# Detecting Unseen Falls from Wearable Devices using Channel-wise Ensemble of Autoencoders

Shehroz S. Khan[a,b,*], Babak Taati[a]

[a] *Toronto Rehabilitation Institute, University Health Network, Toronto, Canada*
[b] *University of Toronto, Canada*

**Abstract**

A fall is an abnormal activity that occurs rarely, so it is hard to collect real data for falls. It is, therefore, difficult to use supervised learning methods to automatically detect falls. Another challenge in automatically detecting falls is the choice of engineered features. In this paper, we formulate fall detection as an anomaly detection problem and propose to use an ensemble of autoencoders to learn features from different channels of wearable sensor data trained only on normal activities. We show that the traditional approach of choosing a threshold as the maximum of the reconstruction error on the training normal data is not the right way to identify unseen falls. We propose two methods for automatic tightening of reconstruction error from only the normal activities for better identification of unseen falls. We present our results on two activity recognition datasets and show the efficacy of our proposed method against traditional autoencoder models and two standard one-class classification methods.

*Keywords:* fall detection, one-class classification, autoencoder, anomaly detection

## 1. Introduction

Falls are a major cause of both fatal and non-fatal injury and a hindrance in living independently. Each year an estimated 424,000 individuals die from falls globally and 37.3 million falls require medical attention (Organization, 2016). Experiencing a fall may lead to a fear of falling (Igual et al., 2013), which in turn can result in lack of mobility, less productivity and reduced quality of life. There exist several commercial wearable devices to detect falls (Pannurat et al., 2014); most of them use accelerometers to capture motion information. They normally come with an alarm button to manually contact a caregiver if the fall is not detected by the device. However, most of the devices for detecting falls produce many false alarms (El-Bendary et al., 2013). Automatic detection of falls is long

*Corresponding author: Tel.: +1 416-597-3422; Fax: +1 416-597-6201;
*Email addresses:* `shehroz.khan@utoronto.ca` (Shehroz S. Khan), `babak.taati@uhn.ca` (Babak Taati)

sought; hence, machine learning techniques are needed to automatically detect falls based on sensor data. However, a fall is a rare event that does not happen frequently (Stone and Skubic, 2015); therefore, during the training phase, there may be very few or no fall samples. Standard supervised classification techniques may not be suitable in this type of skewed data scenario (Khan et al., 2014). Another issue regarding the use of machine learning methods in fall detection is the choice of features. Traditional activity recognition and fall detection methods extract a variety of domain specific features from raw sensor readings to build classification models (Ravi et al., 2005; Khan, 2016). It is very difficult to ascertain the number or types of engineered features, specially in the absence of fall specific training data to build generalizable models.

To handle the problems of lack of training data from real falls and the difficulty in engineering appropriate features, we explore the use of Autoencoders (AE) (Japkowicz et al., 1995) that are trained only on normal activities. AEs can learn generic features from the raw sensor readings and can be used to identify unseen falls as abnormal activities during testing based on a threshold on the reconstruction error. We present two ensembles approaches of AE that train on the raw data of the normal activities from different channels of accelerometer and gyroscope separately and the results of each AE is combined to arrive upon a final decision. Typically, while using AE, the maximum of reconstruction error on the training set is considered as the threshold to identify an activity as abnormal (Dau et al., 2014). However, we experimentally show that such threshold may not be appropriate for detecting falls due to noisy sensor data. We present two threshold tightening techniques to remove few outliers from the normal data. Then, either a new threshold is derived using inter-quartile range or by training a new AE on the training data with outliers removed. We show result on two activity recognition datasets that contain different normal activities along with falls from wearable sensors.

The rest of the paper is organized as follows. In the next section, we present a brief introduction to AEs. Section 3 reviews the literature on detecting falls as anomaly, by using AE and on the use of AE in general outlier detection tasks. We present the proposed channel-wise ensemble of AE and two threshold tightening approaches using reconstruction error in Section 5. Experimental analysis and results are discussed in Section 6, followed by conclusions and future work in Section 7.

## 2. Brief Introduction to Autoencoders

An AE is an unsupervised multi-layer neural network that learns compact representation of the input data (Scholz and Vigário, 2002). An AE tries to learn an identity function such that its outputs are similar to its inputs. However, by putting constraint on the network, such as limiting the number of hidden neurons, it can discover compact representations of the data that can be used as features for other supervised or unsupervised learning tasks. An AE is often trained by using the backpropagation algorithm and consists of an encoder and decoder part. If there is one hidden layer, an AE takes the input $\mathbf{x} \in \mathbb{R}^d$ and maps it onto $\mathbf{h} \in \mathbb{R}^p$, s.t.

$$\mathbf{h} = f(W\mathbf{x} + b) \tag{1}$$

where $\mathbf{W}$ is a weight matrix and $\mathbf{b}$ is a bias term and $f(.)$ is a mapping function. This step is referred to as encoding or learning latent representation, after which $\mathbf{h}$ is mapped back to reconstruct $\mathbf{y}$ of the same shape as $\mathbf{x}$, i.e.

$$\mathbf{y} = g(\mathbf{W'h} + \mathbf{b'}) \tag{2}$$

This step is referred to as decoding or reconstructing the input back from latent representation. An AE can be used to minimize the squared reconstruction error, $L$ i.e.,

$$L(\mathbf{x},\mathbf{y}) = \| \mathbf{x} \text{ -} \mathbf{y} \|^2 \tag{3}$$

AE can learn compact and useful features if $p < d$; however, it can still discover interesting structures if $p > d$. This can be achieved by imposing a sparsity constraint on the hidden units, s.t. neurons are inactive most of the time or the average activation of each hidden neuron is close to zero. To achieve sparsity, an additional sparsity parameter is added to the objective function. Multiple layers of AEs can be stacked on top of each other to learn hierarchical features from the raw data. They are called Stacked AE (SAE). During encoding of a SAE, the output of first hidden layer serves as the input to the second layer, which will learn second level hierarchical features and so on. For decoding a SAE, the output of the last hidden layer is reconstructed at the second last hidden layer, and so on until the original input is reconstructed.

## 3. Related Work

AEs can be used both in supervised and unsupervised settings for identifying falls. In a supervised classification setting, AE is used to learn representative features from both the normal and fall activities. This step can be followed by a standard machine learning classifier trained on these compressed features (Li et al., 2014b) or by a deep network (Jokanovic et al., 2016). In the unsupervised mode or One-Class Classification (OCC) (Khan and Madden, 2014) setting, only data for normal activities is present during training the AE. In these situations, an AE is used to learn representative features from the raw sensor data of normal activities. This step is followed by either employing (i) a discriminative model by using one-class classifiers or (ii) a generative model with appropriate threshold based on reconstruction error, to detect falls and normal activities. The present paper follows the unsupervised AE approach with a generative model and aimed at finding an appropriate threshold to identify unseen falls.

Machine learning techniques are extensively used for handling fall detection problem (Özdemir and Barshan, 2014); it has also been formulated as an anomaly detection problem due to the fact that falls occur rarely (Khan and

Hoey, 2017). Popescu and Mahnot (Popescu and Mahnot, 2009) present a fall detection technique that uses acoustic signals of normal activities for training and detecting fall sounds from it. They train One-class SVM (OSVM), one-class nearest neighbour approach (OCNN) and One-class Gaussian Mixture Model to train models on normal acoustic signals and find that OSVM performs the best in detecting falls. However, it is outperformed by its supervised counterpart. Khan et al. (Khan et al., 2015) propose an unsupervised acoustic fall detection system with interference suppression that makes use of the features extracted from the normal sound samples, and constructs an OSVM model to distinguish falls from non-falls. They show that in comparison to Popescu and Mahnot (Popescu and Mahnot, 2009), their interference suppression technique makes the fall detection system less sensitive to interferences by using only two microphones. Principi et al. (Principi et al., 2016) present a floor acoustic sensor that can automatically discriminates the sounds produced by falls of distinct objects. They show that this sensor can capture fall signals with high signal-to-noise ratio with respect to an aerial microphone by filtering out high frequency components. Tran et al. (Tran et al., 2014) propose to use image and audio to tackle the problem of abnormal events detection, such as, falling, lying motionless, etc. They introduce audio and video based event detection systems that resulted in high sensitivity and low false alarm rate in two setup environments. Medrano et al. (Medrano et al., 2014) propose to identify falls using a smartphone as a novelty from the normal activities and find that OCNN performs better than OSVM but is outperformed by supervised SVM. Micucci et al. (Micucci et al., 2015) evaluate several OCC methods for fall detection using data from smartphone and show that in most of the cases, OCNN performs better or similar to the supervised SVM and KNN. Zhou et al.(Zhou et al., 2012) present a method to detect falls using transitions between the activities as a cue to model falls. They train supervised classification methods using normal activities collected from a mobile device, then extract transitions among these activities, and use them to train an OSVM. They show that this method performs better than an OSVM trained with only normal activities. Khan et al (Khan et al., 2014) present 'X-Factor' HMM approaches that are similar to normal HMMs, but have inflated output covariances that can be used as alternative models to estimate the parameters of unseen falls. Their results show high detection rates for falls on two activity recognition datasets.

