Probability in the Engineering and Informational Sciences, **35**, 2021, 37–50.
doi:10.1017/S0269964819000317

HRNN4F: HYBRID DEEP RANDOM NEURAL NETWORK FOR MULTI-CHANNEL FALL ACTIVITY DETECTION

AHSEN TAHIR
School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow, UK
Department of Electrical Engineering, University of Engineering and Technology, Lahore, Pakistan
E-mails: ahsen.tahir@gcu.ac.uk; ahsan@uet.edu.pk

JAWAD AHMAD, GORDON MORISON, HADI LARIJANI and RYAN M. GIBSON
School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow, UK

DAWN A. SKELTON
School of Health and Life Sciences, Glasgow Caledonian University, Glasgow, UK

Falls are a major health concern in older adults. Falls lead to mortality, immobility and high costs to social and health care services. Early detection and classification of falls is imperative for timely and appropriate medical aid response. Traditional machine learning models have been explored for fall classification. While newly developed deep learning techniques have the ability to potentially extract high-level features from raw sensor data providing high accuracy and robustness to variations in sensor position, orientation and diversity of work environments that may skew traditional classification models. However, frequently used deep learning models like Convolutional Neural Networks (CNN) are computationally intensive. To the best of our knowledge, we present the first instance of a Hybrid Multichannel Random Neural Network (HMCRNN) architecture for fall detection and classification. The proposed architecture provides the highest accuracy of 92.23% with dropout regularization, compared to other deep learning implementations. The performance of the proposed technique is approximately comparable to a CNN yet requires only half the computation cost of the CNN-based implementation. Furthermore, the proposed HMCRNN architecture provides 34.12% improvement in accuracy on average than a Multilayer Perceptron.

Keywords: probabilistic networks, queueing theory, simulation

1. INTRODUCTION

Falls are a significant cause of death and injury in older adults [35]. 43% of the older adults fall in a span of 5 years and 25% of those living alone in a span of 1 year [6]. 62% of

© Cambridge University Press 2019 0269-9648/19 $25.00

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

fall incidents cause injuries resulting in lack of mobility and high mortality rates [6,25]. Falls incur high costs to health care services [35]. Early classification and detection of falls is imperative for life saving health response services. Fall detection systems are known to reduce death incidents by 80% due to prompt medical aid and hospitalization [28].

Current fall detection systems have a significant focus on traditional machine learning algorithms, such as decision tree, random forest algorithm, k-nearest neighbor, support vector machine and principal component analysis [16,17,38]. While deep learning techniques offer significant benefits over traditional machine learning classification models [39]. Deep learning has the ability to potentially extract high level features from raw accelerometer signals [43] within a context-aware setting [2] and can provide a robust and accurate method to variations in work environments, sensor positions and orientations [5,22,26]. For example, a model developed for a waist band sensor with traditional machine learning techniques may not apply to the arm or wrist band sensor because of the variations in acceleration values across different parts of the body [5]. Similarly, a mobile phone-based sensor classification model may not work well with variations in mobile phone position and orientation, since sensors generate different signals in different contexts, to which traditional machine learning models do not provide robust and accurate classification [7,22]. Variability in activities of individuals may also play a role in accuracy of the classification model, for example certain physically demanding work and sports environments may result in wrist band accelerometer values not related to falls and may skew the classification model [22]. Deep learning models have been proposed for fall detection, including Convolutional Neural Networks (CNN) [4,30], combination of CNN and Long Short-Term Memory (LSTM) [23,27,29] and AutoEncoders (AE) [20]. However, the deep learning techniques are computationally intensive both in terms of time cost and hardware resources [9]. Random Neural Networks (RNN) introduced by Gelenbe [10] have been reported to have exact and simple computational algorithms due to their "product form" [14]. Gelenbe et al. introduced the deep learning architectures for RNNs in [14] and improved it for a multichannel dataset in [15].

In this paper, we propose a Hybrid Multichannel RNN (HMCRNN) architecture for fall detection and classification, but without the dense clusters of the Multichannel RNN (MCRNN) in [15]. In contrast to the use of dense clusters, the proposed design is simple and computationally inexpensive for classification of 1D sensor signals for fall activity. The multichannel hybrid architecture consists of RNN layers and fully connected feed forward Multilayer Perceptron (MLP) with Rectified Linear activation Units (ReLU). The underlying concept of the proposed architecture is for the RNN layers to provide low-dimensional feature space relevant to each accelerometer axis, separately. While, merging at the fully connected MLP with ReLU activation functions is used to learn a non-linear function over the entire extracted feature space. It has the effect of diminishing non-relevant features from accelerometer axes representing movement dimensions that may be least relevant to activities and falls. The accelerometer data used for our work consists of values for three axes of motion a_x, a_y and a_z for falls and various activities of daily life. The data from the three accelerometer axes (128 sample window each) are input to each of the channels in the proposed architecture resulting in an input layer of size 3×128 and a fall or no fall decision is obtained from the network. The proposed scheme is discussed in Section 4. The methodology used for training and testing of the proposed architecture is discussed in Section 5. Results are tabulated and discussed in Section 6 where computational time cost and performance accuracy is determined and compared with CNN and MLP implementations. The proposed HMCRNN scheme is also compared with a MCRNN architecture with the same structure and size. The next Section discusses relevant related work.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

