ISSN 2303-4521
Periodicals of Engineering and Natural Sciences *Original Research*
Vol. 9, No. 3, September 2021, pp.958-969

Real-time classification of various types of falls and activities of daily livings based on CNN LSTM network

Kadhum Al-Majdi[1], Raed S.H. AL-Musawi[2], Ali H. Ali[3], Yaqeen S. Mezaal[4]
[1]Medical Instrumentation Engineering Department, Ashur University College, Iraq
[2]Electrical Department, Faculty of Engineering, Babylon University, Iraq
[3]Electronic and Communications Department, Faculty of Engineering, University of Kufa, Iraq
[4]Medical Instrumentation Engineering Department, Al-Esraa University College, Iraq

ABSTRACT

In this research, two multiclass models have been developed and implemented, namely, a standard long-short-term memory (LSTM) model and a Convolutional neural network (CNN) combined with LSTM (CNN-LSTM) model. Both models operate on raw acceleration data stored in the Sisfall public dataset. These models have been trained using the TensorFlow framework to classify and recognize among ten different events: five separate falls and five activities of daily livings (ADLs). An accuracy of more than 96% has been reached in the first 200 epochs of the training process. Furthermore, a real-time prototype for recognizing falls and ADLs has been implemented and developed using the TensorFlow lite framework and Raspberry PI, which resulted in an acceptable performance.

Keywords: Deep Learning; CNN-LSTM; fall detection; ADLs; On-device Inference.

Corresponding Author:
Kadhum Al-Majdi
Medical Instrumentation Engineering Department, Ashur University College
Baghdad, Iraq
E-mail: dr.kadhum@au.edu.iq

1. Introduction

It is a known fact that the proportion of older adults living alone happens to be grown globally wide. Annually, nearly 28% to 35% of elderly individuals aged 65 years and above fall two to four times, increasing with age [1]. Falls are prominent among other causes of unintentional injury. Fall events influence the physical and psychological health of an older adult. Older adults have a weak and vulnerable body, and injuries resulting from falls include physical damage and bone fractures. It may lead to death or prolonged lie inability to recover [2]. Many researchers have studied falls and activities of daily livings (ADLs) using different methodologies, including threshold and machine learning. A threshold-based detection algorithm [3] is broadly used in wearable devices. A fixed threshold will determine other acceleration motions associated with the human body, according to some particular combination of actions, to determine whether a fall or non-fall event has occurred. The primary aims of fall and ADL recognition are to immediately detect the occurring of a fall in real-time and result in a rapid alert that can reduce outcomes with medical help response time. Researches about this interest are broadly categorized into wearable-based devices and vision-based devices. Vision-based devices generally have some constraints; vision-based have come up as an approach to hold the strengths of camera-based systems, as such technology is rich with relevant information about the surrounding place. Wearable-based devices tend to be a more practical approach that can ensure privacy and individual convenience; furthermore, portability and low cost are the most salient features of these devices. This research aims to build a solution to recognize and predict falls and ADLs by using the Sisfall public dataset [4]. Sisfall dataset is the wealthiest dataset with variants of falls and ADLs, carried out by more than 30 participants, one of them an elderly individual who performed all fall events. Using various methods, machine learning and deep learning can be performed on this dataset.

