

PAPER • OPEN ACCESS

Fall detection algorithm based on accelerometer and gyroscope sensor data using Recurrent Neural Networks

Recent citations

- [A Comprehensive Study on the use of Artificial Neural Networks in Wearable Fall Detection Systems](#)
Eduardo Casilari-Pérez and Francisco García-Lagos

To cite this article: I Wayan Wiprayoga Wisesa and Genggam Mahardika 2019 *IOP Conf. Ser.: Earth Environ. Sci.* **258** 012035

View the [article online](#) for updates and enhancements.

Fall detection algorithm based on accelerometer and gyroscope sensor data using Recurrent Neural Networks

I Wayan Wiprayoga Wisesa¹, Genggam Mahardika²

¹ Department of Informatics Engineering, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

² Department of Informatics Engineering, Institut Teknologi Sumatera, Lampung Selatan, Indonesia

Abstract. In our daily life activity, sometimes there is a chance of getting fall unintentionally. Unintentional falls are dangerous to health and may cause a serious problem, especially for elderly people whose have a higher probability of getting fall. In this paper, we develop an algorithm to distinguish falls from other activity daily living (ADL) based on accelerometer and gyroscope sensor data embedded on a wearable device. Several fall detection algorithms exist, with the majority are using rule-based algorithm. We take advantage of recurrent neural networks (RNN) as a tool for analyzing sequence time series data from sensors. The experiment was conducted using publicly available dataset UMA FALL ADL from Universidad de Málaga. The dataset consists of several recorded sensor-tag data, consisting of accelerometer, gyroscope and magnetometer sensor, representing the daily activity of several subjects including falls. Based on our experiment, we found that our algorithm yields a good result distinguishing fall from ADL.

1. Introduction

In our daily life activity, there is sometimes we're getting fall. A fall is an unexpected event that comes naturally while doing some activities. Based on WHO data, there are 37.3 million falls occurs every year, and they need serious medical attention [1]. Form this number, an estimated 646 thousand fatal falls occur every year. Furthermore, falls is considered as the second leading cause of unintentional injury death after the injury caused from the road traffic accident. More than 40% of these fatalities occur in the region of the Western Pacific and South East Asia. Moreover, the death rates are the highest among adults over the age of 60. Among them, elderly people suffer the greatest number of fatal falls. As life goes by, our physical ability such as is balance, sense, and vision are weakened. Such conditions may cause elderly people to have a higher probability of getting falls unintentionally. There also some factors related to the health problem affecting unintentional falls on elderly people such as hypertension, stroke, headache, vertigo, and rheumatic. Some serious injury related to falls might sometimes lead to death for elderly people. Unintentional falls do not only occur on elderly people. Another group having a high risk of getting a fall is the children. All of them needs intensive monitoring in order to minimize the risk of getting serious injury by helping them immediately after falls occur. Consequently, family members need to keep an eye on them every time.

The advancement of technology such as video surveillance enabled us to keep an eye on our family member remotely. Having such technology is very useful, but still inconvenience for some busy person. We still need to pay attention to their movements all the time. Based on such constraint, system for automatically detecting falls are developed by research worldwide. Automatic Fall Detection Systems (AFDS) has become a relevant research topic during the last decade. M Mubashir et.al. [2] classified AFDS into three main categories, vision-based, wearable-sensors-based, and ambience-sensor-based.



Vision-based method mainly use video captured by camera to detect multiple actions simultaneously [3]. The use of Kinect camera or video surveillance camera plays the role for data acquisition. E E Stone and M Skubic [4] use depth image produced by Kinect camera and tracking individuals over time. The ensemble of decision trees was used to classify falls event based on five features extracted from depth image data. The use of computer vision techniques by using combination of motion history and human shape variations extracted from video surveillance was conducted by C Rougier et.al. [5] with promising results. Ambience based approach attempts to combine audiovisual data and event sensing through vibration data [2]. Both vision and ambience-based approach are less intrusive. However, they considered cost ineffectively. Wearable-device based method rely on the device embedded on the body of the subject. The use of tri-axial accelerometer and gyroscope is the most popular sensors to help the identification of falls [6][7][8][9] because of cost-efficient and easy setup [2].