2. RELATED WORK

There have been a number of recent applications of deep learning techniques for activity and fall detection, including CNN and LSTM [37]. Quero et al. [30] detect falls from vision-based sensors and apply three different CNN models to raw data. The work also evaluates the effect of kernel size and number of layers on classification accuracy and reports up to 92% accuracy with three CNN layers. Lu et. al [23] developed a 3D CNN for fall detection, which used both spatial and temporal information for video classification. In addition, the output of 3D CNN is used as an input for the LSTM network for higher accuracy. ConvLSTM [27] applies deep learning to raw accelerometer data for fall risk assessment and compares the performance of three deep learning models CNN, LSTM and a combination of CNN with LSTM. However, our work is based on fall event detection rather than fall risk assessment. Chen and Xue [4] performed multi-class classification on accelerometer signals for falls and daily activities of life with a CNN consisting of three convolutional and three pool layers. Ordóñez and Roggen [29] utilized a CNN in addition with a recurrent LSTM for multimodal activities dataset including falls from a number of sensors. However, a major draw back of CNNs is that they are computationally intensive [34].

Deep learning with RNN was introduced by Gelenbe Yin in [14], which has been further explored by Yin and Gelenbe in [15,40–42]. The work in [14,15,40] is based on the concept that the structure of human brain is composed of clusters of dense nuclei. They developed a mathematical model based on dense clusters also known as the "Dense" RNN, where each cluster is composed of neurons with statistically identical structure of connections implemented as a spiking recurrent RNN. A training procedure based on unsupervised and supervised learning is also proposed, which converts training into a convex optimization problem that can utilize better or comparable solutions for higher speed convergence than the gradient descent algorithm. In [15,40] a Multichannel architecture of 'Dense' RNN is proposed and applied for different applications, such as recognition of 3D objects, classification of chemical gases and activities of daily life. Recent advancements have implemented RNN models to predict the toxicity levels of chemical compounds based on their physical-chemical structure [18]. In [3] a deep learning model based on dense RNNs for the detection of denial-of-service network attacks against Internet of Things gateways is presented. Our proposal utilizes the original single cell architecture of RNN [18], however with a HMCRNN and MLP network for improved accuracy. Furthermore, we achieve high accuracy comparable to CNN without the dense RNN clusters and the proposed learning algorithms in [3,15,40]. Our work utilizes RMSProp algorithm introduced by Hinton [19], which adapts resilient propagation (RProp) [31] algorithm for stochastic gradient descent (SGD) [32]. Riedmiller Braun [31] have utilized RProp algorithm for learning in RNN. While, Basterrech et al. [1] combined RProp with the gradient descent algorithm and used the Levenberg-Marquardt optimization and momentum to achieve high convergence speeds than the gradient descent algorithm.

Our work is based on a MCRNN implementation, however for our fall classification problem which has raw accelerometer window sizes of 128 samples, two RNN hidden layers are utilized for extracting a significant feature space. This paper presents a hybrid implementation of MLP and RNN to achieve accuracies comparable to a CNN implementation. We achieve an added advantage of lower computational time complexity than CNN by not using dense clusters and achieve comparable performance in accuracy to CNN architecture with the RMSProp [19] training algorithm.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

3. RANDOM NEURAL NETWORK

The RNN was introduced in groundbreaking work [10] and the behavior of large RNN networks was first considered in [13]. In RNN, the state of a neuron is described by its potential, which is a non-negative integer and represents the accumulation of signals. Neurons in RNN layers exchange excitatory and inhibitory spiking signals probabilistically. The excitatory and inhibitory spikes are represented by $+1$ and -1, respectively. Hence, given a potential $v_n(t)$ of a neuron n at time t, then $v_n(t) \in \mathbb{R}_0^+$, where an inhibitory spike can only cancel a positive signal and has no effect if $v_n(t) = 0$. If $v_n(t) = 0$, neuron n is in an idle state, otherwise if $v_n(t) > 0$ neuron n is in excited state. The neuron in RNN transmits signals randomly, according to an exponential distribution with firing rate $r(n)$ and mean value of times between signals as $1/r(n)$. The signals are transmitted to the next neuron m either as a positive excitatory signal with probability $p^+(n,m)$ or as a negative inhibitory signal with probability $p^-(n,m)$. The signal may also leave the network with probability $d(n)$. The sum of all probabilities must be equal to unity, mathematically:

$$\sum_{m=1}^{N} [p^+(n,m) + p^-(n,m)] + d(n) = 1, \forall n \tag{1}$$

where N represents the total number of neurons. Given the arrival rates of external excitation and inhibition as $\Lambda(n)$ and $\lambda(n)$ respectively, the probability of firing of neuron n is [36]:

$$q_n = \frac{\lambda^+(n)}{r(n) + \lambda^-(n)} \tag{2}$$

where,

$$\lambda^+(n) = \sum_{m=1}^{N} q_m r(m) p^+(m,n) + \Lambda(n) \tag{3}$$

$$\lambda^-(n) = \sum_{m=1}^{N} q_m r(m) p^-(m,n) + \lambda(n) \tag{4}$$

The output q_n is an activation function of excitatory inputs $\lambda^+(n)$ divided by a sum of inhibitory inputs $\lambda^-(n)$ and a firing rate $r(n)$. $w^+(n,m)$ and $w^-(n,m)$ are non-negative rates for positive and negative signals, respectively, represented as:

$$w^+(n,m) = r(n) p^+(n,m) \geq 0 \tag{5}$$

$$w^-(n,m) = r(n) p^-(n,m) \geq 0 \tag{6}$$

Expression for firing rate $r(n)$ can be derived from Eqs. (1), (5) and (6), as:

$$r(n) = (1 - d(n))^{-1} \sum_{m=1}^{N} [w^+(n,m) + w^-(n,m)] \tag{7}$$

$$\lambda^+(n) < [r(n) + \lambda^-(n)] \tag{8}$$

where Eq. (8) is a sufficient condition for existence of unique solution in RNN. The values $w^+(n,m)$ and $w^-(n,m)$ represented by Eqs. (5) and (6) are equivalent to weights in a classical neural network (Gelenbe [10]) and can be trained using traditional learning algorithms, such as gradient descent.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