The application of deep learning models for falls and ADLs recognition using wearable-based sensors has been an area of recent focus, with new methods being introduced constantly. Schalk [5] conducted the LSTM model using the WISDM public dataset to recognize the activities of daily livings using an accelerometer sensor. Six activities were identified, and the results showed more than 94% accuracy. Wayan [6] used a UMA public dataset [7] composed of falls and ADLs using accelerometer and gyroscope sensors. Their experiments on the LSTM model showed that the best accuracy resulted from the accelerometer sensor. They set a binary classification model to differentiate ADL from falls. Each accelerometer axis is separated and then fed into the neural network to be trained; the final results showed that using x-axis accelerometer data only leads to high classification performance. Musci [8] implemented real-time online fall detection based on the LSTM model. Sisfall public dataset was used with three classes to be detected in real-time: fall, alert, and non-fall; the result of the experiment was compared to the obtained results from the Sisfall authors, and they showed high precision on fall detection. In [9], the authors combined different machine learning and deep learning models, including the LSTM, to enhance the classification of falls and ADLs. Twelve healthy subjects performed a collection of accelerometer and gyroscope sensor data. The multiclassification experiment showed that 99.81% average accuracy resulted from the LSTM model. Also, their research showed high accuracy in traditional machine learning models, while the Support vector machine (SVM) model showed 98.26% average accuracy. Hybrid models also have been used in related work; in [10], the authors combined Convolutional Neural Networks (CNN) with the LSTM model. Hence, the CNN feature extraction ability and LSTM ability for processing time-series sequences were utilized; their results showed that the CNN-LSTM model has higher detection accuracy than the SVM even with small volume datasets. This research proposes two variants of LSTM architectures for classifying different types of falls and ADLs using data gathered from the accelerometer sensor only. Also, we compiled the trained CNN-LSTM model using TensorFlow Lite for real-time hardware implementation on the Raspberry Pi platform.

2. Dataset

In this section, we present the details of the main flow of this research used for fall and ADLs recognition. This approach has been applied primarily to the Sisfall dataset [4]. Figure.1 shows the flowchart for the steps performed in this research to predict the falls and ADLs events from accelerometer sensor data. For convenience, we will discuss it in terms of three significant steps.

