. We choose to stop encoding at this point, in order to avoid collapsing the temporal dimension completely; a larger value of T would allow for a deeper network. Decoding then involves two UpSampling/3D convolution steps, bringing the hidden representation dimensions to $8 \times 64 \times 64 \times 16$. A final 3D convolution layer combines these 16 feature maps into the decoded reconstruction O.

We also test a third 3D convolutional autoencoder variant, based on Tran et al. [8], which we refer to as DSTCAE-C3D. The DSTCAE-C3D network has the same encoding and decoding as DSTCAE-UpSampling, but with an extra 3D Convolution/3D Max-pooling layer in encoding (see table 1), and extra 3D UpSampling/3D convolution in decoding. The extra 3D Max-pooling has padding, with stride and kernel dimensions $1 \times 2 \times 2$. This results in spatial dimension reduction, but not temporal; allowing for greater network depth without collapsing the temporal dimension.

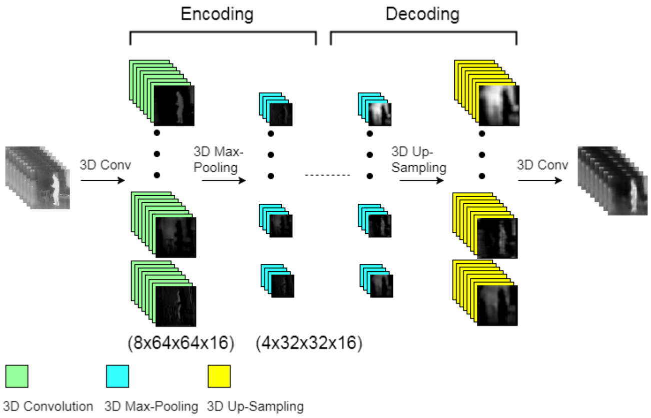


Fig. 1 Outline of DSTCAE layers using fixed UpSampling (DSTCAE-UpSampling). Each layer has dimensions (temporal depth × height × width × number of feature maps)

To summarize, we compare three variants of DSTCAE: 1) DSTCAE-UpSampling: uses UpSampling for decoding, 2) DSTCAE-Deconv: uses deconvolution for decoding (similar to [33]), and 3) DSTCAE-C3D: uses UpSampling for decoding, but contains additional spatial pooling/unpooling layer. These architectures are summarized in Table 1

We compare these DSTCAE variants to 2D frame based models CAE and DAE. Their specifications are shown in Table 2. This table only shows the configuration of the encoding phase of the autoencoder. The decoding configuration is the same, but using UpSampling, or deconvolution for CAE (referred to as CAE-UpSampling, and CAE-Decconv respectively), and more fully connected layers for DAE.

CAE	DAE
Input - (64, 64, 1)	Input - (64,64,1)
2D Convolution - (64, 64, 16)	Fully Connected - (4096)
2D Max-pooling - (32, 32, 16)	Fully Connected - (150)
2D Convolution - (32, 32, 8)	Fully Connected - (100)
2D Max-pooling - (16, 16, 8)	Fully Connected - (50)
2D Convolution - (16, 16, 8)	-
2D Max-pooling - (8, 8, 8)	-

Table 2 Configuration of the encoding phase of CAE and DAE.

4 Anomaly Scores to Detect Unseen Falls

In this problem, we formulated fall detection in a one-class classification framework, where normal ADL are available in abundance no fall data is present during training. However, during testing, falls may be present. Therefore, we detect falls as an anomaly. The general process for detecting unseen falls within the *DeepFall* framework is:

- Train a given type of autoencoder by minimizing reconstruction error on ADL,

– During testing, get reconstitution error of a frame (or a window comprising of different frames); if the reconstruction error is ‘high’, then a fall is flagged.

For CAE and DAE, the input to the network is a single frame; therefore, the reconstruction error is computed per frame. For all the variants of DSTCAE, the input to the network is a video sequence of size T. That is, given a test video sequence, we apply a sliding window as described in section 3. For the i^{th} window I_i, the network outputs a reconstruction of this window, O_i. The reconstruction error ($R_{i,j}$) between the j^{th} frame of I_i and O_i can be calculated as

$$R_{i,j} = \|I_{i,j} - O_{i,j}\|_2^2 \tag{4}$$

Figure 2 shows the sliding window approach for $T = 8$. The first window of $T = 8$ frames, I_1 (Fr_1 to Fr_8) are reconstructed, and their corresponding reconstruction error is stored ($R_{1,1:8}$). For the next window of frames, the input window is shifted forward in time by one frame. This process continues until all frames are used.