3.1. Gradient Descent Algorithm for RNN

A standard gradient descent algorithm for training RNN is presented in this section based on the work in [11] and generalized in [12]; an early application of this algorithm is presented in [8]. Let the training set (X, Y) consist of L input-output pairs, where $X = \{\boldsymbol{x_1}, \ldots, \boldsymbol{x_L}\}$ are successive inputs and $Y = \{\boldsymbol{y_1}, \ldots, \boldsymbol{y_L}\}$ are the respective output vectors, where each vector $\boldsymbol{y_l} = (y_{1l}, \ldots, y_{Nl})$ and each element $y_{nl} \in [0, 1]$. The l^{th} input pattern $\boldsymbol{x_l}$ is represented by the pair $(\boldsymbol{\Lambda_l}, \boldsymbol{\lambda_l})$ of positive and negative flow rates, where $\boldsymbol{\Lambda_l} = [\Lambda_l(1), \ldots, \Lambda_l(N)]$ and $\boldsymbol{\lambda_l} = [\lambda_l(1), \ldots, \lambda_l(N)]$ are vectors. The x_{nl} data input of the l^{th} training pattern is:

$$\begin{cases} \Lambda_l(n) > 0, \lambda_l(n) = 0 & \text{If } x_{nl} > 0 \\ \Lambda_l(n) = 0, \lambda_l(n) > 0 & \text{If } x_{nl} \leq 0 \end{cases} \tag{9}$$

The error cost function E_l of l^{th} input-output pair for the gradient descent algorithm is:

$$E_l = \frac{1}{2} \sum_{n=1}^{N} \alpha_n (q_n - y_{nl})^2, \alpha_n \geq 0 \tag{10}$$

where $\alpha_n \in [0, 1]$ decides whether neuron n is an output neuron, q_n is a differentiable function and y_{nl} is the desired value. The function of gradient descent is to minimize the error cost function described in Eq. (10). Let two RNs a and b be connected, then weights $w^+(a, b)$ and $w^-(a, b)$ are updated according to the expression:

$$w_t^+(a, b) = w_{t-1}^+(a, b) - \eta \sum_{n=1}^{N} \alpha_n (q_n - y_{nl}) [\partial q_n / \partial w^+(a, b)]_{t-1} \tag{11}$$

$$w_t^-(a, b) = w_{t-1}^-(a, b) - \eta \sum_{n=1}^{N} \alpha_n (q_n - y_{nl}) [\partial q_n / \partial w^-(a, b)]_{t-1} \tag{12}$$

where η is the learning rate, $\partial q_n / \partial w^+(a, b)$ and $\partial q_n / \partial w^-(a, b)$ are the activation function derivatives with respect to weights, for which the matrix form is proved by Gelenbe in [11]. The partial derivative terms in Eqs. (11) and (12) can be computed by defining a vector $\mathbf{q} = (q_1, \ldots, q_N)$ and a matrix $\mathbf{W}$ of size $N \times N$, where $\mathbf{W}$ is:

$$\mathbf{W} = [w^+(n, m) - w^-(n, m) q_m] / [r(m) + \lambda^-(m)], \ n, m = 1, \ldots, N. \tag{13}$$

Also, defining vectors $\boldsymbol{\gamma}^+(a, b)$ and $\boldsymbol{\gamma}^-(a, b)$ with N entries, where each entry n is defined as:

$$\gamma_n^+(a, b) = \begin{cases} \dfrac{-1}{r(n) + \lambda^-(n)} & \text{if } a = n, b \neq n \\ \dfrac{1}{r(n) + \lambda^-(n)} & \text{if } a \neq n, b = n \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

$$\gamma_n^-(a,b) = \begin{cases} \dfrac{-1+q_n}{r(n)+\lambda^-(n)} & \text{if } a=n, b=n \\ \dfrac{-1}{r(n)+\lambda^-(n)} & \text{if } a=n, b\neq n \\ \dfrac{-q_n}{r(n)+\lambda^-(n)} & \text{if } a\neq n, b=n \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

One can use the above notation to derive from Eq. (2) the following vector equations:

$$\partial \boldsymbol{q}/\partial w^+(a,b) = \partial \boldsymbol{q}/\partial w^+(a,b)\ \mathbf{W} + \boldsymbol{\gamma}^+(a,b) q_a \tag{16}$$

$$\partial \boldsymbol{q}/\partial w^-(a,b) = \partial \boldsymbol{q}/\partial w^-(a,b)\ \mathbf{W} + \boldsymbol{\gamma}^-(a,b) q_a \tag{17}$$

Equations (16) and (17) can be equivalently written as:

$$\partial \boldsymbol{q}/\partial w^+(a,b) = \boldsymbol{\gamma}^+(a,b) q_a [\mathbf{I}-\mathbf{W}]^{-1} \tag{18}$$

$$\partial \boldsymbol{q}/\partial w^-(a,b) = \boldsymbol{\gamma}^-(a,b) q_a [\mathbf{I}-\mathbf{W}]^{-1} \tag{19}$$

where $\mathbf{I}$ is a size $N \times N$ identity matrix. The training algorithm stops when a certain defined convergence criteria is met or maximum number of iterations are reached.