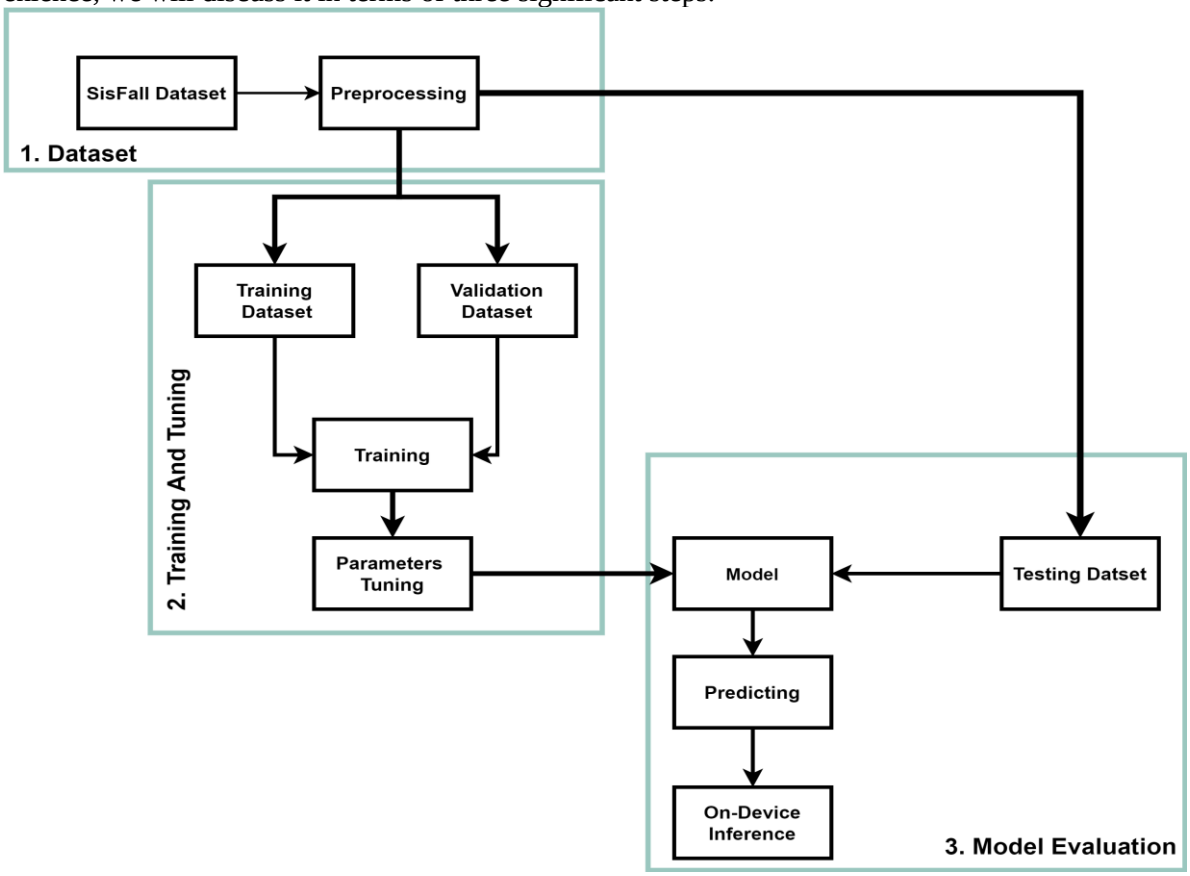


Figure 1. Flowchart for the proposed model in this research

To develop the algorithms for the proposed model, Sisfall public dataset was being used. Sisfall dataset contains 15 falls and 19 ADLs performed by 38 subjects with a sensor fixed on their waist. Among other public domain datasets, Sisfall is distinct since it has older people who performed falls and activities of daily living (ADLs). The Sisfall dataset was collected using three different sensors, two of them are accelerometer sensors, and the third is a gyroscope sensor. The dataset is given in bits and can be easily converted into gravitational acceleration (GA). The ADL activities in the Sisfall dataset include walking, sitting, jogging, standing..., etc. Whereas fall activities are 15 activities, i.e., falling forward, falling backward, falling while walking…etc. The dataset is in CSV file format. It is stored in folders and sorted by subjects and activities, where each file name represents an activity and the subject code and the trials that an issue is performed for an activity. We summarized the significant characteristics of the Sisfall dataset in table.1.

Table 1. The key characteristics of the Sisfall dataset [4]

Characteristics	Sisfall dataset
Sampling frequency	200 Hz
Number of subjects	38 subjects
Number of ADLs	19
Number of falls	15
Subjects age	19-75
Sensors used	Triaxial accelerometer and gyroscope
Position of sensor	Waist

We decided to choose the accelerometer data only since previous studies showed high accuracy when using accelerometer data only [5]. Figure.2 shows the 3-axis acceleration curve for some falls and ADLs recorded in the Sisfall dataset.