The algorithm for AFDS by using wearable sensor approach can be further classified into two major techniques, threshold-based and machine-learning based techniques [9]. This paper mainly focuses on developing the algorithm for recognizing fall using wearable sensor approach and machine learning techniques. We use publicly dataset named UMA Fall Dataset developed by E Casilari et.al. [10] in order to build the algorithm for our AFDS.

This paper organized as follows, in section two we discuss about the dataset and the Long Short-Term Memory architecture used for this research. Result of the experiments will be discussed in section three. We will also present our analysis based on the experiment results in the same section. Finally, the conclusion and our future work will be discussed in section four.

2. Methodology

2.1. The UMA Fall Dataset

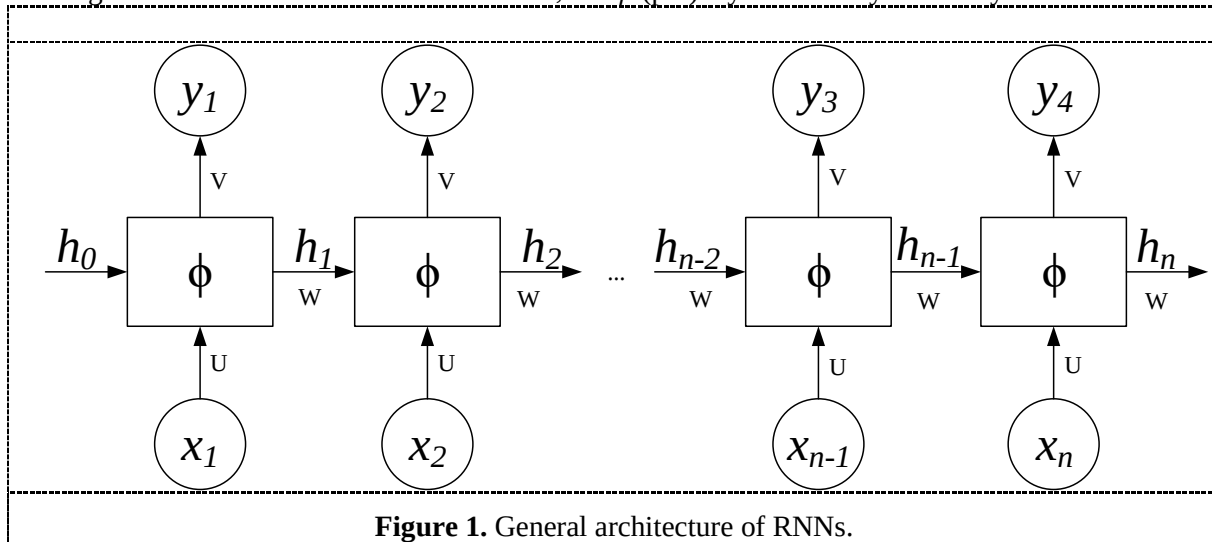
In order to develop the algorithm for AFDS, the UMA Fall Dataset was being used [10]. UMA Fall Dataset consists of 746 sample data from variety of test subject. The experiment was conducted by embedding 5 kind of wireless sensor node on the body of subject, one of them including one smartphone and four sensor tags. The sensors placement is in pocket (for smartphone) and attached to the ankle, wrist, chest, and waist (for all four sensor tags). All five sensor nodes transmit all tri-axial accelerometer, tri-axial gyroscope, and magnetometer data by using Bluetooth communication protocol. There are two kinds of movement scenario being experimented, ADL (Activity Daily Life) and Fall. For ADL scenario, there are 12 sub-scenarios, i.e. applauding, hands up, making a call, opening door, sitting and getting up on a chair, walking, bending, hoping, jogging, lying down on a bed, go downstairs, and go upstairs. Meanwhile, there are three sub-scenarios for falls, i.e. backward fall, forward fall, and lateral fall. The dataset is freely available to download from their official web-pages on the Internet. The dataset are in CSV file format. Each CSV file is one sample from one subject doing one scenario which can be identified by its filename. The content of each CSV file is the sensors data (tri-axial accelerometer, tri-axial gyroscope, and magnetometer) of recorded subject movement from all five sensors attached to the body parts of subject.