A lot of work has been done in evaluating the feasibility of learning generic representations through AEs for general activity recognition and fall detection tasks. Plötz et al. (Plötz et al., 2011) explore the potential of discovering universal features for context-aware application using wearable sensors. They present several feature learning approaches using PCA and AE and show their superior performance in comparison to standard features across a range of activity recognition applications. Budiman et al. (Budiman et al., 2014) use SAEs and marginalized SAE to infer generic features in conjunction with neural networks and Extreme Learning Machines as the supervised classifiers to perform pose-based action recognition. Li et al. (Li et al., 2014a) compare SAE, Denoising AE and PCA for unsupervised feature learning in activity recognition using smartphone sensors. They show that traditional features perform worse than the generic features inferred through AEs. Jokanovic et al. (Jokanovic et al., 2016) use a SAE to learn generic lower dimensional features and use softmax regression classifier to identify falls using radar signals. Other researchers (Jankowski et al., 2015; Wang, 2016) have used AEs to reduce the dimensionality of domain specific features prior to applying traditional supervised classification models or deep belief networks.

AEs have also been extensively used in general anomaly detection tasks. Japkowicz et al. (Japkowicz et al., 1995) present the use of AE for novelty detection. For noiseless data, they propose to use a reduced percentage of maximum of reconstruction error as a threshold to identify outliers. For noisy data, they propose to identify both the intermediate positive and negative regions and subsequently optimizing the threshold until a desired accuracy is achieved. Manevitz and Yousef (Manevitz and Yousef, 2007) present an AE approach to filter documents and report better performance than traditional classifiers. They report to carry out certain type of uniform transformation before training the network to improve the performance. They discuss that choosing an appropriate threshold to identify normal documents is challenging and present several variants.The method that works best in their application is to tighten the threshold sufficiently to disallow the classification of the highest 25 percentile error cases from the training set. Dau et al. (Dau et al., 2014) use an AE to learn compact representation of different benchmark datasets and use maximum of the reconstruction error as the threshold; however, they mention that some instances of anomalies consistently have low reconstruction error. Erfani et al. (Erfani et al., 2016) present a hybrid approach to combine the AEs and OSVM for anomaly detection in high-dimensional and large-scale applications. They first extract generic features using SAE and train an OSVM with a linear kernel on learned features from SAE. They also use SAE as a one-class classifier by setting the threshold to be 3 times of standard deviation away from the mean. Sakurda and Yairi (Sakurada and Yairi, 2014) show the use of AE in anomaly detection task and compare it with PCA and Kernel PCA. They demonstrate that the AE can detect subtle anomalies that PCA could not and is less complex than Kernel PCA. Marchi et al. (Marchi et al., 2015a,b) present an unsupervised approach to acoustic novelty detection using auditory features and DAE with bidirectional Long Short-Term Memory recurrent neural networks. They use reconstruction error as a basis to identify unseen novel sounds. They set the threshold proportional to the median of the reconstruction error of the error signal of a sequence.

Ensembles of AE have been used to learn diverse feature representations, mainly in the supervised settings. Ithapu et al. (Ithapu et al., 2014) present an ensemble of SAE by presenting it with randomized inputs and randomized sample sets of hyper-parameters from a given hyperparameter space. They show that their approach is more accurately related to different stages of Alzheimer's disease and leads to efficient clinical trials with very less sample estimates. Reeve and Gavin (Reeve and Brown, 2015) present a modular AE approach that consists of $M$ AE modules trained separately on different data representations and the combined result is defined by taking an average of all the modules present. Their results on several benchmark datasets show improved performance in comparison to baseline of bootstrap version of the AE. Dong and Japkowicz (Dong and Japkowicz, 2016) present a supervised and unsupervised ensemble approach for stream learning that uses multi-layer neural networks and AE. They train their models from multi-threads which evolve with data streams. The ensemble of the AE is trained using only the data from positive class and is accurate when anomalous training data are rare. Their method performs better as compared to the state-of-the-art in terms of detection accuracy and training time for the datasets.

The literature review shows that AEs can successfully learn generic features from raw sensor data for activity and fall recognition tasks. We observe that AE can be effectively used for anomaly detection tasks and their ensembles can perform better than a single AE. In this paper, fall detection problem is formulated as an OCC or anomaly detection, where abundant data for normal activities is available during training and none for falls. We investigate the utility of features learned through AEs and their ensembles for the task of fall detection in the next section.

## 4. Autoencoder Ensemble for Detecting Unseen Falls

In the absence of training data for falls, a fall can be detected by training an AE/SAE on only the normal activities to learn generic features from a wearable device. These features can be fed to standard OCC algorithms to detect a test sample as a normal activity or not (a fall in our case). Alternatively, a threshold can be set on the reconstruction error of the AE/SAE to identify a test sample as an abnormal activity (a fall in our case), if its reconstruction error is higher than a given threshold. Intuitively, this would mean that the test sample is very different from the training data comprising of normal activities. Below, we discuss two types of AE approaches used in the paper.

### *4.1. Monolithic Autoencoders*

Figure 1 shows the AE/SAE for training normal activities using raw sensor data from a three-channel accelerometer and gyroscope. The raw sensor readings coming from each of the channels of accelerometer $(a_x, a_y, a_z)$ and gyroscope $(\omega_x, \omega_y, \omega_z)$ are combined and presented as input to the AE/SAE. For a sliding window of a fixed length ($n$ samples), $\mathbf{a_x} = [a_x^1, a_x^2, ..., a_x^n]$, $\mathbf{a_y} = [a_y^1, a_y^2, ..., a_y^n]$, and so on. The input vector for a time window is constructed by concatenating these sensor readings as $\mathbf{f} = [\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}, \boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}]^T$. We call this feature learning approach as *monolithic* because it combines raw sensor data from different channels as one input to an AE.

### *4.2. Channel-wise Autoencoders*

Li et al. (Li et al., 2014a) present the use of ensemble of SAE by extracting generic features per each of the three accelerometer channels and additional

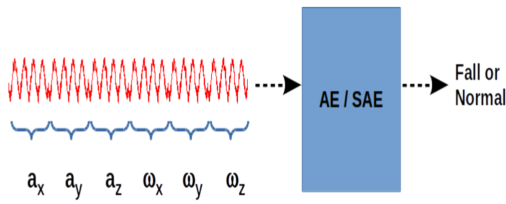


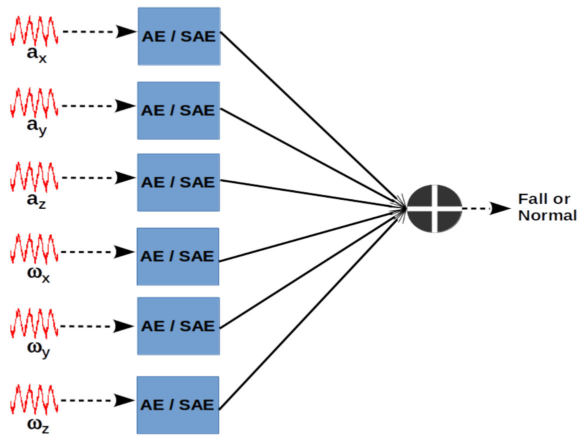

Figure 1: Monolithic AE for detecting unseen falls.

channel for the magnitude of the accelerometer vector in 3-dimensions. They extract fixed number of features for each of these 4 channels and concatenate them. Supervised classification methods are then used on these extracted features. This setting can work for supervised classification but not in OCC scenario. In our case, we deal with OCC scenario with only normal data available during training. Therefore, separate AEs are trained on the raw data from different channels of the sensors. Each AE can detect a test sample as an unseen fall or not based on the reconstruction error and their overall result is combined to take a final decision. We propose to use two types of channel-wise ensemble strategies for detecting unseen falls as follows:

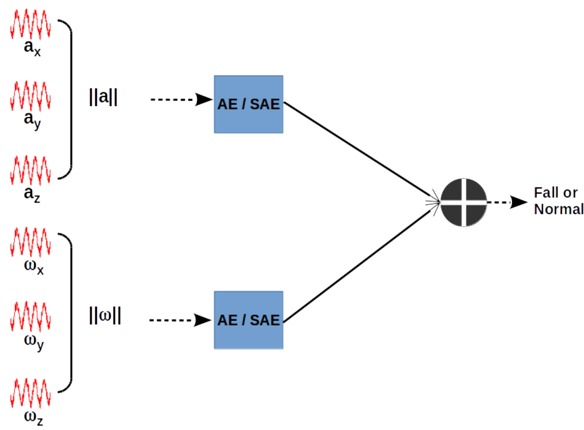
- Six Channel Ensemble ($6\_CE$): For each of the 6 channels of an accelerometer and a gyroscope (i.e., $\mathbf{a_x}, \mathbf{a_y}, \mathbf{a_z}, \boldsymbol{\omega_x}, \boldsymbol{\omega_y}, \boldsymbol{\omega_z}$), 6 separate AE/SAE are trained to learn a compact representation for each channel. A decision threshold is computed separately for each of these 6 AEs to decide whether a test sample is normal or a fall.