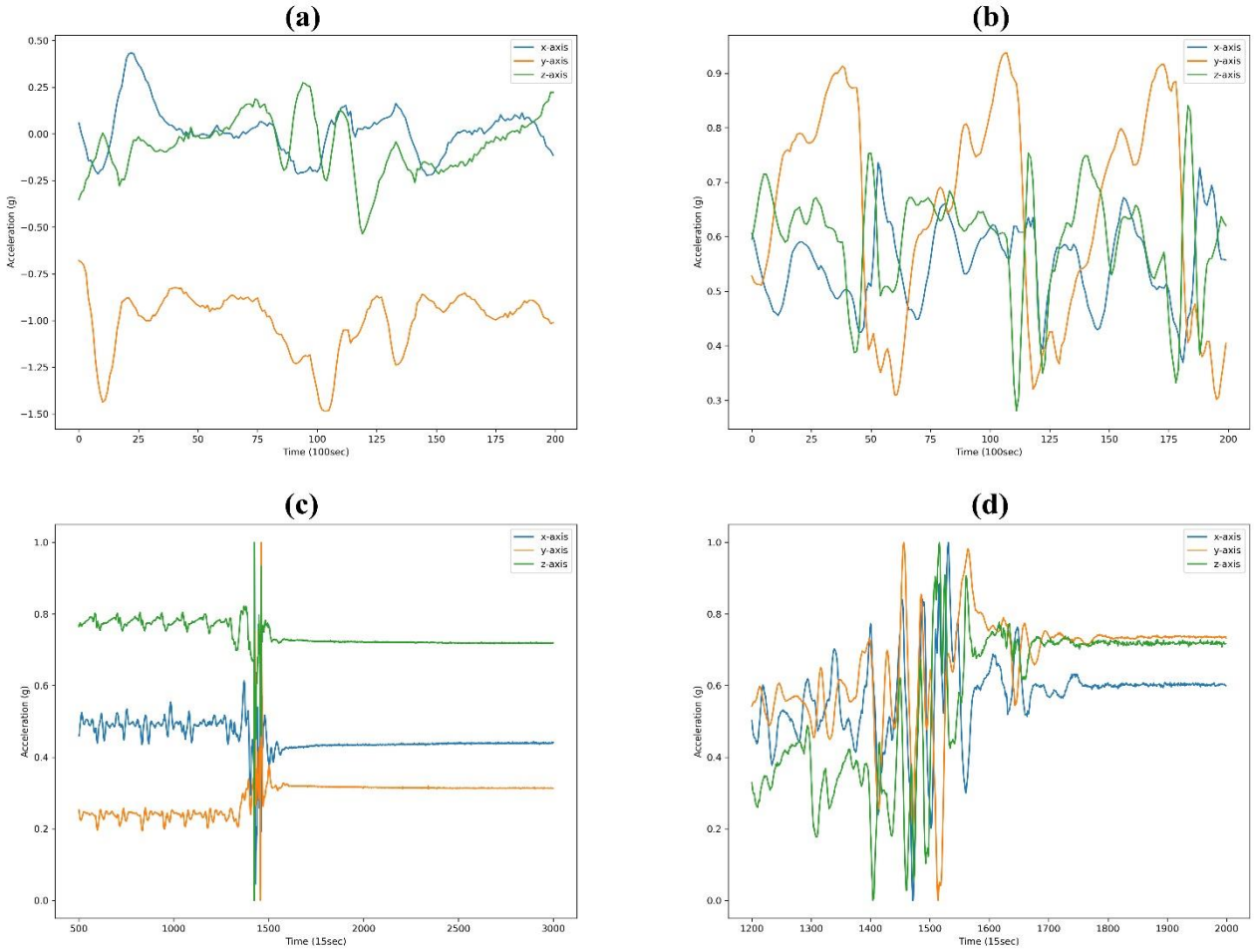


Figure 2. Acceleration curves of fall and ADL: (a) the acceleration curve of strolling, (b) the acceleration curve of jogging, (c) the acceleration curve of fall forward while walking caused by a slip, and (d) the acceleration curve of fall backward while walking caused by a slip

Since deep learning is computationally costly in the training stage or even at the prediction stage, we selected distinct classes from Sisfall to evaluate the LSTM and CNN-LSTM models. Table.2 shows the selected categories.

Table 2. Selected falls and ADLs from the Sisfall public dataset

Code	Activity	Trials	Duration
D01	Walking Slowly	1	100s
D02	Walking quickly	1	100s
D03	Jogging slowly	1	100s
D04	Jogging quickly	1	100s
D05	Walking upstairs and downstairs slowly	5	25s
F01	Fall forward by a slip	5	15s
F02	Fall backward by a trip	5	15s
F03	Lateral fall while walking caused by a slip	5	15s
F04	Walking and Fall forward by a trip	5	15s
F05	Jogging and fall forward by a trip	5	15s

3. Proposed model

This research uses pure long-short-term memory (LSTM), and convolutional neural networks (CNN) combined with LSTM.

3.1. Long-Short-Term Memory (LSTM) Architecture

In this part of the research, we used the LSTM model to classify falls and ADLs activities; the design of LSTM is a part of recurrent neural networks (RNNs). LSTM consists of six layers: three LSTM layers, a dropout layer, a dense layer, and a Softmax layer. Figure. 3 shows the LSTM architecture and the used layers.