4. PROPOSED HYBRID MCRNN

The proposed architecture for fall classification and detection is a HMCRNN. It consists of MCRNN layers where each channel is fully connected. The separate channels are merged at a MLP with two hidden layers and an output layer. Each channel in HMCRNN consists of two hidden layers with 128 and 64 random neurons respectively, resulting in 3×128 and 3×64 neurons in the two hidden RNN layers. The output activation vector $\boldsymbol{q}_{\text{RNN}}$ for an RNN layer implementation in Tensorflow/Keras can be derived from Eq. (2), using Eqs. (3), (4), (5) and (6). The activation function q_n for a neuron n can be given as:

$$q_n = \frac{\sum_{m=1}^{N} q_m w^+(m,n)}{r(n) + \sum_{m=1}^{N} q_m w^+(m,n)} \tag{20}$$

where $\Lambda(n)$ and $\lambda(n)$ are 0 for neurons with no external input. For an entire RNN layer with $n = 1, \ldots, N$ neurons, weight matrices $\mathbf{W}^+$ and $\mathbf{W}^-$ can be used to represent weights as:

$$\mathbf{W}^+ = [w^+(m,n)],\ m,n = 1,\ldots,N \tag{21}$$

$$\mathbf{W}^- = [w^-(m,n)],\ m,n = 1,\ldots,N \tag{22}$$

while the activation function outputs of neuron from the previous layer can be represented as a vector $\mathbf{x}$, where $\mathbf{x}$ is:

$$\mathbf{x} = [q_m],\ m = 1,\ldots,N \tag{23}$$

The final RNN layer can be represented in matrix form as:

$$\boldsymbol{q}_{\text{RNN}} = \frac{\mathbf{W}^+ \cdot \mathbf{x}}{\boldsymbol{r}_{\text{RNN}} + \mathbf{W}^- \cdot \mathbf{x}} \tag{24}$$

where $\boldsymbol{q}_{\text{RNN}}$ is a vector of activation values of the RNN layer. $\mathbf{W}^+, \mathbf{W}^- \in \mathbb{R}^+$ corresponding to matrix forms of Eqs. (5) and (6) represent weight parameter matrices, while $\boldsymbol{r}_{\text{RNN}}$ is a

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

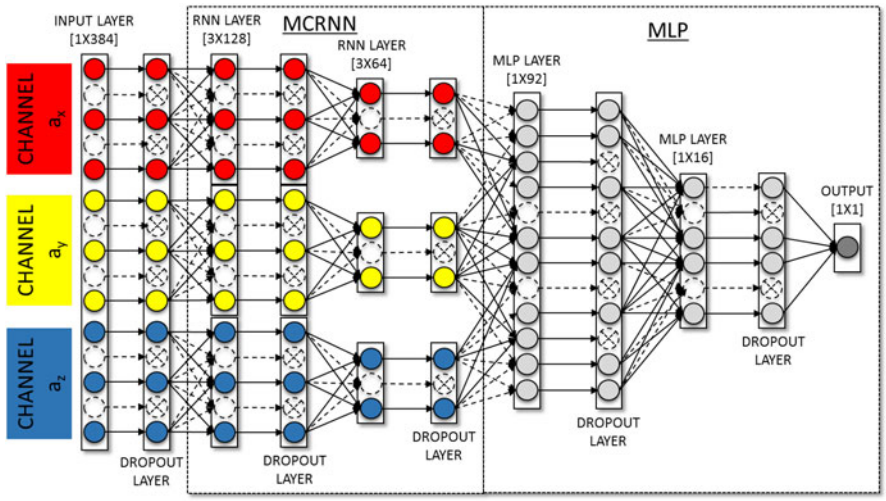


FIGURE **1.** Hybrid deep MCRNN with dropout layers in Tensorflow/Keras.

vector of constants. The three channels are fully connected to 92 neurons at the first MLP layer followed by 16 neurons in the second MLP layer. The hidden MLP layers uses the ReLU activation function. The output activation vector $\boldsymbol{q}_{\text{MLP}}$ for a ReLU MLP layer with a preceding RNN layer is given as:

$$\boldsymbol{q}_{\text{MLP}} = f_{ReLU}(\mathbf{W}_{\text{MLP}} \cdot \boldsymbol{q}_{\text{RNN}} + \boldsymbol{b}) \tag{25}$$

where $\mathbf{W}_{\text{MLP}}$ and $\boldsymbol{b}$ are weight matrix and bias vector of the MLP layer, respectively. $\boldsymbol{q}_{\text{RNN}}$ is the output activation vector for the preceding RNN layer. The final output layer is a single input neuron with a sigmoid activation function for binary classification.

The proposed HMCRNN is implemented with and without dropout regularization [33] in Tensorflow and Keras. Figure 1 shows a typical dropout regularization implementation in Keras, where each layer including the input and hidden layers are followed by a dropout layer. The dropout layer works by randomly dropping or turning off a fraction of units, see Section 5 on methodology for fractions used. The three raw accelerometer signals a_x, a_y and a_z from each of the three axes of motion x, y and z respectively, are normalized between 0 and 1. The normalized signals are used as inputs to the HMCRNN for classification of fall activity. Each RNN channel layer processes the three accelerometer signals separately. The output of these layers are then processed with the merged and fully connected MLP layers.