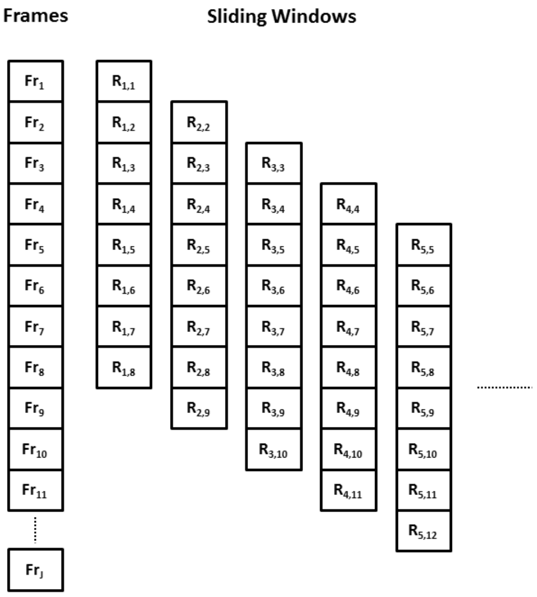


Fig. 2 Temporal sliding window showing reconstruction error ($R_{i,j}$) per frame (Fr_j) with $T = 8$.

4.1 Cross-Context Anomaly Score

To compare DSTCAE models with CAE and DAE, we must get a score per frame. A frame can appear in multiple windows. For instance, frame Fr_2 (see Figure 2) appears in two windows and, thus gives two reconstruction errors: $R_{1,2}$, and $R_{2,2}$. The former is attained when frame Fr_2 was the second frame in the window, and the latter case when frame Fr_2 was the first frame of the input window. Each window that a frame appears in provides a different temporal *context* within which this frame can be viewed. The cross-context anomaly score gives scores on a per-frame basis, by considering all of the reconstruction errors obtained for a frame across different windows (temporal contexts) [22]. For a frame j, an anomaly score can be calculated based on the mean (C_μ^j) or standard deviation (C_σ^j) of the reconstruction errors ($R_{i,j}$) across different windows (or contexts):

$$\begin{aligned} C_\mu^j &= \begin{cases} \frac{1}{j}\sum_{i=1}^{j} R_{i,j} & j < T \\ \frac{1}{T}\sum_{i=1}^{T} R_{i,j} & j \geq T \end{cases} \\ C_\sigma^j &= \begin{cases} \sqrt{\frac{1}{j}\sum_{i=1}^{j}(R_{i,j} - C_\mu^j)} & j < T \\ \sqrt{\frac{1}{T}\sum_{i=1}^{T}(R_{i,j} - C_\mu^j)} & j \geq T \end{cases} \end{aligned} \tag{5}$$

C_μ^j and C_σ^j give an anomaly score per-frame, while incorporating information from the past and future. A large value of C_μ^j or C_σ^j means that the j^{th} frame, when appearing at different positions in subsequent windows, is reconstructed with a high average error or highly variable error. In a normal ADL case, the reconstruction error of a frame should not vary a lot with its position in subsequent windows; however, if it does, then this may indicate anomalous behaviour, such as a fall.

4.2 Within-Context Anomaly Score

In a case when videos are considered as sequences of frames, we may want to designate the class of the whole window instead of deciding the class of every frame across different windows (or contexts). This way, all the frames present in a window will be used for predicting its class and they will have no influence on subsequent windows. Therefore, we call this method as within-context anomaly score. This method considers the reconstruction of frames within a context (or windows shown vertically in Figure 2), and gives a single score. In particular, the mean (W_μ^i) and standard deviation (W_σ^i) of the reconstruction error of all the frames within a window I_i, can be computed as an anomaly score to identify a fall,

$$W_\mu^i = \frac{1}{T}\sum_{j=i}^{T+i-1} R_{i,j}, \quad W_\sigma^i = \sqrt{\frac{1}{T}\sum_{j=i}^{T+i-1}(R_{i,j} - W_\mu^i)} \tag{6}$$

A large value of W_μ^i means that the average reconstruction error of the T frames within the window is high. Similarly, a large value of W_σ^i means that the variation of reconstruction error among the T frames of a window is high. Both the situations indicate that high reconstruction error may be happening due to one or more frames present within a window. Therefore, these situations represent the case when DSTCAE reconstructs the T frames in the window with high/variable reconstruction error, which may represent an anomalous behaviour such as a fall.

If the window of frames has one fall frame and it is termed as a fall, then *DeepFall* may be very sensitive and may generate many false positives. Whereas, if all the frames in a window comprises of falls and it is termed as a fall, then it may miss some falls. Therefore, choosing the number of the fall frames that makes a window of frames to be designated as a fall is a hyper-parameter of the *DeepFall* framework.