- Two Channel Ensemble ($2\_CE$): Alternatively, we can compute the magnitude/norm of the 3 accelerometer channels and that of the 3 gyroscope channels.The magnitude vector gives direction invariant information. The input vector for a time window is the concatenation of these norms.We train two separate AE/SAE to learn a compact representation for each of the two magnitude channels. Thresholding the reconstruction error on these two channels can be used to decide whether a test sample is normal activity or a fall.

For a given test sample, the $6\_CE$ will give 6 different decisions and the $2\_CE$ give 2 decisions. These decisions can be combined by majority voting to arrive at a final decision; as a convention, ties are considered as falls. For simplicity, we keep the hyper-parameters for each AE/SAE corresponding to a channel as the same. The ensemble approach can be faster than the monolithic approach because AE/SAE per channel uses less amount of data in comparison to the combined 6 channel data to a single AE/SAE. Figure 2 shows the graphical representation of the $6\_CE$ and $2\_CE$ approaches.




(a) An Ensemble of 6 separate channels of accelerometer and gyroscope.




(b) An Ensemble of 2 channels of the magnitude of accelerometer and gyroscope.

Figure 2: Channel-wise AE for detecting unseen falls.

## 5. Optimizing the Threshold on the Reconstruction Error

For the fall detection problem, we assume that fall data is rare and is not present during training phase (Khan et al., 2017). Therefore, we train monlithinc and channel-wise AE and SAE on the raw sensor data to learn a compact representation of the normal activities. The next step is to identify a test sample as normal or fall based on the trained AE/SAEs. The typical approach to identify a fall as an anomaly is to set a threshold on the reconstruction error. This threshold is generally set as the maximum of the reconstruction error on the full training data (Dau et al., 2014). We call this threshold *MaxRE*. During testing, any sample that has a reconstruction error greater than this value can be identified as a fall. However, sensor readings are not perfect and may contain spurious data (Khan et al., 2014), which can affect this threshold. Due to the presence of a few outliers in the training data, *MaxRE* is often too large, which could result in many of the falls being missed during testing time. To handle this situation, tightening of threshold is often required (as discussed in Section 3). We use the approach of Erfani et al. (Erfani et al., 2016) that sets the threshold as 3 standard deviations away from the mean of the training data reconstruction error. We call this threshold method as *StdRE*. The *StdRE* threshold can result in identifying more falls during testing in comparison to *MaxRE* at the cost of few false alarms because the threshold in this case is smaller in comparison to *MaxRE*. A problem with *StdRE* is that it is chosen in an ad-hoc manner and it may not be an appropriate choice for a given dataset.

We now present two new approaches to tighten the threshold on reconstruction error. These approaches derive the threshold from the training data such that it can better identify unseen falls. These methods are similar to finding an optimal operating point on an ROC curve by reducing false negatives at the cost of false alarms. However, in a OCC framework, it is difficult to adopt a traditional ROC approach because of the unavailability of the validation data for the negative class. The proposed methods overcome these difficulty by removing outliers from the training data prior to setting a threshold using only the training data.

### *5.1. Reduced Reconstruction Error*

As we discussed earlier, the raw sensor data may not be perfect and may contain spurious or incorrectly labeled readings (Khan et al., 2014). If an AE/SAE is trained on normal activities on such data, the reconstruction error for some of the samples of the training set may be very large. In this case, choosing the maximum of reconstruction error as the threshold to identify falls may lead to accepting most of the falls as normal activities.

Khan et al. (Khan et al., 2017) propose to use the concept of quartiles from descriptive statistic to remove few outliers present in the normal activities. They train an hidden Markov model (HMM) on the normal activities, then compute the log-likelihood of all the training samples, followed by applying IQR technique on these log-likelihood values to find outliers in the training data. We use a similar idea but adapt it to AE/SAE to tighten the threshold on the reconstruction error. We first train an AE/SAE on normal activities, then find the reconstruction error of each training sample. Given the reconstruction error on the training data comprising of only samples of normal activities, the lower quartile ($Q_1$), the upper quartile ($Q_3$) and the inter-quartile range ($IQR = Q_3 - Q_1$), a point $P$ is qualified as an outlier of the normal class, if

$$P > Q_3 + \Omega \times IQR \quad || \quad P < Q_1 - \Omega \times IQR \tag{4}$$

where $\Omega$ is the rejection rate that represents the percentage of data points that are within the non-extreme limits. Based on $\Omega$, the extreme values of reconstruction error that represents spurious training data can be removed and a threshold can be chosen as the maximum of the remaining reconstruction errors. We call this method as Reduced Reconstruction Error (*RRE*). The value of $\Omega$ can be found experimentally or set to remove a small fraction of the normal activities data. We describe a cross-validation technique in Section 5.3 to find *RRE* from only the normal activities.

### *5.2. Inlier Reconstruction Error*

In this method, we first train an AE/SAE on full normal data and then remove the corresponding anomalous training samples based on $\Omega$ from the training set (as discussed in the previous section). After this step, we are left with training data without the outlier samples (or inlier). Then, we train a new AE on this reduced data comprising of just the inlier. The idea is that the variance of reconstruction error for such inlier data will not be too high and its maximum can serve as the new threshold. We call this method as Inlier Reconstruction Error (*IRE*).

For the channel-wise ensemble approach, each AE/SAE is trained only using the raw sensor data from a specific channel of normal activities, then various thresholds, i.e., *MaxRE*, *StdRE*, *RRE* and *IRE* are computed for each channel separately. During testing, for a given threshold method, the final decision is taken as the majority voting outcome of all the AE/SAE. The intuition behind *RRE* and *IRE* is that they should provide a better trade-off between false positive and true positive rate in comparison to *MaxRE* and *StdRE*. The threshold *MaxRE* may work better in detecting only normal activities, whereas *StdRE* is an adhoc approach to reduce the reconstruction error to allow for detection of falls. Both *RRE* and *IRE* are derived from the data and not arbitrarily set to a fixed number. These methods attempt to find a threshold after removing spurious sensor data from the normal activities, which can improve the sensitivity of AE/SAE in detecting unseen falls.

### *5.3. Cross Validation*

The parameter $\Omega$ to tighten the threshold for *RRE* and *IRE* cannot be directly optimized because there is no validation set due to the absence of fall data during training. Khan et al. (Khan et al., 2017) propose to remove some outliers from the normal data and consider them as proxy for unseen falls. They show that rejected outliers from the normal activities can be used to create a validation set and tune parameters of a learning algorithm in the absence of fall data. They use HMMs and show that some of the proxy for falls bear resemblance to actual falls. We modify this idea with respect to the AEs and present a cross-validation method to optimize $\Omega$ in our setting.

Firstly, we train an AE on full normal data and compute reconstruction error of each training example. Then we reject some samples from the normal activities based on a parameter $\rho$, using their reconstruction error. The parameter $\rho$ also uses IQR technique (as discussed in Section 5.1); however it is very different from the parameter $\Omega$. The parameter $\rho$ represents the amount of outlier data removed from normal activities to generate samples for proxy falls to create a validation set. The remaining normal activities are called non-falls. The parameter $\Omega$ represents the amount of reconstruction error removed to set a 'threshold' to identify unseen falls during testing. For a given value of $\rho$, several values of $\Omega$ can be tested and the best is used for further analysis. Therefore, $\rho$ is considered as a hyper-parameter and $\Omega$ as a parameter to find *RRE* and *IRE*. Then the data from both the classes (non-falls and proxy fall) is divided into $K$-folds. The non-fall data from $(K-1)$ folds is combined and an AE is trained on it. The data from $K^{th}$ fold for non-fall and proxy fall is used for testing and tuning the parameters. The process is repeated $K$ times for different values of $\Omega$; the one with the best average performance over $K$ folds is chosen for further analysis. The peformance metric is discussed in Section 6.1. Lastly, for a given $\rho$, we retrain on the non-fall data. The maximum of the reconstruction error corresponding to the best $\Omega$ (obtained in the step discussed above for a given $\rho$) is taken as *RRE*. To compute *IRE*, we remove the outliers from the non-falls corresponding to $\Omega$; then retrain on the reduced training set and take the maximum of the reconstruction error as *IRE*.