The proposed hybrid HMCRNN architecture was trained and tested for fall detection with the RMSProp optimization of the gradient decent algorithm. The RMSProp optimization exploits the moving average of the square of gradients to normalize the gradient. It balances step size and avoids the vanishing or exploding of the gradient with the propagation of energy. It is equivalent to the first update vector of the Adadelta [44], and can be expressed as:

$$\mathbb{E}[\text{grad}^2]_t = \gamma \mathbb{E}[\text{grad}^2]_{t-1} + (1-\gamma)\text{grad}_t^2 \tag{26}$$

$$\phi_{t+1} = \phi_t - \frac{\eta}{\sqrt{\mathbb{E}[\text{grad}^2]_t + \varepsilon}} \tag{27}$$

where $\mathbb{E}[\text{grad}^2]_t$ and grad_t^2 are the running average and gradient at time t respectively, while $\mathbb{E}[\text{grad}^2]_{t-1}$ is the running average at time $t-1$. γ is the moving average constant.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

The $\phi_{t+1} \in \mathbb{R}^d$ represents the model's parameter at time $t+1$, η is the learning rate and the second term of Eq. (27) represents the parameter update vector. The training and testing methodology for all the networks is discussed in the next section.

5. METHODOLOGY

Fall Dataset: The data used for our work is a publicly available accelerometer dataset by Kwolek and Kepski [21] for falls and various Activities of Daily Life (ADL) including walking up/down stairs, sitting on a chair, lying down, picking up objects, standing up and sitting down. The fall and ADL data is acquired with an Inertial Management Unit (IMU). The IMU device used is a motion sensing platform with an overall sampling rate of 256 Hz. The data is transmitted in real-time, wirelessly through Bluetooth for processing. The IMU device consists of a 16-bit tri-axis gyroscope and a 12-bit tri-axis accelerometer. The accelerometer sensor of the device was used for analysis and classification of falls in our work. The accelerometer sensor measures acceleration in units of G-force (g) for three axes of motion a_x, a_y and a_z with obtained values ranging from -84 to 8g. The sensor was worn near the pelvis by five volunteers who performed three kinds of falls, namely forward, lateral and backward apart from ADLs.

The received accelerometer values are normalized between 0 and 1 to satisfy the RNN constraints, as outlined by Gelenbe in [11]. The sensor values are divided into 128 sample windows for classification of falls and ADLs. The duration of activities varies and 128 samples capture basic movements, such as falling, sitting down and lying down within the dataset used for fall classification. The activities that span longer or shorter durations are divided into integer multiples of 128 sample windows with zero padding. Tri-axes accelerometer values for 128 sample segments are illustrated in Figure 2 for fall activity. Samples from each of the activities were used for training and validation of the proposed architecture.

Experimental Specifications: The training and testing of fall activity dataset for classification was performed on a hardware platform with Intel core i5 quad core CPU and 8 GB of RAM with Ubuntu 16.04 Xenial operating system. The proposed system was designed, implemented and tested in Tensorflow and Keras. Four deep multilayer neural networks, namely MLP, CNN, MCRNN and the proposed "Hybrid" architecture HMCRNN were implemented. The MLP, CNN and HMCRNN were implemented with sigmoid output activation functions for binary classification between falls and no fall activities. Our CNN output is also based on the sigmoid activation function, since it is well known that the multi-class softmax activation function reduces to a sigmoid for two class problems [24]. The structures were chosen to give best results on the fall activity dataset [21]. The structures of deep neural networks with input, hidden and output layers are summarized in Table 1 and are discussed below.

The deep MLP has fully connected dense layers with an input layer of size 384, four hidden layers of size 384-192-92-16 and a single node output layer with sigmoid activation functions, resulting in a total of 6 layer MLP. The CNN structure in Gelenbe and Yin [15] is used for comparison with the HMCRNN. It consists of two convolution layers, one pooling and a fully connected layer, apart form input and output layers. The three channel input (3×128) from each of the accelerometer's axes a_x, a_y, a_z is reshaped into $(3 \times 8 \times 16)$ before input to the convolutional layer. While, Gelenbe and Yin [15] reshape due to their large input vector size of 3000 samples. We observe that reshaping provides better performance for our three axis accelerometer fall dataset. The convolutional layers use 30 kernels with $[2 \times 2]$ size. While, pooling layer is a max-pool layer and is flattened before connecting to the Fully Connected (FC) layer. The MCRNN structure consists of three-channel layers

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

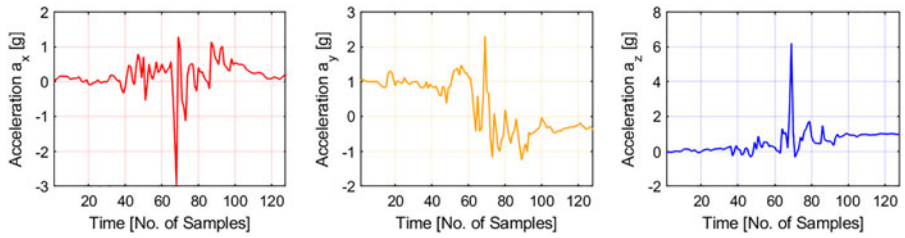


FIGURE **2.** Tri-axes accelerometer signals a_x, a_y and a_z for fall activity.