2.2. Recurrent Neural Networks

Deep learning and Recurrent Neural Networks (RNNs) getting most attention by the researchers for last two decades. RNN are designed to learn sequence or time-dependent data [11]. Such application of RNNs are sequence learning, sequence classification, and value memorization [12]. RNN is a member of neural networks family. They are unique with closed-loop feedback connections. They exploit time-dependent data by having a hidden state (or memory), describing data that has been seen so far (from previous timestep) [13]. At any point of timestep n , the value of hidden state h_n is a function of the value of hidden state and the input at current timestep described by equation (1).

$$h_t = \phi(h_{t-1}, x_t) \quad (1)$$

We can represent the RNN networks graphically just like a traditional neural network. Figure 1 shows general architecture of RNNs. Note that, the ϕ (phi) layer is usually a $\tanh$ layer.

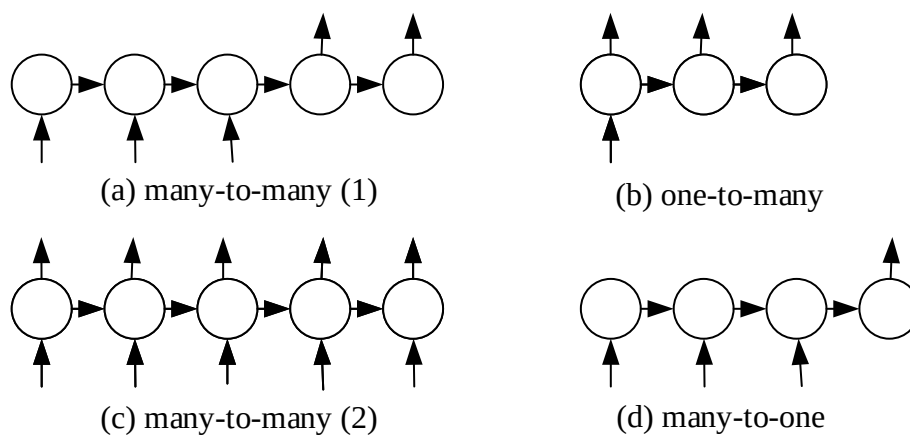


Based on Figure 1, input sequences with length of n are represented as $x_1, x_2, \dots, x_n$ and output sequence can be represented as $y_1, y_2, \dots, y_n$ for each timestep. Feedback output for each timestep n is represented by h_n , that is the sum of the product of weight vector/matrix W and the feedback output at previous timestep h_{t-1} and the product of weight vector/matrix U and current input data x_n passed through a layer (e.g. $\tanh$ layer). The output at current timestep y_n is the product of weight vector/matrix V and the current state of feedback output h_n with SoftMax function. The equations for RNN computation at each timestep n are formally defined by equation (2) and equation (3).

$$h_n = \tanh(h_{n-1}W + x_nU) \quad (2)$$

$$y_n = \text{softmax}(h_tV) \quad (3)$$

Unlike neural networks such as MLP (*Multi-Layer Perceptron*) or CNN (*Convolutional Neural Networks*) which accepts fixed-size input and produce fixed-size output. However, RNNs do not have this limitation. You can have sequences either for the input and output, or both. The output can be either a single output (many-to-one) i.e. y for sequence classification problem, or sequence output (many-to-many) for sequence learning problem. Some variations of RNNs topologies are shown in Figure 2.



2.3. Long Short-Term Memory

Just like traditional neural networks, there are many RNN variants developed by researchers around. The Long Short-Term Memory or LSTM networks is another variant of RNN developed by Hochreiter and Schmidhuber [3]. They have the ability to learn long-term dependencies and also the most widely used type of RNN [13]. We can see the architecture of LSTM cell on Figure 3.