The value of hyper-parameter $\rho$ can be varied to observe an overall effect on the performance of the proposed threshold tightening methods, *RRE* and *IRE*. Intuitively, a large value of $\rho$ means less number of samples are removed from normal activities as proxy for fall, which may lead to classify a lot of test samples as normal activities and may miss to identify some falls. Whereas, a small value of $\rho$ means more samples from normal activities may be rejected as proxy for falls; thus, the normal class will be smaller that may result in identifying most of the falls but at the cost of more false alarms. In summary, we exect that with increase in $\rho$, both the true positive rate and false positive rate should reduce (fall is the positive class). By varying $\rho$, we can find an optimum range of operation with a good balance of true positives and false positives. It is to be noted that in this cross-validation method, no actual falls from the training set are used because it is only comprised of normal activities and all the parameters are tuned in the absence of actual falls.

## 6. Experimental Analysis

### *6.1. Performance Metrics*

We consider a case for detecting falls where they are not available during training and occur only during testing. Therefore, during the testing phase, we expect a skewed distribution of falls. Hence, the standard performance metrics such as accuracy may not be appropriate because it may give an over-estimated view of the performance of the classifier. To deal such a case, we use the geometric mean (*gmean*) (Kubat and Matwin, 1997; Khan et al., 2014) as the performance metric to present the test results and optimize the parameters during cross-validation. *gmean* is defined as the square root of the multiplication of true positive and true negative rate, i.e.

$$\begin{aligned} gmean &= \sqrt{TPR * TNR} \\ gmean &= \sqrt{TPR * (1 - FPR)} \end{aligned} \tag{5}$$

where $TPR$ is the true positive rate, $TNR$ is the true negative rate and $FPR$ is the false positive rate. The value of *gmean* varies from 0 to 1, where a 1 means a perfect classification among falls and normal activities and 0 as the worst outcome. We also use the $TPR$ and $FPR$ as other performance metrics to further elaborate our results.

To evaluate the performance of the proposed approaches for fall detection, we perform leave-one-subject-out cross validation (LOOCV) (He and Jin, 2009), where *only* normal activities from $(N-1)$ subjects are used to train the classifiers and the $N^{th}$ subject's normal activities and fall events are used for testing. This process is repeated $N$ times and the average performance metrics are reported. This evaluation is person independent and demonstrates the generalization capabilities as the subject who is being tested is not included in training the classifiers.

### *6.2. Datasets*

We show our results on two activity recognition datasets that includes different normal activities and fall events collected via wearable devices.

#### *6.2.1. German Aerospace Center (DLR) (Nadales, 2010)*

This dataset is collected using an Inertial Measurement Unit with a sampling frequency of 100 Hz. The dataset contains samples from 19 people of both genders of different age groups. The data is recorded in indoor and outdoor environments under semi-natural conditions. The sensor is placed on the belt either on the right or the left side of the body or in the right pocket in different orientations. The dataset contains labelled data of the following 7 activities: Standing, Sitting, Lying, Walking (up/downstairs, horizontal), Running/Jogging, Jumping and Falling. One of the subjects did not perform fall activity; therefore, their data is omitted from the analysis.

#### *6.2.2. Coventry Dataset (COV) (Ojetola et al., 2015)*

This dataset is collected using two SHIMMER™ sensor nodes strapped to the chest and thighs of subjects with a sampling frequency of 100 Hz. Two protocols were followed to collect data from subjects. In Protocol 1, data for six types of fall scenarios are captured (forward, backward, right, left, real fall-backward and real fall forward) and a set of ADL (standing, lying, sitting on a chair or bed, walking, crouching, near falls and lying). Protocol 2 involved ascending and descending stairs. 42 young healthy individuals simulated various ADL and fall scenarios (32 in Protocol 1 and 10 in Protocol 2). The data from different types of falls are joined together to make one separate class for falls. The subjects for Protocol 2 did not record corresponding fall data; therefore, that data is not used. In our analysis, we used accelerometer and gyroscope data from the sensor node strapped to the chest.

### 6.3. Experimental Setup

For both the datasets, all the normal activities are joined together to form a normal class. For COV dataset, different types of falls are joined to make a fall class. The raw sensor data is processed using a 50% overlapping sliding window. The time window size is set to 1.28 seconds for the DLR dataset and 2.56 seconds for the COV dataset (as shown in Khan et al. (Khan et al., 2017)). After pre-processing, the DLR dataset has 26576 normal activities and 84 fall segments, and the COV dataset has 12392 normal activities and 908 fall segments.

We test two types of AE for the analysis; one with a single hidden layer and other with three layered SAE. For the monolithic AE, the raw data within a time window for each of the 3 channels of accelerometer and gyroscope is concatenated, which leads to $768(=128 \times 6)$ input layer neurons for the DLR dataset and $1536(=256 \times 6)$ input layer neurons for the COV dataset. The number of hidden neurons in the monolithic AE is set to 31 (as suggested for the engineered features case in the work of Khan et al. (Khan et al., 2017)). For the monolithic SAE, the number of hidden neurons in the first layer is chosen to be half of the number of input neurons, i.e. 384 for DLR dataset and 768 for COV dataset and the second layer has 31 number of features. For the channel-wise ensemble method, each channels is fed to the AE/SAE separately. Therefore, the number of neurons in the input layer per AE is set to 128 for DLR dataset and 256 for COV dataset and the hidden layers has 31 neurons. For the channel-wise SAE, the hidden neurons for first layer is half the number of input layer, i.e. 64 for DLR dataset and 128 for the COV dataset. The second hidden layer for both the datasets has 31 neurons. The number of training epochs is fixed to 10 for all the different autoencoders. Rest of the parameters such as the sparsity parameter, activation function etc., are kept at the default values (MATLAB, 2017a). Compressed features learned through monolithic AE and SAE are further used to train OSVM and OCNN classifiers (Khan and Madden, 2014) for comparison.

#### 6.3.1. Internal Cross-Validation

For OCNN, the number of nearest neighbours to identify an outlier is kept as 1. OSVM has a parameter $\nu$ (or the outlier fraction), which is the expected proportion of outliers in the training data. The value of this parameter is tuned, similar to parameter optimization discussed in Section 5.3. That is, reject a small portion of normal class data as a proxy for unseen falls for a given $\rho$ and create a validation set. Then perform a $K$ fold cross-validation for different values of $\nu$ and choose the one with the largest average *gmean* over all the $K$-folds. The 'KernelScale' parameter is set to 'auto' and 'Standardize' to 'true', other parameters are kept to default values (MATLAB, 2017b).

An internal $K = 3$-fold cross validation is employed to optimize the parameters $\nu$ for the OSVM and $\Omega$ for the *RRE* and *IRE* thresholding methods. The parameter $\Omega$ is varied from from $[0.001, 0.01, 0.1, 0.5, 1, 1.5, 1.7239, 2, 2.5, 3]$ and $\nu$ is varied from $[0.1, 0.3, 0.5, 0.7, 0.9]$. The best parameter is chosen based on the average *gmean* over $K$ folds. To understand the effect of removing outlier data from the normal activities in building classification models for unseen falls, the hyper-parameter $\rho$ is varied from $[0.001, 0.01, 0.1, 0.5, 1, 1.5, 1.7239, 2, 2.5, 3]$.