TABLE 1. Deep neural network structures

Deep network	Input layer	Hidden layers	Output layer
Deep MLP	384	384-192-92-16	1 (Sigmoid)
CNN	$3 \times 8 \times 16$	Conv-Conv-MaxPool-FC	1 (Sigmoid)
MCRNN	3×128	3×128-3×64-1×92-1×16	1 (Sigmoid)
HMCRNN	3×128	3×128-3×64-1×92-1×16	1 (Sigmoid)

TABLE 2. Learning algorithm parameters

Learning parameters	
Learning algorithm	RMSProp
Learning rate, η	0.001
Moving average parameter, γ	0.9
Epochs, ϵ	40

followed by merged layers. It consists of 3×128 input layer followed by two multichannel hidden layers 3×128–3×64 and 2 FC dense MLP layers 1×92–1×16, followed by a single node output layer. The proposed HMCRNN for fall detection and classification consists of a single 3×128 input layer followed by two multichannel RNN hidden layers 3×128–3×64 and 2 FC MLP dense layers 1×92–1×16, followed by a single node output layer, resulting in a total of 6 layers. All the networks were trained and tested on the fall dataset with the stratified k-fold procedure.

Stratified K-Fold Procedure: The fall dataset was split into 4 folds, with 3 folds for training and 1 fold for testing, resulting in a split of 75%–25% between training and test data. All the networks were trained with the RMSProp optimizer [19] for all four permutations of stratifed k-fold. The learning parameters for RMSProp algorithm are given in Table 2. The final results were obtained from mean and standard deviation of the stratified k-fold permutations.

Dropout Regularization: The training and testing procedures were repeated with regularization to reduce over fitting of the training model and improve testing accuracy. The dropout technique introduced by Srivastava et al. [33] was used for regularization. Dropout procedure was applied to both input layer and the hidden layers. The dropout value of 15% was initially used and increased up to 30% to evaluate higher performance for testing. The best testing performance was used as the final value with the dropout method.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

TABLE 3. Impact of HMCRNN layers on accuracy

Network		I	II	III	IV	V	VI
Total layers		5	5	6	7	7	8
Hidden layers	RNN layers	1	2	2	2	3	3
	MLP layers	2	1	2	3	2	3
Total neurons		877	977	1069	1133	1165	1230
Accuracy		85.7	85.78	91.5	82.84	82.92	75.82

6. EXPERIMENTAL RESULTS AND DISCUSSION

The test results give the highest accuracy of 91.5% for 6-layer HMCRNN architecture, as illustrated by Network III in Table 3. However, if the number of layers is increased to 7 by either adding RNN or MLP layer, the accuracy decreases to approximately 82%. Accuracy of the network further decreases with increasing the number of layers to 8. Similarly, a decrease in number of layers to 5 results in lower accuracy of 85%. The proposed network is optimal with the highest accuracy of 91.5%, while an increase or decrease in layers results in lower accuracy.

The test results of the proposed 6-layer HMCRNN, MLP, MCRNN and CNN are presented as percentages of confusion matrix values in Table 4 for comparison. The percentage values for True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) classifications are averaged over all 4-folds. It is clear from Table 4 that HMCRNN gives the highest true positive values for falls at 38.56% as compared to other classifiers, as well as the lowest false classifications overall of 8.5%.

The FP values for all the classifiers are further analyzed in Table 5 and provide insight into the activities misclassified or confused by classifiers as falls. Table 5 shows that the classifiers MCRNN, CNN and HMCRNN misclassify "lying down" as falls. CNN and HMCRNN produce the highest false positive values for falls due to the "lying down" activity. Furthermore, amongst all the activities, only lying down activity is misclassified as falls by CNN. Also, MCRNN misclassifies both lying and sitting activities as falls, while the false positives for MLP are zero in Table 5. This is due to the poor classification response of MLP with a lower positive classification value of 2.1% (TP) and 0% (FP), as illustrated in Table 4.

The final training and testing accuracies of fall detection with and without the dropout regularization are illustrated in Table 6. The MLP architecture with 6 layers (Table 1) provides 57.2% testing accuracy. No improvements in testing accuracy of MLP is observed with dropout even though dropout results in a slight decrease in training accuracy. MCRNN performs better than MLP at 69.85% with dropout. However, the performance accuracy of MCRNN is considerably low as compared to the proposed HMCRNN and CNN implementations. The CNN architecture provides an improvement of 20% over a traditional MCRNN with an accuracy of 91.22%. While, the proposed hybrid HMCRNN has the highest testing accuracy for fall detection at 92.23% with dropout, which is comparable to the CNN performance of 91.22%.

The CNN architecture is computationally intensive and has a high computational time cost of $\sim 2\times$ compared to the proposed HMCRNN architecture. The HMCRNN computational time cost is 51% less without dropout regularization and 43% less with dropout than the CNN architecture. The total computational time cost of each deep learning architecture is illustrated in Figure 3, normalized to an MLP implementation without dropout regularization. The total time costs of the three architectures MLP, MCRNN and HMCRNN

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

TABLE 4. Positive and negative classifications

Classifier	TP (%)	TN (%)	FP (%)	FN (%)
MLP	2.10	55.09	0	42.81
MCRNN	18.63	48.61	8.58	2.42
CNN	37.17	52.86	2.94	7.03
HMCRNN	38.56	52.94	4.25	4.25

TABLE 5. False positives: activities misclassified as falls

Activities	MLP (%)	MCRNN (%)	CNN (%)	HMCRNN (%)
Sitting down	0	1.47	0	0
Sitting on chair	0	2.94	0	0
Sitting down/Standing up	0	1.39	0	0
Lying down	0	1.39	2.94	2.86
Lying on bed	0	1.39	0	0
Picking up objects	0	0	0	1.39
Walking	0	0	0	0
Total FP	0	8.58	2.94	4.25