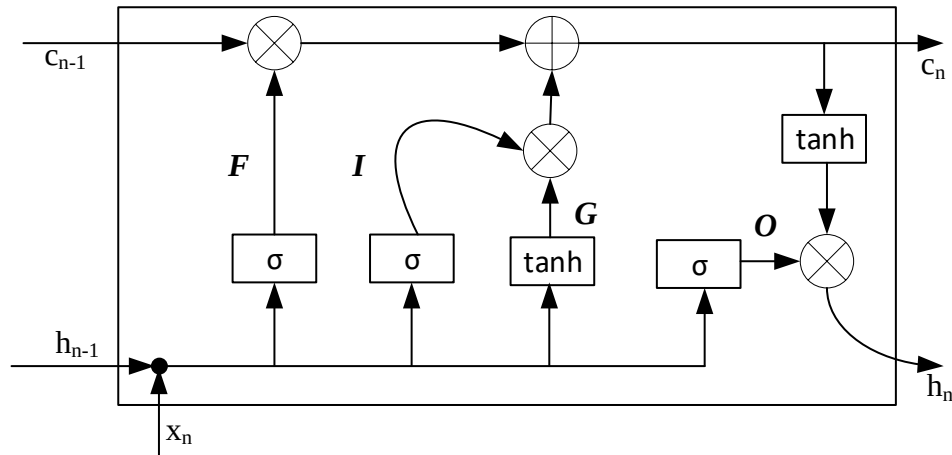


Figure 3. Basic LSTM cell operation architecture at timestep n [13].

As we seen on Figure 3, there are four layers or gates interacting in order to define cell state at certain timestep n denoted by F , I , O , and G gate each represent forget, input, output gate, and internal hidden state gate. These parameters value will be learned during training phase, and such mechanisms enabled the LSTM avoiding the vanishing gradient problem.

In order to understand how LSTM works, we will explain step-by-step computation on single LSTM cell at certain timestep n . Firstly, the F layer (or forget gate) defines how much the previous hidden state h_{n-1} information which we want to retain based on current input data x_n . A value of 1 as the result of the forget gate indicated that we are completely keep all previous information to be used as our information for computing current LSTM state. Conversely, the value of 0 indicate we will completely ignore previous information for computing current LSTM state. Then, the next step is to decide what new information we will store in current LSTM cell. This step involving two parts of computation. The I layer (or input gate) and defines which value of current information x_n you want to pass through based on previous hidden state h_{n-1} and G layer gives additional information of how much information you want to consider. Now, it's time to update the old cell state c_{n-1} as the new state c_n by adding the value of input gate multiplied by scaling factor G with the information which we decided to forget or remember previously, as defined by equation (4):

$$c_n = F * c_{n-1} + G * I \quad (4)$$

Finally, we compute output for the LSTM cell. The output will be a filtered version of our current cell state c_n defined by equation (5):

$$h_n = O * \tanh(c_n) \quad (5)$$

The O gate (or output gate) defines how much of the current hidden state you want to expose to the next layer at time step $n + 1$.

2.4. Experiment

We decided to use the data obtained from the sensor placed on the waist of the subject from UMA Fall Dataset based on majority of works in developing AFDS using wearable sensor approach such as in [6][7] [8][14]. Among 746 samples of data, only 617 samples were valid for tri-axial accelerometer and

gyroscope sensor attached on the waist. We also decided to omit magnetometer data in order to limit our scope. For experiment purpose, the dataset is being divided into two sets, for training phase and validation of our LSTM networks. Each sets of data were chosen randomly from 617 samples, yielding 494 samples for training the model (80 percent) and 123 samples for validating model. Because the original dataset consists of 15 different activities including falls, we decided to restrict our sequence classification problem to discriminate falls and ADL. Thus, the dataset's original label was mapped into one of the two labels, i.e. falls and ADL. Table 1 provides the mapping scheme of UMA Fall dataset for this research.

Table 1. Label mapping of UMA Fall Dataset for the experiment. Original label was mapped exactly into one label, either ADL or Fall.