5 Experiments and Results

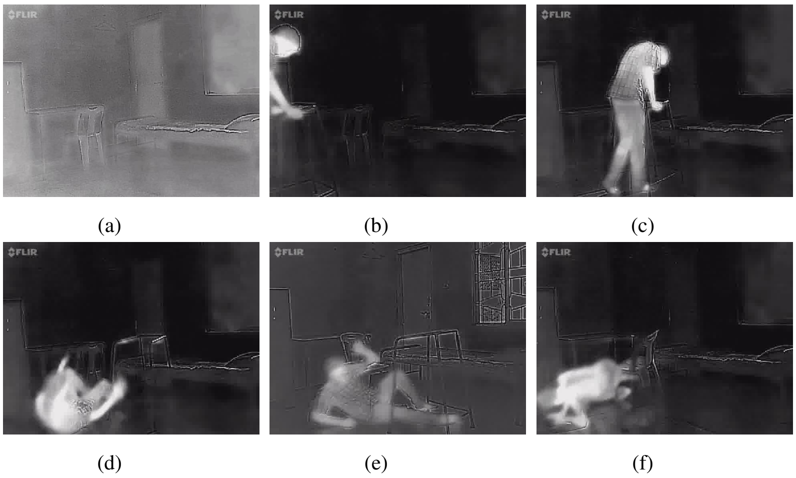
5.1 Datasets

We test the *DeepFall* framework on the following three activity data sets that collected normal ADL and falls using non-invasive sensing modalities.

5.1.1 Thermal Fall Dataset

This dataset consists of videos captured by a FLIR ONE thermal camera mounted on an Android phone in a room setting with a single view [29]. The videos have a frame rate of either 25 fps or 15 fps, which was obtained by observing the properties of each video. A total of 44 videos are collected, out of which 35 videos contain a fall along with normal ADL, and 9 videos contain only ADL. The spatial resolution of the thermal images is 640×480. The Thermal dataset contains many empty frames; that is, scenes where no person is present. It also contains frames of people entering the scene from the left and from the right. Examples of thermal ADL and fall frames are given in Figure 3.

In order to create windows of contiguous video frames to give as input to the DSTCAE for training and testing, we perform a sliding window on all video frames, as described in 3.1. We choose $T = 8$ because we do not want our window length to exceed the length of the shortest fall sequence, which is found to be 13 frames. We thus chose the smallest largest power of 2 smaller than 13, which is 8. We use the same window length for all datasets for simplicity. The Thermal dataset contains $22,116$ ADL frames from 9 videos. After windowing the ADL videos individually, we generate $22,053$ windows of contiguous frames (using equation 1), which are used for training spatio-temporal autoencoders.



(a) (b) (c)

(d) (e) (f)

Fig. 3 Thermal Dataset – ADL frmes (a) Empty Scene, (b) Person Entering the Scene, (c) Person in the Scene, and Fall Frames (d), (e), and (f).

5.1.2 UR Fall Detection Dataset

The UR dataset (UR) [3] contains 70 depth videos collected using a Microsoft Kinect camera at 30 fps. that was mounted parallel to the floor. Of these, 30 videos contain a fall, and 40 videos contain various ADL, such as walking, sitting down, crouching down, and lying down in bed. Five persons performed two types of falls – from standing position and from sitting on the chair. The pixels in the depth frames indicate the calibrated depth in the scene. The depth map is provided in a 640×480 resolution. The UR dataset contains empty frames. It also contains frames of people entering the scene towards the camera. Samples of original ADL and fall depth frame of the UR dataset are shown in Figure 4. After applying the sliding window, we obtain $8,661$ windows of contiguous frames used for training different spatio-temporal autoencoders.

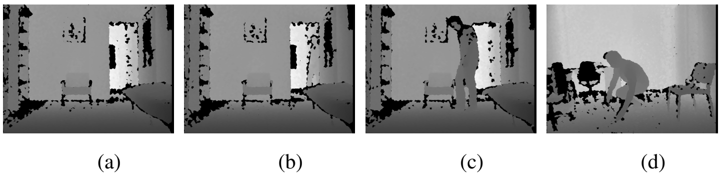


Fig. 4 UR Dataset - Original Depth frames with holes (a) Empty Scene (b) Person entering the Scene, (c) Person in the Scene, (d) Fall.