Along with 6 channel raw data to train different classifiers to detect unseen falls, we also use 2 channel magnitude data from each of the datasets to train different classifiers. Therefore, in the experiment we compare the following different classifiers for two types of channels data (i.e. 6 and 2 channels):

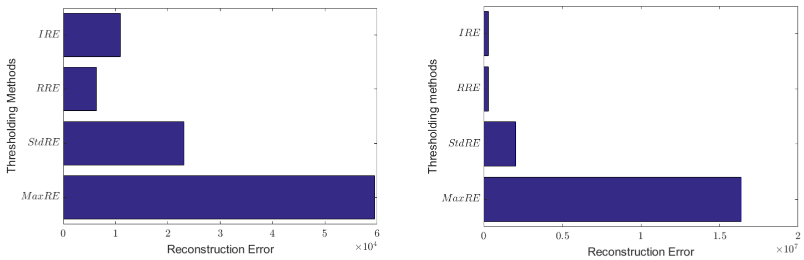
- Two types of AE, i.e. single layer AE and three layered SAE.
- Four types of thresholding methods i.e. *MaxRE*, *StdRE*, *RRE* and *IRE*.
- Two types of feature learning techniques - (i) monolithic and (ii) channel-wise ensemble.
- Two one-class classifiers (OCNN and OSVM ) trained on features learned from AE and SAE (not for the channel-wise case).

This results in 20 different classifiers trained per 6 / 2 channels input raw sensor data; we compare their performance in the next section.

### *6.4. Results and Discussion*

Tables 1 and 2 show the results for the DLR datasets for 6 and 2 channel input raw data. The results correspond to $\rho = 1.5$. Tables 1d and 2d show the results when the features learned using AE and SAE are fed to OSVM and OCNN. We observe that for the 6 channel case, the best *gmean* is obtained for channel-wise AE with *RRE* followed by *IRE* method (see Tables 1a). The traditional thresholding methods of *MaxRE* and *StdRE* do not perform well. We observe similar results for DLR dataset with 2 channel input; however, the *gmean* values using 6 channel input data are higher. Both the *RRE* and *IRE* methods with channel-wise AE give good trade-off between $TPR$ (see Tables 1b and 2b) and $FPR$ (see Tables 1c and 2c). Figure 3a shows the distribution of various thresholds after training on combined normal data of the first $(N - 1)$ subjects. The value of $MaxRE$, which represents the maximum value of reconstruction error on the training set is very large. Due to this large threshold, most of the test samples are classified as normal data. This large value also indicates the presence of some outliers within the normal data, which are hard to reconstruct and result in large reconstruction error. The values of $RRE$ and $IRE$ are much lower than $MaxRE$ and *StdRE* and gives a good balance between $TPR$ and $FPR$ for unseen falls.

Results on the COV dataset for the 6 and 2 channels input data are shown in Table 3 and 4. For the 6 channel case, channel-wise ensemble method with AE for $RRE$ and $IRE$ give equivalent values of *gmean*, which is higher than other methods of thresholding. Both the best methods give a good trade-off between $TPR$ and $FPR$ (see Tables 3b and 3c). Figure 3b shows that $RRE$ and $IRE$ have very similar value and much lower than $MaxRE$ and *StdRE* that leads to good identification of unseen falls. For the 2 channel case, the $IRE$




(a) DLR Dataset

(b) COV dataset

Figure 3: Different thresholds after training a 6-channel monolithic AE on combined normal activities of Subject 1 to $(N-1)$. $RRE$ and $IRE$ are corresponding to $\rho = 1.5$.

threshold method for both the monolithic and channel-wise approaches for AE and SAE give equivalent performance along with monolithic SAE with $RRE$. The channel-wise approach gives more false alarms but detects more falls than the monolithic approach. For both the DLR and COV datasets, the OCNN classifier perform worse than the proposed methods because it gives a large number of false alarms; whereas, OSVM classifies all the test samples as falls (see Tables 1d, 2d, 3d, 4d).

By convention, we classify a test sample as a fall in case of a tie in the channel-wise approach. The probability of a tie occurring is higher in 2 channel ensemble method than in 6 channel ensemble; therefore, its sensitivity to detect falls is higher than the 6 channel case with an increase in the false alarm rate. We observe this behavior for both the DLR and COV datasets (see the Channel-wise rows in Tables 1c, 2c, and 3c, 4c ). From this experiment, we infer that the traditional methods of thresholding, i.e., $MaxRE$ and $StdRE$, are not suitable for the task of fall detection. $MaxRE$ may perform poorly because of the presence of noise in the sensor data that can significantly increase the value of maximum of reconstruction error of an AE/SAE, leading to classify most of the test samples as normal activity. The $StdRE$ is an ad-hoc approach that arbitrarily chooses a threshold lower than the maximum of the reconstruction error of an AE/SAE that may identify more falls. The proposed threshold tightening methods, $RRE$ and $IRE$, attempt to find a discriminating threshold from the training dataset to get a good trade-off between $TPR$ and $FPR$. Our experiments suggest that for both the datasets, $RRE$ and $IRE$ with channel-wise ensemble approach perform equivalently and consistently better than the traditional methods of threshold tightening.

We vary the hyper-parameter $\rho$ to understand its impact on the performance of different thresholding techniques. Figures 4 and 5 show the variation of $TPR$, $FPR$ and $gmean$ with increasing $\rho$. We observe that as the value of $\rho$ increases, both $TPR$ and $FPR$ reduce. The reason is that at smaller values of $\rho$, a large portion of normal data is rejected as outliers and used for parameter tuning; thus, the number of samples in the non-fall class is small. This means that the

(a) *gmean* values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0 | 0.106 | 0.825 | 0.757 |
| | SAE | 0 | 0.234 | 0.840 | 0.837 |
| Channel-wise | AE | 0 | 0.547 | 0.860 | 0.849 |
| | SAE | 0 | 0.334 | 0.818 | 0.811 |

(b) $TPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0 | 0.056 | 0.856 | 0.762 |
| | SAE | 0 | 0.138 | 0.893 | 0.893 |
| Channel-wise | AE | 0 | 0.428 | 0.902 | 0.840 |
| | SAE | 0 | 0.226 | 0.774 | 0.750 |

(c) $FPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 5.9e-6 | 0.025 | 0.189 | 0.169 |
| | SAE | 5.9e-6 | 0.025 | 0.199 | 0.204 |
| Channel-wise | AE | 0 | 0.010 | 0.169 | 0.122 |
| | SAE | 0 | 0.008 | 0.088 | 0.079 |

(d) Performance on OSVM and OCNN methods.

| **Classifier** | **Autoencoder Type** | *gmean* | $TPR$ | $FPR$ |
|---|---|---|---|---|
| OSVM | AE | 0 | 1 | 1 |
| | SAE | 0 | 1 | 1 |
| OCNN | AE | 0.460 | 0.911 | 0.761 |
| | SAE | 0.423 | 0.681 | 0.701 |

Table 1: Performance of different fall detection methods on DLR dataset (6 channels) for $\rho = 1.5$

AE/SAE will learn on a smaller dataset and will reject most of the deviations from this small subset of normal activities as falls. Consequently, many falls will also be identified correctly. The reverse behavior will happen when $\rho$ is

(a) *gmean* values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0 | 0 | 0.504 | 0.774 |
| | SAE | 0 | 0 | 0.630 | 0.776 |
| Channel-wise | AE | 0.013 | 0.487 | 0.839 | 0.822 |
| | SAE | 0 | 0.446 | 0.678 | 0.655 |

(b) $TPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0 | 0 | 0.966 | 0.949 |
| | SAE | 0 | 0 | 0.959 | 0.941 |
| Channel-wise | AE | 0.003 | 0.323 | 0.941 | 0.926 |
| | SAE | 0 | 0.329 | 0.629 | 0.579 |

(c) $FPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 7.4e-5 | 0.037 | 0.705 | 0.363 |
| | SAE | 3.7e-5 | 0.039 | 0.544 | 0.353 |
| Channel-wise | AE | 1.0e-4 | 0.032 | 0.245 | 0.264 |
| | SAE | 7.7e-5 | 0.033 | 0.099 | 0.094 |

(d) Performance on OSVM and OCNN methods.

| **Classifier** | **Autoencoder Type** | $gmean$ | $TPR$ | $FPR$ |
|---|---|---|---|---|
| OSVM | AE | 0 | 1 | 1 |
| | SAE | 0 | 1 | 1 |
| OCNN | AE | 0.459 | 0.815 | 0.719 |
| | SAE | 0.318 | 0.317 | 0.449 |

Table 2: Performance of different fall detection methods on DLR dataset (2 channels) for $\rho = 1.5$

large, less number of normal data is rejected as outliers and the class of normal activities will be large. This will reduce the number of false alarms but can also lead to missing to identify some falls. The experimental observation for each of

(a) *gmean* values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0.015 | 0.744 | 0.774 | 0.771 |
| | SAE | 0.019 | 0.743 | 0.772 | 0.771 |
| Channel-wise | AE | 0.014 | 0.463 | 0.795 | 0.795 |
| | SAE | 0 | 0.226 | 0.737 | 0.707 |