Mapped Label	Original Label
ADL	<i>Applauding; Hands up; Making a call; Opening door; Sitting and getting up on a chair; Walking; Bending; Hoping; Jogging; Lying down on a bed; Go downstairs; Go upstairs.</i>
Fall	<i>Backward fall; Forward fall; Lateral fall.</i>

The LSTM networks are being implemented by using Keras framework, a high-level framework for deep learning for python programming language. For the experiment, we designed a simple RNN networks with one layer of LSTM cell with 100 hidden neurons. Our AFDS problem is treated as sequence classification problem, so we forced a fixed length sequence input for our LSTM. The input for LSTM cell is set to 306 features based with respect to maximum number of sequences in the dataset. Zero padding at the beginning of sequence were employed for sequence data shorter than 306. For output, we employ sigmoid activation layer for each sequence input. Figure 4 shows the diagram for our networks structure.

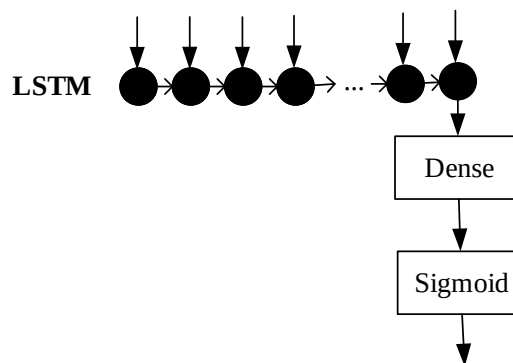


Figure 4. LSTM networks structure for experiment

3. Results and Analysis

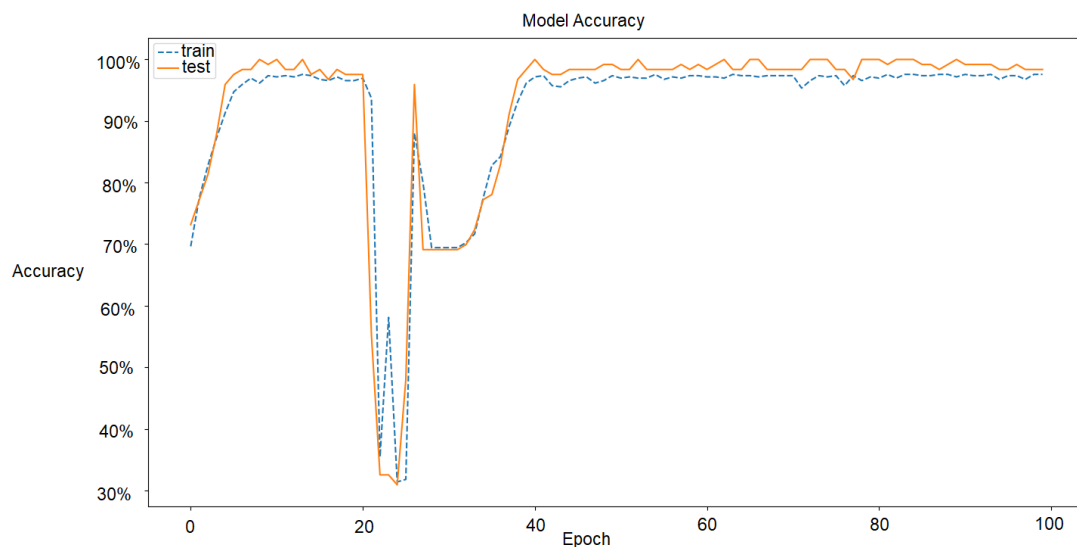
In this section, we present our experimental results of developing algorithm for AFDS using RNN. We separate our experiment into two scenarios. The first scenario is to classify each of tri-axial sensor data (gyroscope and accelerometer) using our LSTM model. This scenario is intended to bring insight about the characteristic of each sensor data in order to determine fall. The second scenario, we tried to combine all sensor data to be classified by the LSTM model in order to get understanding about the effect of combining all sensor data against the classification performance.

The first scenario we classify each sensor data separately. Best accuracy achieved using data from X-axis accelerometer data. Best validation accuracy achieved for this data is 100%. The results for all single sensor data are presented in Table 2.

Table 2. Average training and validation accuracy for the first scenario.