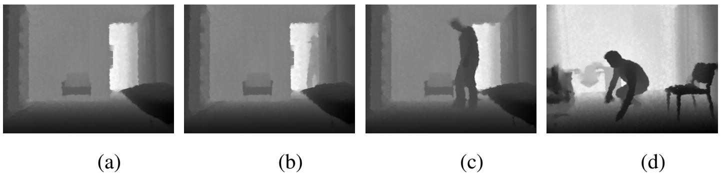


Fig. 5 UR Dataset - Depth frames after holes filling (a) Empty Scene (b) Person entering the Scene, (c) Person in the Scene, (d) Fall.

5.1.3 SDU Dataset

The SDU dataset [17], contains 1800 videos depth videos collected by a Microsoft Kinect camera (30 trials each, 10 subjects, 6 actions). The data that was shared with us contains $1,197$ depth videos. Of these videos, 997 contain the following ADL: bending, squatting, sitting, lying, and walking. The remaining 200 videos contain a fall, as well as other various ADL. The activities are performed by 20 young men and women. Each person performs 10 trials, which consist of simulating a fall, as well as performing each of the above ADL. The videos are recorded at 30 fps, with a spatial resolution of 320×240, and an average length of 5 seconds. After applying the sliding window, we obtain $163,573$ windows of contiguous frames used for training spatio-temporal autoencoders. The DAE and CAE variants are trained on the video frames without windowing. The SDU dataset contains empty frames. It also contains frames of people entering the scene from the left and right. Samples of original ADL and fall depth frame from the SDU dataset are shown in Figure 6.

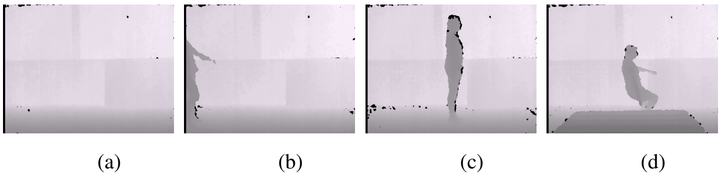


Fig. 6 SDU Dataset - Original Depth frames with holes (a) Empty Scene (b) Person entering the Scene, (c) Person in the Scene, (d) Fall.

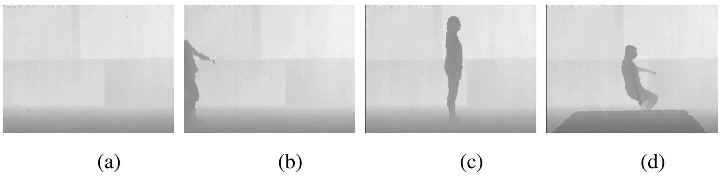


Fig. 7 SDU Dataset - Depth frames after hole filling, (a) Empty Scene (b) Person entering the Scene, (c) Person in the Scene, (d) Fall.

5.2 Data-preprocessing

The Thermal data set frames were extracted from mp4 video files. The UR dataset's frames were already available in png format. The SDU dataset frames were extracted from AVI video files. All the frames in all the datasets are normalized by dividing the pixel values by 255 to keep them in the range $[0,1]$, and subtracting the per-frame mean from each frame, resulting in pixel values in the range $[-1,1]$. As discussed in section 3.1, all the frames are also re-sized to 64×64. The video frames used for training the models were not annotated because they were all considered normal ADL. In order to test the models, fall videos are used that contained both fall and non-fall frames. The fall frames were manually annotated in these videos. Figure 8 shows the training and testing scheme for evaluating different autoencoders used in the paper.

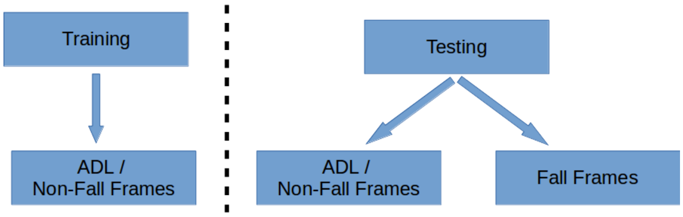


Fig. 8 Training and testing scheme for evaluating different autoencoders.

In both the UR dataset and the SDU dataset, there were many regions of missing pixels, termed as ‘holes’. Both of these datasets uses Microsoft Kinect depth sensor, which is a structured 3D scanner and light measurements from it can contain errors due to multiple reflections, transparent objects, occlusion or scattering [30]. These these holes can be seen as black regions in Figures 4 and 6. These holes can be detrimental to learning useful spatio-temporal features using the *DeepFall* framework. Therefore, we employed hole filling methods to fill in these regions. For the UR dataset, we used a method based on depth colorization [21]. Figure 5 shows the corresponding depth frame after filling holes in the frames shown in Figure 4. For the SDU dataset, we did not have information of distance in the depth frames, since they were extracted from AVI formatted videos. That is, the depth images contain values in the range $[0, 255]$, where darker pixels correspond to a lesser distance from the camera; true distance is not known for these values. Distance information is required for using the depth colorization algorithm [21]. Thus, we used a simple inpainting technique, based on an OpenCV [12] implementation of the inpainting algorithm presented in [2]. Figure 7 shows the results of applying this inpainting technique on the frame shown in Figure 6.