(b) $TPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0.004 | 0.589 | 0.744 | 0.740 |
| | SAE | 0.007 | 0.588 | 0.738 | 0.738 |
| Channel-wise | AE | 0.003 | 0.248 | 0.7 | 0.7 |
| | SAE | 0 | 0.082 | 0.665 | 0.573 |

(c) $FPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 1.1e-4 | 0.017 | 0.169 | 0.169 |
| | SAE | 1.1e-4 | 0.017 | 0.166 | 0.167 |
| Channel-wise | AE | 0 | 0.002 | 0.067 | 0.072 |
| | SAE | 0 | 5.9e-5 | 0.128 | 0.078 |

(d) Performance on OSVM and OCNN methods.

| **Classifier** | **Autoencoder Type** | *gmean* | $TPR$ | $FPR$ |
|---|---|---|---|---|
| OSVM | AE | 0 | 1 | 1 |
| | SAE | 0 | 1 | 1 |
| OCNN | AE | 0.432 | 0.977 | 0.805 |
| | SAE | 0.484 | 0.949 | 0.751 |

Table 3: Performance of different fall detection methods on COV dataset (6 channels) for $\rho = 1.5$

the 6 and 2 channel datasets is consistent with this intuition discussed in Section 5.3. Similar observation can be made for the COV dataset from Figures 6 and 7. For both datasets, we notice that at large value of $\rho$, the performance of best

(a) *gmean* values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0.041 | 0.743 | 0.668 | 0.785 |
| | SAE | 0.019 | 0.724 | 0.784 | 0.784 |
| Channel-wise | AE | 0.337 | 0.767 | 0.726 | 0.788 |
| | SAE | 0.331 | 0.757 | 0.739 | 0.786 |

(b) $TPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 0.012 | 0.587 | 0.729 | 0.698 |
| | SAE | 0.007 | 0.557 | 0.677 | 0.674 |
| Channel-wise | AE | 0.147 | 0.621 | 0.805 | 0.779 |
| | SAE | 0.142 | 0.606 | 0.781 | 0.779 |

(c) $FPR$ values.

| **Features Types** | **Autoencoder Type** | **Thresholding** | | | |
|---|---|---|---|---|---|
| | | ***MaxRE*** | ***StdRE*** | ***RRE*** | ***IRE*** |
| Monolithic | AE | 1.7e-4 | 0.013 | 0.287 | 0.094 |
| | SAE | 1.7e-4 | 0.012 | 0.06 | 0.056 |
| Channel-wise | AE | 1.1e-4 | 0.015 | 0.298 | 0.186 |
| | SAE | 1.1e-4 | 0.016 | 0.255 | 0.182 |

(d) Performance on OSVM and OCNN methods.

| **Classifier** | **Autoencoder Type** | $gmean$ | $TPR$ | $FPR$ |
|---|---|---|---|---|
| OSVM | AE | 0 | 1 | 1 |
| | SAE | 0 | 1 | 1 |
| OCNN | AE | 0.594 | 0.905 | 0.606 |
| | SAE | 0.580 | 0.606 | 0.432 |

Table 4: Performance of different fall detection methods on COV dataset (2 channels) for $\rho = 1.5$

thresholding approaches drops slower. This experimental observation suggests that a smaller amount of data (corresponding to $\rho \geq 1.5$) may be removed from the normal activities class as outliers, which can be used as a validation set to optimize the parameters of the AE/SAE and better performance can be achieved for identifying unseen falls. We also infer that channel-wise approach outperforms monolithic in all the 6 and 2 channel data variants of both the datasets.

## 7. Conclusions and Future Work

A fall is a rare event; therefore, it is difficult to build classification models using traditional supervised algorithms in the absence of training data. An associated challenge for fall detection problem is to extract discriminative features in the absence of fall data for training generalizable classifiers. In this paper, we presented solutions to deal with these issues. Firstly, we formulated a fall detection problem as a OCC or anomaly detection problem. Secondly, we presented the use of AE, more specifically a novel way to train separate AE for each channel of the wearable sensor, to learn generic features and create their ensemble. We proposed threshold tightening methods to identify unseen falls accurately. This work provides useful insights that an ensemble based on channels of a wearable device with optimized threshold is a useful technique to identify unseen falls. In future work, we are exploring extreme value theory and combining it with the proposed approaches to identify unseen falls. We are also interested in tuning the parameters of autoencoders (such as learning rate, activation functions, network topology etc) using the IQR technique discussed in the paper to improve the performance of the proposed ensemble methods in detecting unseen falls.

## Acknowledgments

This work was partially supported by the AGE-WELL NCE Trainee Award Program and by the Canadian Consortium on Neurodegeneration in Aging (CCNA).

## References

A. Budiman, M. I. Fanany, and C. Basaruddin. Stacked denoising autoencoder for feature representation learning in pose-based action recognition. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pages 684–688, Oct 2014.

Hoang Anh Dau, Vic Ciesielski, and Ady Song. *Simulated Evolution and Learning: 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings*, chapter Anomaly Detection Using Replicator Neural Networks Trained on Examples of One Class, pages 311–322. Springer International Publishing, Cham, 2014.

Yue Dong and Nathalie Japkowicz. *Advances in Artificial Intelligence: 29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Victoria, BC, Canada, May 31 - June 3, 2016. Proceedings*, chapter Threaded Ensembles of Supervised and Unsupervised Neural Networks for Stream Learning, pages 304–315. Springer International Publishing, Cham, 2016.

Nashwa El-Bendary, Qing Tan, Frédérique C Pivot, and Anthony Lam. Fall detection and prevention for the elderly: A review of trends and challenges. *International Journal on Smart Sensing and Intelligent Systems*, 6(3):1230–1266, 2013.

Sarah M. Erfani, Sutharshan Rajasegarar, Shanika Karunasekera, and Christopher Leckie. High-dimensional and large-scale anomaly detection using a linear one-class {SVM} with deep learning. *Pattern Recognition*, 58:121 – 134, 2016.

Zhenyu He and Lianwen Jin. Activity recognition from acceleration data based on discrete consine transform and svm. In *SMC*, pages 5041–5044. IEEE, 2009.

Raul Igual, Carlos Medrano, and Inmaculada Plaza. Challenges, issues and trends in fall detection systems. *BioMedical Engineering OnLine*, 12(1):1–24, 2013.

Vamsi K Ithapu, Vikas Singh, Ozioma Okonkwo, and Sterling C Johnson. Randomized denoising autoencoders for smaller and efficient imaging based ad clinical trials. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2014*, pages 470–478. Springer, 2014.

Stanislaw Jankowski, Zbigniew Szymanski, Uladzimir Dziomin, Pawel Mazurek, and Jakub Wagner. Deep learning classifier for fall detection based on ir distance sensor data. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on*, volume 2, pages 723–727. IEEE, 2015.

Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *Proceedings of the 14th international joint conference on Artificial intelligence-Volume 1*, pages 518–523. Morgan Kaufmann Publishers Inc., 1995.

Branka Jokanovic, Moeness Amin, and Fauzia Ahmad. Radar fall motion detection using deep learning. In *2016 IEEE Radar Conference (RadarConf)*, pages 1–6, 2016.

Muhammad Salman Khan, Miao Yu, Pengming Feng, Liang Wang, and Jonathon Chambers. An unsupervised acoustic fall detection system using source separation for sound interference suppression. *Signal Processing*, 110: 199 – 210, 2015. ISSN 0165-1684. Machine learning and signal processing for human pose recovery and behavior analysis.

Shehroz Khan. *Classification and decision-theoretic framework for detecting and reporting unseen falls*. PhD thesis, University of Waterloo, 2016.

Shehroz S Khan and Jesse Hoey. Review of fall detection techniques: A data availability perspective. *Medical Engineering & Physics*, 39:12–22, 2017.

Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29: 345–374, 2014. ISSN 1469-8005.

Shehroz S. Khan, Michelle E. Karg, Dana Kulić, and Jesse Hoey. X-factor HMMs for detecting falls in the absence of fall-specific training data. In L. Pecchia et al., editor, *Proceedings of the 6th International Work-conference on Ambient Assisted Living (IWAAL 2014)*, volume 8868, pages 1–9, Belfast, U.K., Dec 2014. Springer International Publishing Switzerland.

Shehroz S Khan, Michelle E Karg, Dana Kulić, and Jesse Hoey. Detecting falls with x-factor hidden markov models. *Applied Soft Computing*, 55:168–177, 2017.