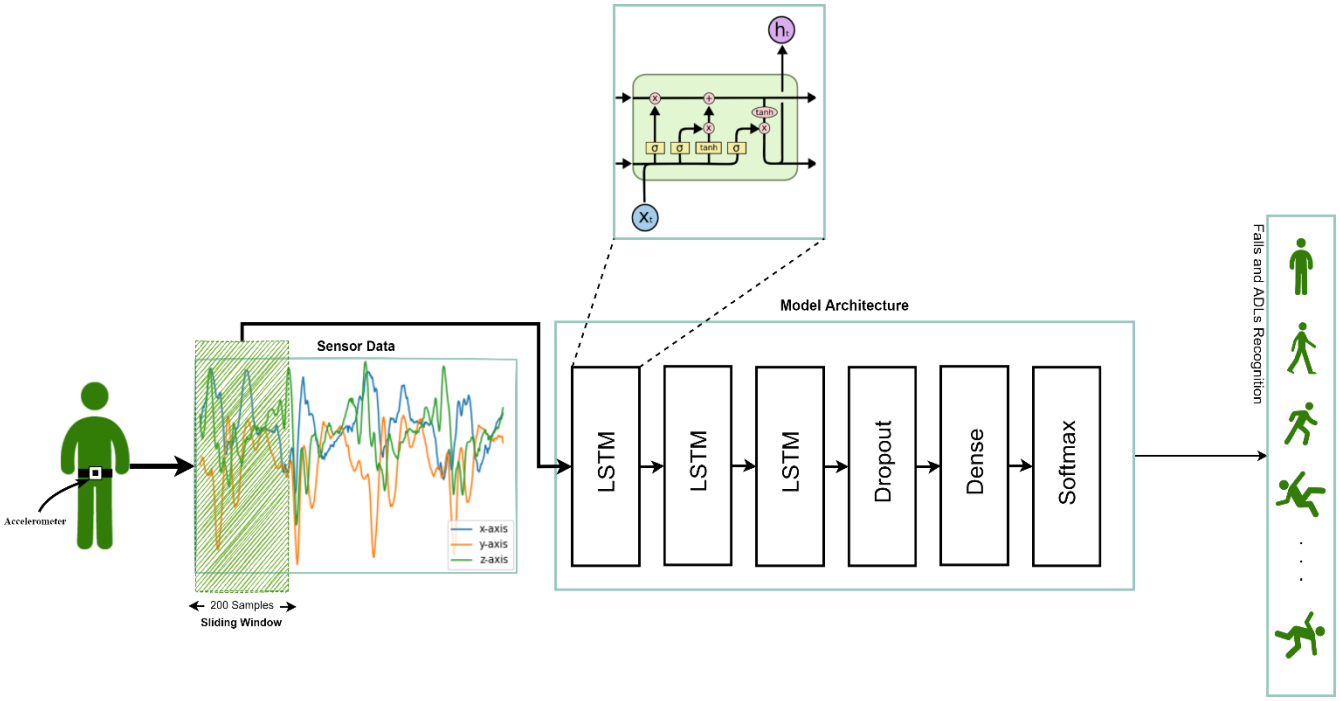


Figure 3. LSTM architecture and its layers

LSTM can process a single sequence (sample) at each time step for different motions and can process an entire sequence of data (samples). It learns future dependencies between time steps associated with the input data. LSTM layer accepted the data to be three-dimensional shape; these three dimensions are samples, time-steps, and features.