To evaluate the effectiveness of hole filling in detecting falls using different autoencoder methods, we created another version of the UR and SDU dataset in which the holes are filled for all the frames, we call them as UR-Filled and SDU-Filled.

5.3 Experimental Setup

Different types of autoencoders are trained on only the ADL videos, which only contain ADL / non-fall frames. The testing was carried out on fall videos, which contain fall activity as well as some ADL (see Figure 8) For comparing the performance of different variants of spatial-temporal autoencoder, we implemented Deep Autoencoder (DAE) and Deep Convolutional Autoencoder (CAE). All the different autoencoders are trained for 500 epochs. Adadelta optimizer was used in training these models. The training batch size is set to 32 for DAE and CAE models, and 16 for DSTCAE variants, where each batch consists of a stack of 8 frames. To train DAE and CAE variants, we augment the data by performing horizontal flipping. No data augmentation was performed when training DSTCAE variants, as it did not improve results. Also, dropout is applied to layer 1 for DAE. We also use dropout for layer 2 for all DSTCAE variants. In all cases the dropout probability is set to 0.25. Dropout was not found to be beneficial for CAE models, and so was not used [1].

The within-context anomaly score can only be calculated for the DSTCAE variants, because it calculates a score for each window of T frames rather than individual frames. In this method, if there are α amount of fall frames, then the ground truth of the entire sequence of frames is labeled as a fall. The value of α is varied from $\alpha = 1$ to $\alpha = T = 8$. These anomaly scores are then used to compute the area under the curve (AUC) of the ROC with fall as the class of interest. The cross-context anomaly score gives a score per frame (for a given video); therefore, it can be directly compared with DAE and CAE variants. The anomaly scores obtained for every frame is used to calculate AUC of the ROC, with fall as the class of interest. For DAE and CAE variants, the reconstruction error of frames of a test video are considered as anomaly score. The reported AUC is the average of AUC across all test videos. As discussed in Section 3.3, we test two variants of DSTCAE. Thus the DSTCAE variants tested are DSTCAE-UpSampling, DSTCAE-Deconv, DSTCAE-C3D.

5.4 Results

5.4.1 Cross-Context Anomaly Score

The results for all frame-based models i.e. CAE variants, DAE, and DSTCAE variants using the cross context score, are presented in Table 3. We also present the AUC values computed

[1] The code for DSTCAE is available at https://github.com/JJN123/Fall-Detection

using an ensemble of one-class JK nearest neighbour method [14] trained over the features learned from DAE; features from the fourth fully connected layer were used, which have dimension 1×50. The nearest neighbours are set to 1 (JKNN-11) and 10 (JKNN-JK) and the ensemble sizes is fixed to 25. The results for UR and SDU using the one-class JK nearest neighbour method could not be obtained in reasonable time because it calculates euclidean distance for all pairs of input; thus making the method very slow. We also compare the results from our previous work of using CLSTMAE [22] over all the datasets with cross-context scores C_σ and C_μ. The first column of 3 indicates the model tested, and the other columns show the dataset. Each numerical value in the table represents the average ROC AUC across all videos for the model indicated by the row, and the data set indicated by the column. The respective standard deviation of ROC AUC across all videos of the dataset is shown in small brackets. The results shown in Figure 3 can be summarized as follows

- All the variants of spatio-temporal autoencoders outperforms spatial feature extraction methods (i.e. JKNN, DAE and two CAE variants). This confirms the fact that extracting both spatial and temporal features are important in video based fall detection approach.
- All the variants of DSTCAE outperforms the CLSTM based methods. This marks an improvement on our previous work [22]. This may be the case since LSTM based models are tailored to long term sequential modelling, but falls occur on short time scales. It further confirms that DSTCAE is better suited for such problems of short length sequences.
- The autoencoders trained on holes filled versions of UR and SDU datasets performed better for all the classifiers (except for one CLSTMAE-C_σ for the SDU dataset, which is worse than the best AUC obtained for this dataset with holes-filled). This result shows that if the depth data contains holes or missing pixels, it must be filled with some technique for the autoencoder to detect falls appropriately.
- The difference between DSTACE-Deconv and DSTCAE-UpSampling appears to be insignificant, but DSTCAE-C3D performs the best on two out of three datasets (excluding depth datasets without holes filled, which are degenerative cases).
- We see that C_σ outperforms on C_μ on Thermal and UR-Filled data.