TABLE 6. Accuracies of deep neural networks on fall dataset

Deep network	Training accuracy (%)	Testing accuracy (%)
MLP	58.59 (±2.85)	57.19 (±1.63)
MLP-Dropout	57.62 (±0.67)	57.19 (±1.63)
CNN	96.66 (±0.84)	90.03 (±4.59)
CNN-Dropout	96.19 (±0.04)	91.22 (±6.25)
MCRNN	70.04 (±5.39)	67.24 (±2.29)
MCRNN-Dropout	69.23 (±2.36)	69.85 (±1.35)
HMCRNN	95.89 (±2.31)	91.5 (±2.61)
HMCRNN-Dropout	95.12 (±2.25)	92.23 (±1.82)

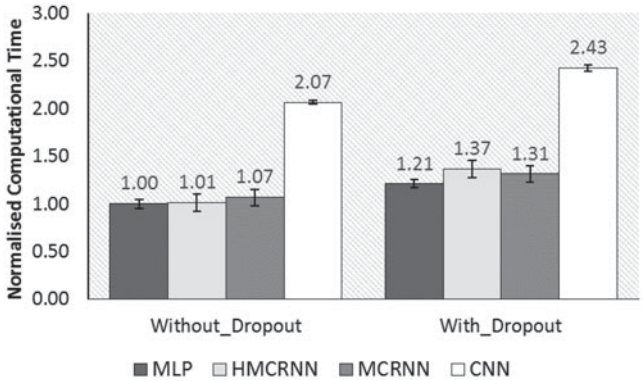


FIGURE 3. Normalized time cost w.r.t."MLP without dropout".

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

for falls and activities dataset, are comparable with a maximum of 7% difference without dropout and 13% with dropout regularization.

The proposed work demonstrates comparable accuracy to CNN architecture and lower computational time with a hybrid network for fall detection and classification. While, activity recognition utilizing RNN models have achieved high-accuracy classification with dense RNN clusters and enhanced learning algorithms [15,40], the proposed hybrid approach is orthogonal to the dense RNN structure and the efficient learning algorithms in [3,15,40]. The proposed work can therefore benefit from the opportunities of enhanced learning algorithms and dense RNN clusters to further improve accuracy.

7. CONCLUSION AND FUTURE WORK

Falls are a health hazard in older adults and lead to high mortality rates. Deep learning techniques have been explored for fall detection to achieve high accuracy. However, frequently used deep learning models like CNN achieve high accuracy at the cost of high computational time. In contrast to CNN-based techniques, a novel computationally efficient RNN-based hybrid architecture is presented. The proposed HMCRNN architecture for fall classification and detection exploits raw accelerometer signals to provide high accuracy of classification. HMCRNN performs better than the MCRNN with an improvement of 20%. HMCRNN implementation offers a high accuracy of 92.23% for fall detection, comparable to the CNN performance of 91.22% at a much lower computational cost. The CNN architecture takes up to $\sim 2\times$ the computational time of the HMCRNN for the fall dataset. The proposed scheme offers 51% less total time cost without dropout regularization and 43% less with dropout regularization for fall classification. In comparison to an MLP implementation, our scheme incurs 10% higher computational time cost on average with a 34.12% improvement in accuracy. Future work intends to use multiple datasets including smart phone accelerometer data for classification of falls. Additionally, this will also extend the present work to include a larger number of activities such as running, slow and fast walking etc.

References

1. Basterrech, S., Mohammed, S., Rubino, G., & Soliman, M. (2009). Levenberg–marquardt training algorithms for random neural networks. *The Computer Journal* 54: 125–135.
2. Boyle, T. & Ravenscroft, A. (2012). Context and deep learning design. *Computers and Education* 59: 1224–1233.
3. Brun, O., Yin, Y., & Gelenbe, E. (2018). Deep learning with dense random neural network for detecting attacks against IoT-connected home environments. *Procedia Computer Science* 134: 458–463.
4. Chen, Y. & Xue, Y. (2015). A deep learning approach to human activity recognition based on single accelerometer. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, IEEE, pp. 1488–1492.
5. Cleland, I., Kikhia, B., Nugent, C., Boytsov, A., Hallberg, J., Synnes, K., McClean, S., & Finlay, D. (2013). Optimal placement of accelerometers for the detection of everyday activities. *Sensors* 13: 9183–9200.
6. Consultants PCP market research (2012). Falls : measuring the impact on older people. http://www.wrvs.org.uk/Uploads/Documents/Reports and Reviews/Falls report_web_v2.pdf.
7. Coskun, D., Incel, O.D., & Ozgovde, A. (2015). Phone position/placement detection using accelerometer: Impact on activity recognition. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, IEEE, pp. 1–6.
8. Cramer, C.E. & Gelenbe, E. (2000). Video quality and traffic qos in learning-based subsampled and receiver-interpolated video sequences. *IEEE Journal on Selected Areas in Communications* 18: 150–167.
9. Delaye, E., Sirasao, A., Dudha, C., & Das, S. (2017). Deep learning challenges and solutions with xilinx fpgas. In *Proceedings of the 36th International Conference on Computer-Aided Design*, IEEE, pp. 908–913.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