3.2. CNN-LSTM architecture

Convolutional neural networks (CNN) combined with long-short term memory (LSTM) architecture have been used. Which we think is more suitable for fall and ADLs recognition than LSTM alone. CNN-LSTM hybrid models inspired from [14], [15]. CNN was designed to process images and classify them by a 2D filter. However, the acceleration data from the Sisfall dataset is time-series data with samples and time steps (in seconds). For that reason, we need a filter of one dimension instead of two dimensions. 1D-CNN model is used to extract useful features from time-series data and used for time series forecasting. 1D CNN can process data and extract discriminative features with a fixed sliding window from a dataset. The sliding window for our dataset has a fixed window length of 200 samples. The CNN architecture has four convolutional layers with ReLU activation, followed by a Batch-Normalization layer and Max-Pooling layer, as shown in the figure. 4, the output data from the previous CNN layer is passed through a flattened layer and then passed through three LSTM layers, a dense layer, and a Softmax layer, which is the output. Figure. 4 shows the CNN-LSTM architecture.

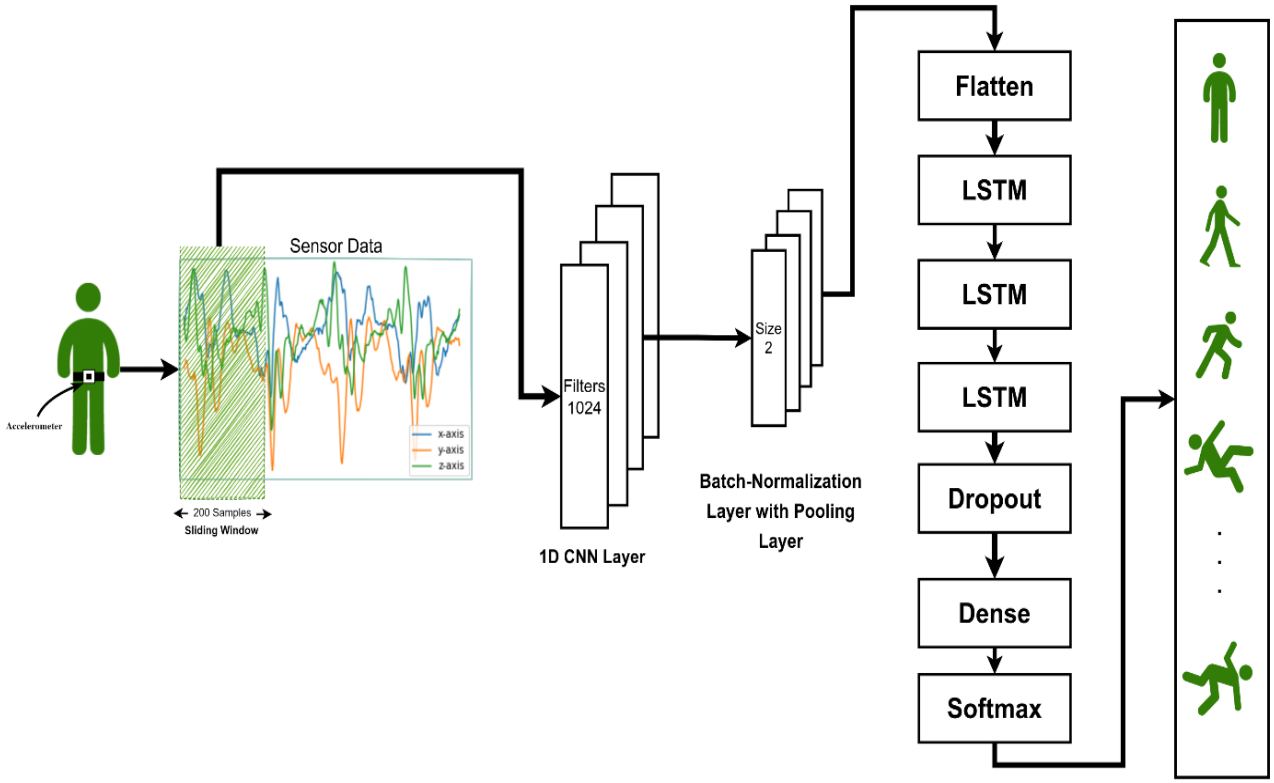


Figure 4. CNN-LSTM architecture

We used TensorFlow [11], supported by Python programming language, a deep learning framework, to build the CNN-LSTM model. The LSTM layer learns future dependencies between time steps associated with the input data. To have many batches of data close at hand, it is necessary to preprocess and split each class in this dataset into a fixed set of samples and store them into segments. This approach is applicable for CNN and LSTM layers. Hence, we chose a sliding window of 200 pieces with a step size of 40 works to be about 5 seconds worth of data at a time. These segments can be fed into the CNN layer for the feature extraction phase, and the LSTM layer will learn what features belong to what class. 1D CNN layer processes data in one dimensional and the data must shaped as input_shape (time-steps, parts for the time-steps). Figure.5 illustrates 1D CNN with its time-steps and features; CNN has the kernel size, which is the filter size that can move along the axis of time.