Sensor Type	Average accuracy		Sensor Type	Average accuracy	
	Train	Validation		Train	Validation
X-axis accelerometer	91,43%	92,31%	X-axis gyroscope	73,89%	74,47%
Y-axis accelerometer	76,89%	77,01%	Y-axis gyroscope	82,71%	72,24%
Z-axis accelerometer	86,62%	88,26%	Z-axis gyroscope	77,31%	77,25%

Figure 5 shows the history of training and validation for each epoch for X-axis accelerometer as the best individual classification performance.

**Figure 5.** Training accuracy and validation accuracy history for X-axis accelerometer. Dashed line represents training accuracy and solid line represent validation accuracy.

In order to get the detail about the performance of our LSTM networks, after training and validation process, we then saved the model we've created before and do the prediction (testing) phase for each model. The confusion matrix for each prediction phase are presented in Table 3.

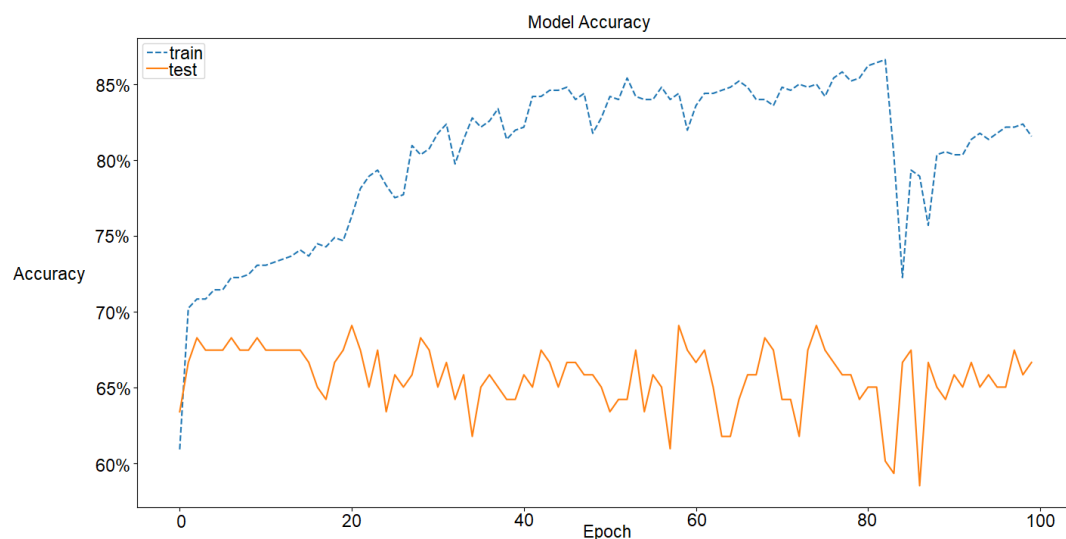
Table 3. Detail of classification performance for each model of individual sensor.

Sensor	ADL > ADL	ADL > FALL	FALL > ADL	FALL > FALL
ACC_X	83	2	0	38
ACC_Y	85	0	29	9
ACC_Z	84	1	22	16
GYR_X	63	22	1	37
GYR_Y	76	9	17	21
GYR_Z	77	8	1	37

Note: **ADL > FALL** at the top rows of the table means the correct label was **ADL** but classified as **FALL**.
ACC_X at the leftmost column correspond to X-axis accelerometer, **GYR_Y** correspond to Y-axis gyroscope, etc.

The best performance achieved by the model developed using X-axis accelerometer data. All fall events are correctly predicted. While, there are two ADL data incorrectly classified as fall event. This means, we got a good classification performance on UMA Fall Dataset by using only from X-axis accelerometer data.

For the second scenario, we tried to classify all sensor data by combining them as one instance. The effect of combining all sensor data against the classification performance are observed. We present the history of training and validation of this scenario in Figure 6. The best training accuracy achieved was 86,63%, while the best validation accuracy was 69,10%. This number is rather low compared to the performance of the first scenario by using data from X-axis accelerometer only.

**Figure 6.** Training accuracy and validation accuracy history for the second scenario

Based on Figure 6, we got a rather poor classification performance when data from all sensors are combined. In order to get the details about the classification performance for the second scenario, we present the classification performance on Table 4.

Table 4. Detail of classification performance for model of combined sensor data.