10. Gelenbe, E. (1989). Random neural networks with negative and positive signals and product form solution. *Neural Computation* 1: 502–510.
11. Gelenbe, E. (1993). Learning in the recurrent random neural network. *Neural Computation* 5: 154–164.
12. Gelenbe, E. & Hussain, K.F. (2002). Learning in the multiple class random neural network. *IEEE Transactions on Neural Networks* 13: 1257–1267.
13. Gelenbe, E. & Stafylopatis, A. (1991). Global behavior of homogeneous random neural systems. *Applied Mathematical Modelling* 15: 534–541.
14. Gelenbe, E. & Yin, Y. (2016). Deep learning with random neural networks. In *Proceedings of SAI Intelligent Systems Conference*, Springer, pp. 450–462.
15. Gelenbe, E. & Yin, Y. (2017). Deep learning with dense random neural networks. In *International Conference on Man–Machine Interactions*, Springer, pp. 3–18.
16. Gibson, R.M., Amira, A., Ramzan, N., Casaseca-de-la-Higuera, P., & Pervez, Z. (2016). Multiple comparator classifier framework for accelerometer-based fall detection and diagnostic. *Applied Soft Computing* 39: 94–103.
17. Gibson, R.M., Amira, A., Ramzan, N., Casaseca-de-la Higuera, P., & Pervez, Z. (2017). Matching pursuit-based compressive sensing in a wearable biomedical accelerometer fall diagnosis device. *Biomedical Signal Processing and Control* 33: 96–108.
18. Grenet, I., Yin, Y., Comet, J.-P., & Gelenbe, E. (2018). Machine learning to predict toxicity of compounds. In *International Conference on Artificial Neural Networks*, Springer, Cham, pp. 335–345.
19. Hinton, G. (2014). Lecture notes in neural networks for machine learning.
20. Khan, S.S. & Taati, B. (2017). Detecting unseen falls from wearable devices using channel-wise ensemble of autoencoders. *Expert Systems with Applications* 87: 280–290.
21. Kwolek, B. & Kepski, M. (2014). Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer Methods and Programs in Biomedicine* 117: 489–501.
22. Lane, N.D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A.T. (2010). A survey of mobile phone sensing. *IEEE Communications Magazine* 48: 140–150.
23. Lu, N., Ren, X., Song, J., & Wu, Y. (2017). Visual guided deep learning scheme for fall detection. In *13th IEEE Conference on Automation Science and Engineering (CASE), 2017*, IEEE, pp. 801–806.
24. Martins, A. & Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pp. 1614–1623.
25. Masud, T. & Morris, R.O. (2001). Epidemiology of Falls. *Age and Ageing* 30(4): 3–7.
26. Mathie, M.J., Coster, A.C., Lovell, N.H., & Celler, B.G. (2004). Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement. *Physiological Measurement* 25: R1.
27. Nait Aicha, A., Englebienne, G., van Schooten, K.S., Pijnappels, M., & Kröse, B. (2018). Deep Learning to Predict Falls in Older Adults Based on Daily-Life Trunk Accelerometry. *Sensors* 18: 1–14.
28. Noury, N., Rumeau, P., Bourke, A., ÓLaighin, G., & Lundy, J. (2008). A proposal for the classification and evaluation of fall detectors. *Innovation and Research in Biomedical engineering (IRBM)* 29: 340–349.
29. Ordóñez, F.J. & Roggen, D. (2016). Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors* 16: 115.
30. Quero, J., Burns, M., Razzaq, M., Nugent, C., & Espinilla, M. (2018). Detection of falls from non-invasive thermal vision sensors using convolutional neural networks. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 2 p. 1236.
31. Riedmiller, M. & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Proceedings of the IEEE international conference on neural networks*, IEEE, pp. 586–591.
32. Robbins, H. & Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407.
33. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15: 1929–1958.
34. Sun, F., Wang, C., Gong, L., Xu, C., Zhang, Y., Lu, Y., Li, X., & Zhou, X. (2017). A power-efficient accelerator for convolutional neural networks. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, IEEE, pp. 631–632.
35. Tian, Y., Thompson, J., Buck, D., & Sonola, L. (2013). *Exploring the system-wide costs of falls in older people in Torbay*. King's Fund.
36. Timotheou, S. (2010). The random neural network: A survey. *The computer journal* 53: 251–267.
37. Wang, J., Chen, Y., Hao, S., Peng, X., & Hu, L. (2019). Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters* 119: 3–11.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press

38. Xu, T., Zhou, Y., & Zhu, J. (2018). New advances and challenges of fall detection systems: A survey. *Applied Sciences* 8: 418.
39. Xue-Wen, C. & Xiaotong, L. (2014). Big data deep learning: Challenges and perspectives. *IEEE Access* 2: 514–525.
40. Yin, Y. & Gelenbe, E. (2016a). Deep learning in multi-layer architectures of dense nuclei. *arXiv preprint arXiv:1609.07160*.
41. Yin, Y. & Gelenbe, E. (2016b). Nonnegative autoencoder with simplified random neural network. *arXiv preprint arXiv:1609.08151*.
42. Yin, Y. & Gelenbe, E. (2017). Single-cell based random neural network for deep learning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp. 86–93.
43. Zebin, T., Scully, P.J., & Ozanyan, K.B. (2017). Human activity recognition with inertial sensors using a deep learning approach. *Proceedings of IEEE Sensors*, pp. 1–3.
44. Zeiler, M.D. (2012). Adadelta: An adaptive learning rate method). *arXiv preprint arXiv:1212.5701*.

https://doi.org/10.1017/S0269964819000317 Published online by Cambridge University Press