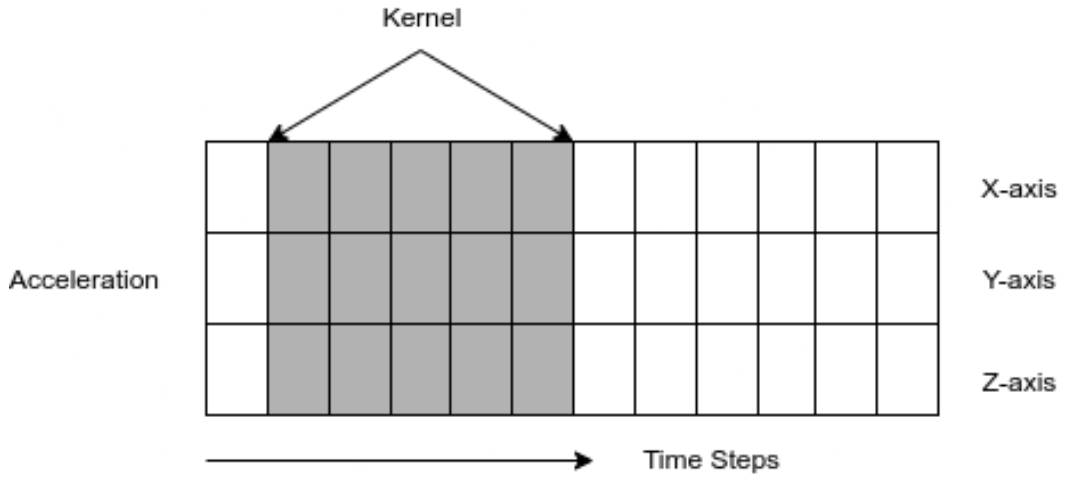


Figure 5. Illustration of the input shape for the 1D CNN layer

There are three features at the first layer: the three-axis acceleration raw data. We scaled the raw data to a fixed range between -1 and 1 through Min-Max scaling. These scaling factors restrain the effect of outliers, observations, or null values in the acceleration values. Equation (1) describes the Min-Max scaling that is usually done in deep learning models:

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

We divide the dataset into training, validation, and testing sets of 60:20:20 of the entire dataset. Figure.5 shows an illustration of these proportions. We then sampled randomly from the random state with 40 different random states. The reason to have validation data is to evaluate the quality of the model and avoid under- and overfitting during the training process.

Figure 6. Uniform datasets splitting to training set, validation, testing sets

To alleviate the overfitting on the training data, which results in a low accuracy on the validation dataset and testing dataset, a dropout layer is used with a scalar value of (0.3 ~ 0.5).

3.3. Implementation of real-time fall and ADL recognition system

3.3.1. TensorFlow Lite

TensorFlow Lite is an open-source deep learning framework from Google Company [12]. It's a flexible platform that allows the deployment of pre-trained neural network models for on-device inference with high reaction time and small file size that applies for embedded and IoT devices. TensorFlow Lite supports many languages such as Python, JavaScript, R, and Swift. Figure.7 shows the proposed real-time prototype for the fall and ADL recognition system.