Sensor	ADL > ADL	ADL > FALL	FALL > ADL	FALL > FALL
COMB	75	10	31	7

Note: **FALL > ADL** at the top rows of the table means the correct label was **FALL** but classified as **ADL**.
COMB at the leftmost column correspond to the combination of all tri-axial accelerometer and tri-axial gyroscope.

By looking at the result on Table 4, the classification performance tends to detect majority of activities on the dataset as ADL. There are 31 falls data that incorrectly classified as ADL.

4. Conclusions and Future Works

In this experiment, we used LSTM (as part of Recurrent Neural Networks variations) in order to develop algorithm to differentiate Activity Daily Living (ADL) with falls as a part of Automatic Fall Detection System (AFDS). Our experiment shows that, by using X-axis accelerometer data, we got a good classification performance on UMA Fall Dataset. There are also some improvements needed in order to achieve a better classification performance such as applying average filter at pre-processing stage could reduce noise from the signal. Another scenario to be explored in the future is by using total accelerometer (or gyroscope) value as used in [6].

5. Acknowledgement

This research is supported by internal grant from Lembaga Penelitian, Pengabdian kepada Masyarakat dan Penjaminan Mutu Pendidikan Institut Teknologi Sumatera (Hibah Mandiri ITERA Smart 2018) no. 134am/IT9C1/PP/2018.

References

- [1] W. H. O. WHO, "Falls," 2018. [Online]. Available: view-source:<http://www.who.int/news-room/fact-sheets/detail/falls>. [Accessed: 16-Sep-2018].
- [2] M. Mubashir, L. Shao, and L. Seed, "Neurocomputing A survey on fall detection : Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, 2013.
- [3] S. Hochreiter and J. Schmidhuber, "Long Short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [4] E. E. Stone and M. Skubic, "Fall Detection in Homes of Older Adults Using the Microsoft Kinect," 2014.
- [5] C. Rougier, J. Meunier, A. St-arnaud, and J. Rousseau, "Fall Detection from Human Shape and Motion History Using Video Surveillance Fall Detection from Human Shape and Motion History using Video Surveillance," no. April 2014, 2007.
- [6] A. Z. Rakhman, L. E. Nugroho, Widyawan, and Kurnianingsih, "Fall Detection System Using Accelerometer and Gyroscope Based on Smartphone," in *ICITACEE*, 2014.
- [7] A. K. Bourke, J. V. O. Brien, and G. M. Lyons, "Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm," vol. 26, pp. 194–199, 2007.
- [8] N. Shibuya, B. T. Nukala, A. I. Rodriguez, and J. Tsa, "A Real-Time Fall Detection System Using a Wearable Wireless Gait Analysis Sensor and a Support Vector Machine (SVM) Classifier," pp. 66–67, 2015.
- [9] M. A. Habib, M. S. Mohktar, S. B. Kamaruzzaman, K. S. Lim, T. M. Pin, and F. Ibrahim, "Smartphone-Based Solutions for Fall Detection and Prevention: Challenges and Open Issues," pp. 7181–7208, 2014.
- [10] E. Casilari, J. A. Santoyo-ramón, and J. M. Cano-garcía, "ScienceDirect ScienceDirect UMAFall : A Multisensor Dataset for the Research on Automatic Fall Detection," *Procedia Comput. Sci.*, vol. 110, pp. 32–39, 2017.

- [11] L. R. Medsker and L. C. Jain, "RECURRENT NEURAL Design and Applications."
- [12] J. Brownlee, "5 Examples of Simple Sequence Prediction Problems for Learning LSTM Recurrent Neural Networks," *Machine Learning Mastery*, 2017. [Online]. Available: <https://machinelearningmastery.com/sequence-prediction-problems-learning-lstm-recurrent-neural-networks/>. [Accessed: 16-Sep-2018].
- [13] F. F. Q. Xjui, P. O. B. O. E. Mpx, and A. Gulli, *Deep Learning with Keras*. .
- [14] S. Abbate, M. Avvenuti, F. Bonatesta, G. Cola, and P. Corsini, "A smartphone-based fall detection system," 2012.