Models	Thermal	UR	UR-Filled	SDU	SDU-Filled
JKNN-11	0.53	-	0.5	-	0.72
JKNN-JK	0.53	-	0.5	-	0.86
DAE	0.65(0.14)	0.38(0.14)	0.75(0.15)	0.72(0.16)	0.94(.05)
CAE-UpSampling	0.73(0.12)	0.38(0.14)	0.67(0.19)	0.64(0.16)	0.89(.06)
CAE-Deconv	0.75(0.17)	0.38(0.11)	0.76(0.21)	0.66(0.16)	0.92(0.07)
CLSTMAE: C_σ	0.64(0.16)	0.49(0.09)	0.67(0.20)	0.66(0.10)	0.56(0.11)
CLSTMAE: C_μ	0.58(0.20)	0.43(0.11)	0.82(0.16)	**0.77(0.11)**	0.92(0.06)
DSTCAE-UpSampling:C_σ	0.96(0.03)	0.69(0.18)	**0.89(0.09)**	0.74(0.12)	0.90(0.06)
DSTCAE-UpSampling:C_μ	0.95(0.04)	0.50(0.14)	0.88(0.12)	0.70(0.18)	0.92(0.05)
DSTCAE-Deconv:C_σ	0.96(0.02)	**0.72(0.17)**	0.88(0.10)	0.70(0.13)	0.92(0.05)
DSTCAE-Deconv:C_μ	0.94(0.04)	0.48(0.14)	0.85(0.11)	0.69(0.18)	0.92(0.05)
DSTCAE-DSTCAE-C3D:C_σ	**0.97(0.02)**	0.54(0.13)	0.80(0.13)	0.76(0.11)	0.92(0.06)
DSTCAE-DSTCAE-C3D:C_μ	0.93(0.07)	0.45(0.12)	0.86(0.17	0.71(0.18)	**0.95(0.04)**

Table 3 ROC AUC values for different methods, for each dataset, based on cross-context anomaly score.