Miroslav Kubat and Stan Matwin. Addressing the curse of imbalanced training sets: One-sided selection. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

Yongmou Li, Dianxi Shi, Bo Ding, and Dongbo Liu. *Mining Intelligence and Knowledge Exploration: Second International Conference, MIKE 2014, Cork, Ireland, December 10-12, 2014. Proceedings*, chapter Unsupervised Feature Learning for Human Activity Recognition Using Smartphone Sensors, pages 99–107. Springer International Publishing, Cham, 2014a.

Yongmou Li, Dianxi Shi, Bo Ding, and Dongbo Liu. Unsupervised feature learning for human activity recognition using smartphone sensors. In *Mining Intelligence and Knowledge Exploration*, pages 99–107. Springer, 2014b.

Larry Manevitz and Malik Yousef. One-class document classification via neural networks. *Neurocomputing*, 70(79):1466 – 1481, 2007. Advances in Computational Intelligence and Learning14th European Symposium on Artificial Neural Networks 200614th European Symposium on Artificial Neural Networks 2006.

E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1996–2000, April 2015a.

Erik Marchi, Fabio Vesperini, Felix Weninger, Florian Eyben, Stefano Squartini, and Björn Schuller. Non-linear prediction with lstm recurrent neural networks for acoustic novelty detection. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015b.

MATLAB. Train autoencoder. `http://www.mathworks.com/help/nnet/ref/trainautoencoder.html`, 2017a. Accessed on $23^{rd}$ February, 2017.

MATLAB. Train binary support vector machine classifier. `https://www.mathworks.com/help/stats/fitcsvm.html`, 2017b. Accessed on $23^{rd}$ February, 2017.

Carlos Medrano, Raul Igual, Inmaculada Plaza, and Manuel Castro. Detecting falls as novelties in acceleration patterns acquired with smartphones. *PloS one*, 9(4):e94811, 2014.

Daniela Micucci, Marco Mobilio, Paolo Napoletano, and Francesco Tisato. Falls as anomalies? an experimental evaluation using smartphone accelerometer data. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2015.

Maria Josefa Vera Nadales. Recognition of human motion related activities from sensors. Master's thesis, University of Malaga and German Aerospace Cener, 2010.

Olukunle Ojetola, Elena Gaura, and James Brusey. Data set for fall events and daily activities from inertial sensors. In *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 243–248, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3351-1.

World Health Organization. Falls fact sheet, reviewed september 2016. `http://www.who.int/mediacentre/factsheets/fs344/en/`, 2016. Accessed on $7^{th}$ March 2017.

Ahmet Turan Özdemir and Billur Barshan. Detecting falls with wearable sensors using machine learning techniques. *Sensors*, 14(6):10691–10708, 2014.

Natthapon Pannurat, Surapa Thiemjarus, and Ekawit Nantajeewarawat. Automatic fall monitoring: a review. *Sensors*, 14(7):12900–12936, 2014.

Thomas Plötz, Nils Y. Hammerla, and Patrick Olivier. Feature learning for activity recognition in ubiquitous computing. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Two*, IJCAI'11, pages 1729–1734. AAAI Press, 2011. ISBN 978-1-57735-514-4.

M. Popescu and A. Mahnot. Acoustic fall detection using one-class classifiers. In *Annual International Conference of the IEEE EMBC*, pages 3505–3508, Sept 2009.

Emanuele Principi, Diego Droghini, Stefano Squartini, Paolo Olivetti, and Francesco Piazza. Acoustic cues from the floor: A new approach for fall classification. *Expert Systems with Applications*, 60:51–61, 2016.

Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, IAAI'05, pages 1541–1546. AAAI Press, 2005. ISBN 1-57735-236-x.

Henry W. J. Reeve and Gavin Brown. Modular autoencoders for ensemble feature extraction. In *NIPS 2015 Workshop on Feature Extraction: Modern Questions and Challenges.*, pages 242–259, 2015.

Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2Nd Workshop on Machine Learning for Sensory Data Analysis*, MLSDA'14, pages 4:4–4:11, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3159-3.

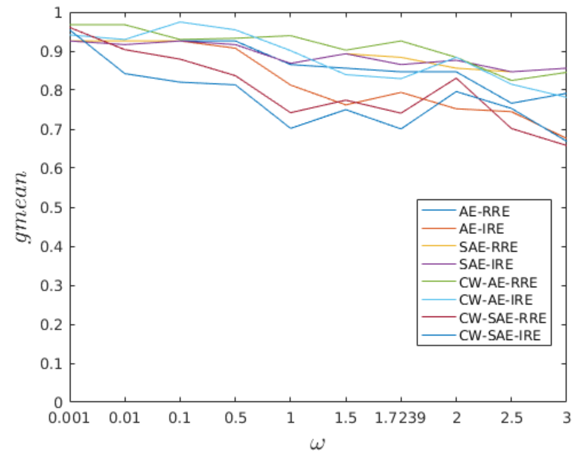
Matthias Scholz and Ricardo Vigário. Nonlinear pca: a new hierarchical approach. In *ESANN*, pages 439–444, 2002.

E.E. Stone and M. Skubic. Fall detection in homes of older adults using the microsoft kinect. *Biomedical and Health Informatics, IEEE Journal of*, 19(1): 290–301, Jan 2015.

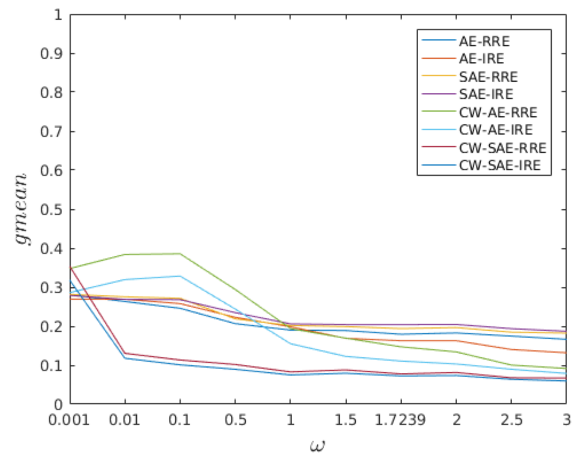
Do-Dat Tran, Thi-Lan Le, et al. Abnormal event detection using multimedia information for monitoring system. In *Communications and Electronics (ICCE), 2014 IEEE Fifth International Conference on*, pages 490–495. IEEE, 2014.

Lukun Wang. Recognition of human activities using continuous autoencoders with wearable sensors. *Sensors*, 16(2):189, 2016.

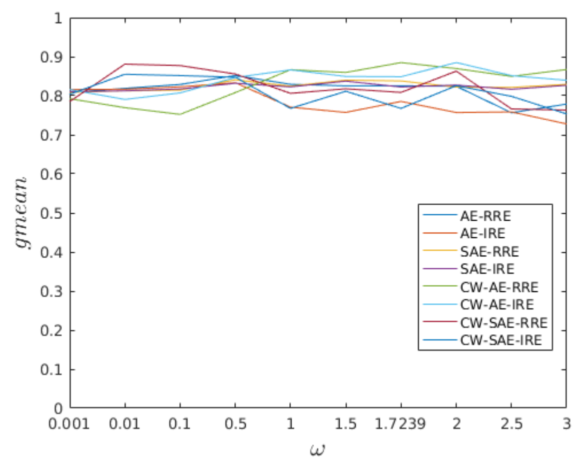
Min Zhou, Shuangquan Wang, Yiqiang Chen, Zhenyu Chen, and Zhongtang Zhao. An activity transition based fall detection model on mobile devices. In *Human Centric Technology and Service in Smart Space*, volume 182 of *Lecture Notes in Electrical Engineering*, pages 1–8. Springer Netherlands, 2012. ISBN 978-94-007-5085-2.