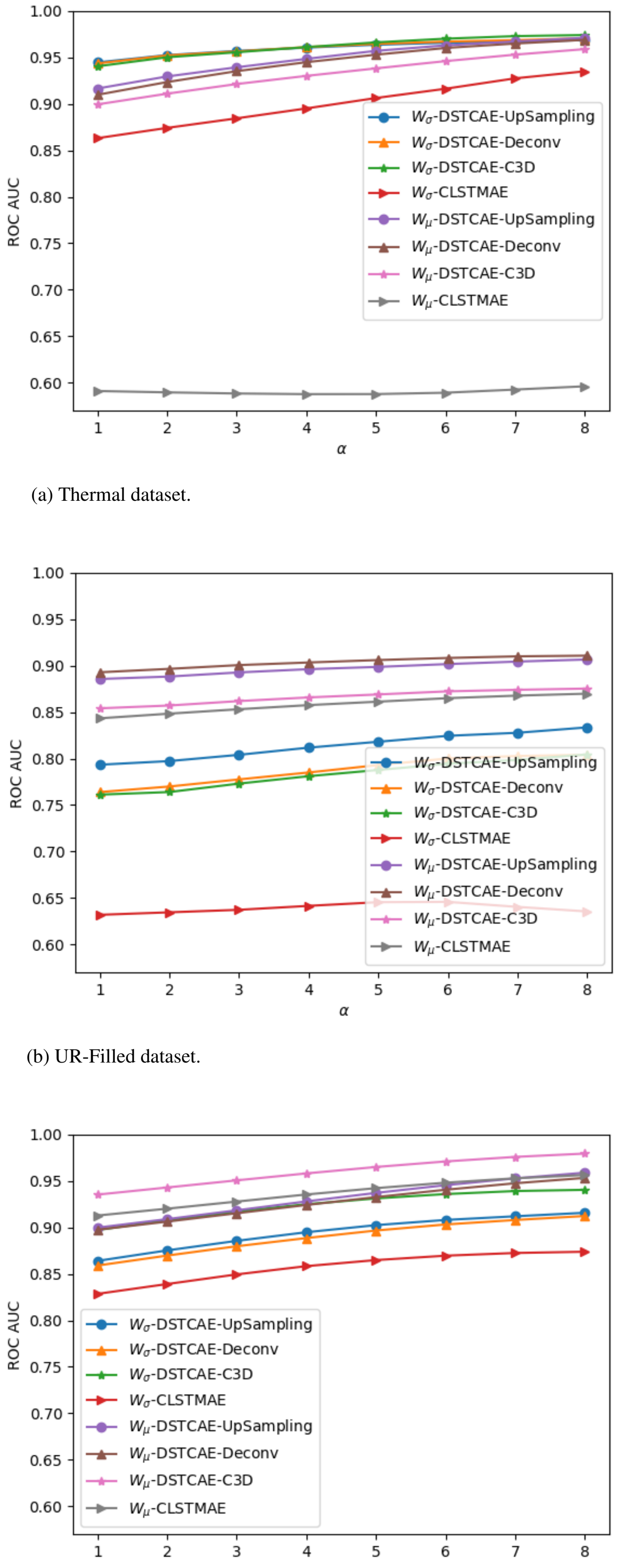
5.4.2 Within-Context Anomaly Score

The results achieved using the wintin-context anomaly scores, W_μ and W_σ, for all the datasets are shown in Figure 9. We do not show scores for SDU and UR with holes, because results were significantly worse than those with holes filled. We observe that increasing α increases the ROC AUC for Thermal and SDU-Filled datasets, and to a lesser extent, UR-Filled dataset. This indicates that reducing false positives is effective in increasing the ROC AUC score, and thus the ability to detect falls.

We also observe that the ROC AUC attained using W_μ and W_σ (with sufficiently high α) is at least as high, and in some cases higher than using frame based scores C_σ and C_μ. That is, for Thermal dataset, the highest ROC AUC remained consistent from within-context to cross-context scores. For UR-Filled, the ROC AUC improved from 0.89 (C_σ) to 0.91 (W_μ), and for SDU-Filled dataset, ROC AUC improved from 0.95 (C_σ) to 0.98 (W_μ).

6 Conclusions and Future Work

Detecting falls in a non-invasive manner is a challenging problem; especially, as falls occur rarely. In this paper, we formulated detecting falls as an anomaly detection problem and presented the *DeepFall* framework, which shows the development and comparison of three variants of DSTCAE. We tested the *DeepFall* framework on three datasets that captured ADL and falls in a non-invasive manner using thermal and depth camera. We presented a new anomaly scoring method - referred to as the within-context anomaly score. The results showed that the *DeepFall* framework outperforms the standard CAE, as well as traditional DAE. We also found that DSTCAE variants outperformed CLSTMAE. It was also observed that the within-context anomaly score, outperformed the per-frame based anomaly scoring method (or cross-context anomaly score) on two out of three datasets (excluding depth data sets without hole filling). In the future, we will explore the application of Generative Adversarial Networks [9] to the task of fall detection in an unsupervised setting.



(a) Thermal dataset.

(b) UR-Filled dataset.

(c) SDU-Filled dataset.

Fig. 9 Effect of changing fall frames in a stack of frames (α) on the ROC AUC.