(a) True Positive Rate

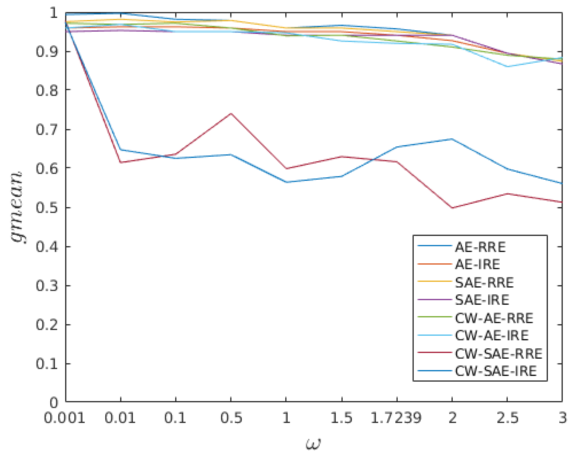



(b) False Positive Rate

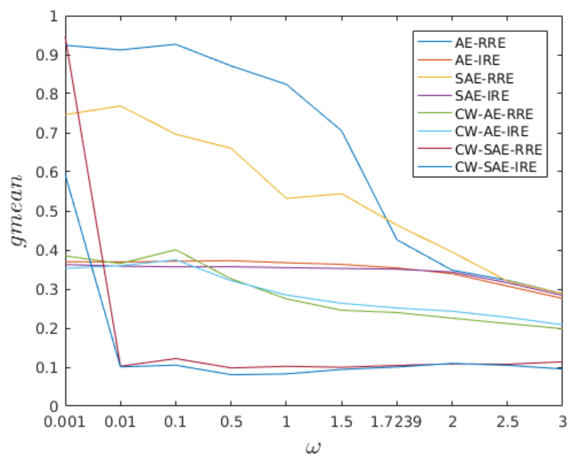



(c) *gmean*

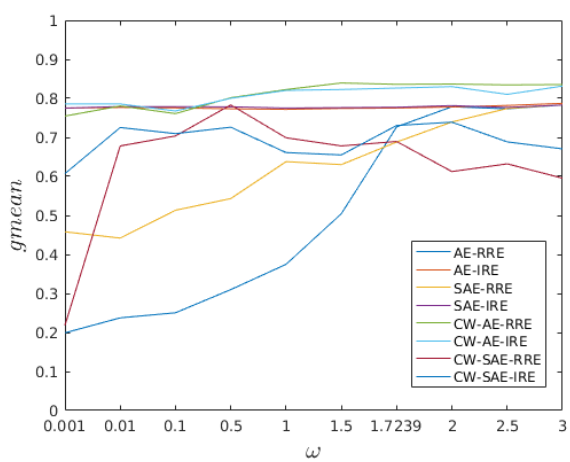
Figure 4: Performance of top 5 fall detection methods by varying $\rho$ on DLR dataset. AE - Single Layer Autoencoder, SAE - 3 Layer Stacked Autoencoder, RRE - Reduced Reconstruction Error, IRE - Inlier Reconstruction Error, CW - Channel-wise Ensemble




(a) True Positive Rate

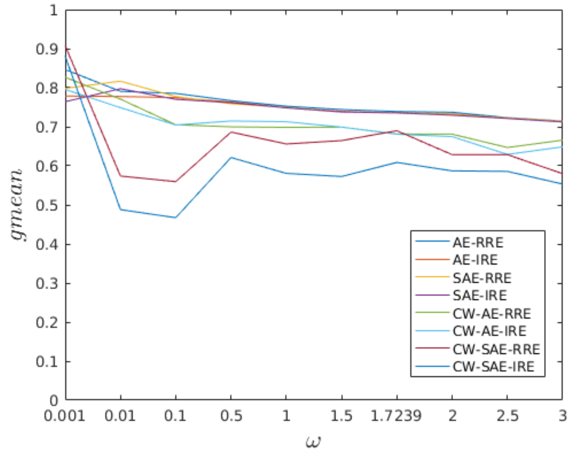



(b) False Positive Rate

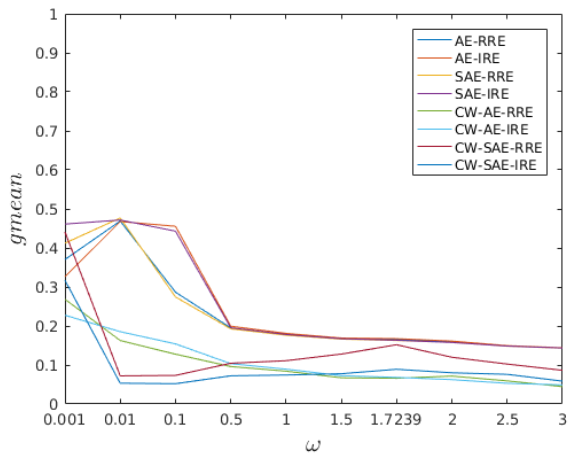



(c) *gmean*

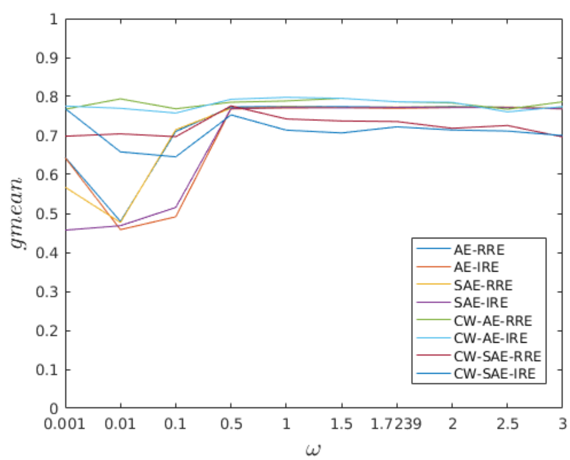
Figure 5: Performance of top 5 fall detection methods by varying $\rho$ on DLR-norm dataset. AE - Single Layer Autoencoder, SAE - 3 Layer Stacked Autoencoder, RRE - Reduced Reconstruction Error, IRE - Inlier Reconstruction Error, CW - Channel-wise Ensemble




(a) True Positive Rate

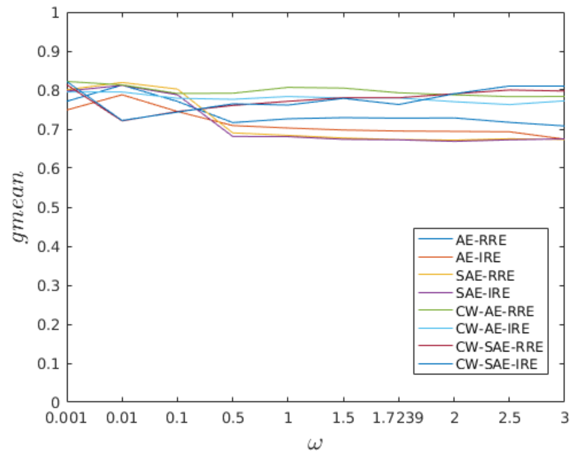



(b) False Positive Rate

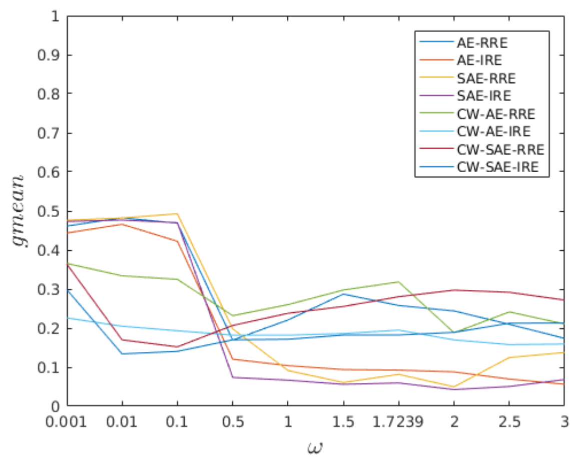



(c) *gmean*

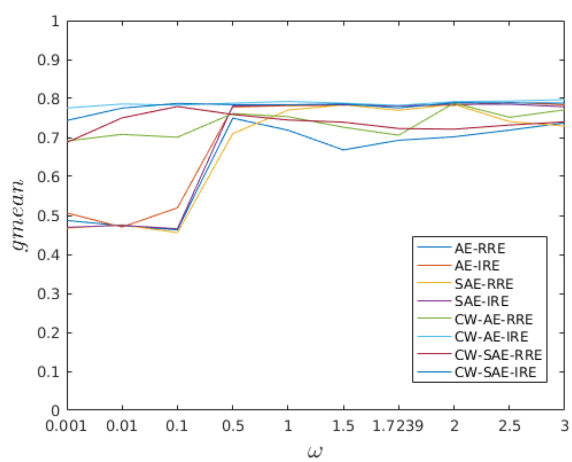
Figure 6: Performance of top 5 fall detection methods by varying $\rho$ on COV dataset. AE - Single Layer Autoencoder, SAE - 3 Layer Stacked Autoencoder, RRE - Reduced Reconstruction Error, IRE - Inlier Reconstruction Error, CW - Channel-wise Ensemble




(a) True Positive Rate




(b) False Positive Rate




(c) *gmean*

Figure 7: Performance of top 5 fall detection methods by varying $\rho$ on COV-norm dataset. AE - Single Layer Autoencoder, SAE - 3 Layer Stacked Autoencoder, RRE - Reduced Reconstruction Error, IRE - Inlier Reconstruction Error, CW - Channel-wise Ensemble