References

1. Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., Baskurt, A.: Spatio-temporal convolutional sparse autoencoder for sequence classification. In: In BMVC. Citeseer (2012)
2. Bertalmio, M., Bertozzi, A.L., Sapiro, G.: Navier-stokes, fluid dynamics, and image and video inpainting. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol. 1, pp. I–355–I–362 vol.1 (2001). DOI 10.1109/CVPR.2001.990497
3. Bogdan Kwolek, M.K.: Human fall detection on embedded platform using depth maps and wireless accelerometer. Computer Methods and Programs in Biomedicine **117**, 489–501 (2014)
4. CDC: enter of Disease Control and Prevention – Important Facts about Falls. `https://www.cdc.gov/homeandrecreationalsafety/falls/adultfalls.html` (2017). [Online accessed 22-December-2017]
5. Chalapathy, R., Menon, A.K., Chawla, S.: Robust, deep and inductive anomaly detection. In: European Conference on Machine Learning. Skopje (2017)
6. Chollet, F., et al.: Keras: The python deep learning library. `https://github.com/fchollet/keras` (2015). Online accessed 20-January-2018
7. Chong, Y.S., Tay, Y.H.: Abnormal event detection in videos using spatiotemporal autoencoder. In: International Symposium on Neural Networks, pp. 189–196. Springer (2017)
8. Du Tran Lubomir Bourdev, R.F.L.T.M.P.: Learning spatiotemporal features with 3d convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). IEEE (2015). DOI 10.1109/ICCV.2015.510
9. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems, pp. 2672–2680 (2014)
10. Gutoski, M., Aquino, N.M.R., Ribeiro, M., Lazzaretti, A.E., Lopes, H.S.: Detection of video anomalies using convolutional autoencoders and one-class support vector machines. In: Proc. XIII Brazilian Congress on Computational Intelligence (2017)
11. Hasan, M., Choi, J., Neumann, J., Roy-Chowdhury, A.K., Davis, L.S.: Learning temporal regularity in video sequences. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 733–742 (2016)
12. Itseez: Open source computer vision library. `https://github.com/itseez/opencv` (2015)
13. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE transactions on pattern analysis and machine intelligence **35**(1), 221–231 (2013)
14. Khan, S.S., Ahmad, A.: Relationship between variants of one-class nearest neighbors and creating their accurate ensembles. IEEE Transactions on Knowledge and Data Engineering **30**(9), 1796–1809 (2018). DOI 10.1109/TKDE.2018.2806975
15. Khan, S.S., Karg, M.E., Kulić, D., Hoey, J.: Detecting falls with x-factor hidden markov models. Applied Soft Computing **55**, 168–177 (2017)
16. Khan, S.S., Taati, B.: Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders. Expert Systems with Applications (2017)
17. Ma, X., Wang, H., Xue, B., Zhou, M., Ji, B., Li, Y.: Depth-based human fall detection via shape features and improved extreme learning machine. IEEE Journal of Biomedical and Health Informatics **18**(6), 1915–1922 (2014). DOI 10.1109/JBHI.2014.2304357
18. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. In: International Conference on Artificial Neural Networks, pp. 52–59. Springer (2011)
19. Mercuri, M., Garripoli, C., Karsmakers, P., Soh, P.J., Vandenbosch, G.A., Pace, C., Leroux, P., Schreurs, D.: Healthcare system for non-invasive fall detection in indoor environment. In: Applications in Electronics Pervading Industry, Environment and Society, pp. 145–152. Springer (2016)
20. Munawar, A., Vinayavekhin, P., De Magistris, G.: Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space. In: Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on, pp. 1017–1025. IEEE (2017)
21. Nathan Silberman Derek Hoiem, P.K., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: ECCV (2012)
22. Nogas, J., Khan, S.S., Mihailidis, A.: Fall detection from thermal camera using convolutional lstm autoencoder. In: Proceedings of the 2^{nd} workshop on Aging, Rehabilitation and Independent Assisted Living, IJCAI Workshop (2018)
23. Penttilä, J.: A method for anomaly detection in hyperspectral images, using deep convolutional autoencoders. Master's thesis, University of Jyväskylä, Finland (2017)
24. Ribeiro, M., Lazzaretti, A.E., Lopes, H.S.: A study of deep convolutional auto-encoders for anomaly detection in videos. Pattern Recognition Letters (2017)

25. Rubenstein, L.Z., Robbins, A.S., Josephson, K.R., Schulman, B.L., Osterweil, D.: The value of assessing falls in an elderly population: a randomized clinical trial. Annals of internal medicine **113**(4), 308–316 (1990)
26. Sabokrou, M., Fayyaz, M., Fathy, M., Klette, R.: Deep-cascade: cascading 3d deep neural networks for fast anomaly detection and localization in crowded scenes. IEEE Transactions on Image Processing **26**(4), 1992–2004 (2017)
27. Skubic, M., Harris, B.H., Stone, E., Ho, K., Su, B.Y., Rantz, M.: Testing non-wearable fall detection methods in the homes of older adults. In: Engineering in Medicine and Biology Society (EMBC), 2016 IEEE 38th Annual International Conference of the, pp. 557–560. IEEE (2016)
28. Tran, H.T., Hogg, D.: Anomaly detection using a convolutional winner-take-all autoencoder. In: Proceedings of the British Machine Vision Conference 2017. Leeds (2017)
29. Vadivelu, S., Ganesan, S., Murthy, O.R., Dhall, A.: Thermal imaging based elderly fall detection. In: ACCV Workshop, pp. 541–553. Springer (2016)
30. Viacheslav, V., Alexander, F., Vladimir, M., Svetlana, T., Oksana, L.: Kinect depth map restoration using modified exemplar-based inpainting. In: Signal Processing (ICSP), 2014 12th International Conference on, pp. 1175–1179. IEEE (2014)
31. Xu, D., Yan, Y., Ricci, E., Sebe, N.: Detecting anomalous events in videos by learning deep representations of appearance and motion. Computer Vision and Image Understanding **156**, 117–127 (2017)
32. Yusif, S., Soar, J., Hafeez-Baig, A.: Older people, assistive technologies, and the barriers to adoption: A systematic review. International journal of medical informatics **94**, 112–116 (2016)
33. Zhao, Y., Deng, B., Shen, C., Liu, Y., Lu, H., Hua, X.S.: Spatio-temporal autoencoder for video anomaly detection. In: Proceedings of the 2017 ACM on Multimedia Conference, MM '17. ACM, New York, NY, USA (2017)