3.3.2. Hardware implementation

Raspberry Pi, a minicomputer, was used in this system as the primary hardware. Raspberry Pi carries a Linux operation system and can be used as a developer platform using its pins or as a computer. We have used it as a developer board to evaluate our model in our work. ADXL345 accelerometer sensor has been used to collect tri-axial acceleration data from the outside world. The hardware devices used in the proposed real-time recognition system are shown in Figure.8. Moreover, TensorFlow lite models can work with different devices like an app on a mobile phone or with Microcontroller units (MCUs).

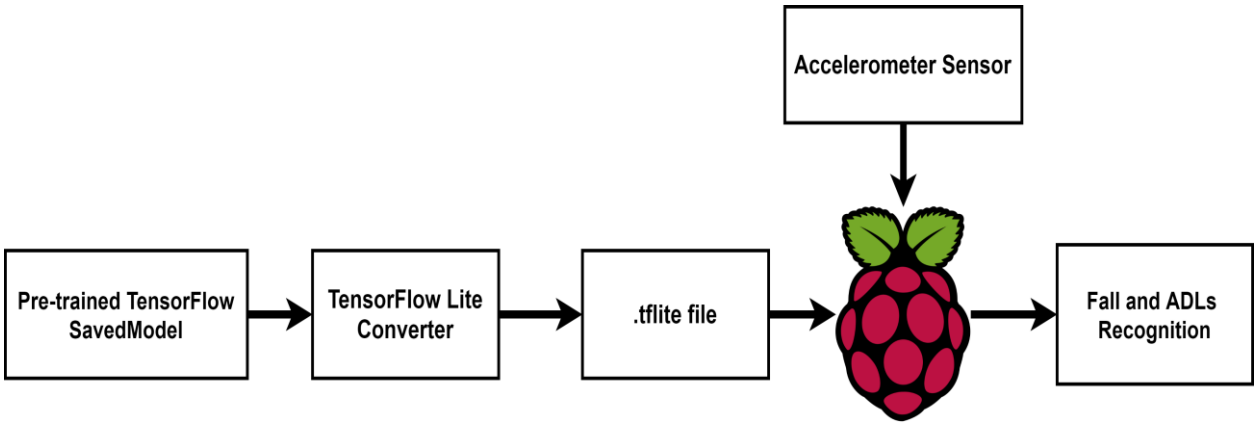
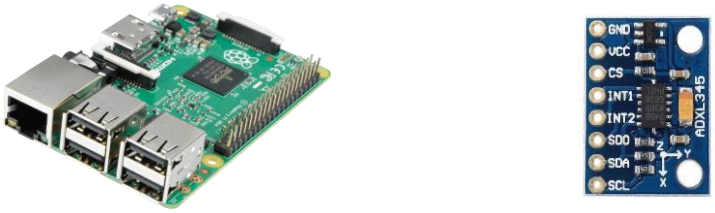


Figure 7. The proposed real-time recognition system



(a) **(b)**

Figure 8. Hardware (a) Raspberry Pi 3 Model B+, (b) ADXL345 sensor

3.3.3. Software implementation

Python was used as the primary programming language to implement this system; the code is straightforward to be implemented. It starts with importing the TensorFlow Lite file into the Python script and then resizing or manipulating the input tensor shape to the corresponding inputs from the ADXL345 sensor. For real-time recognition systems, inference latency should be minimized, and this could be achieved by sampling the ADXL345 sensor readings with a fixed sample set.

4. Results and discussion

This section presents the results of training our LSTM and CNN-LSTM models to classify falls and activities of daily living (ADLs). Experimental results showed an accuracy of 93.11% using the pure LSTM model. Figures 9 and 10 show the training and validation loss and confusion matrix of the LSTM model, respectively.

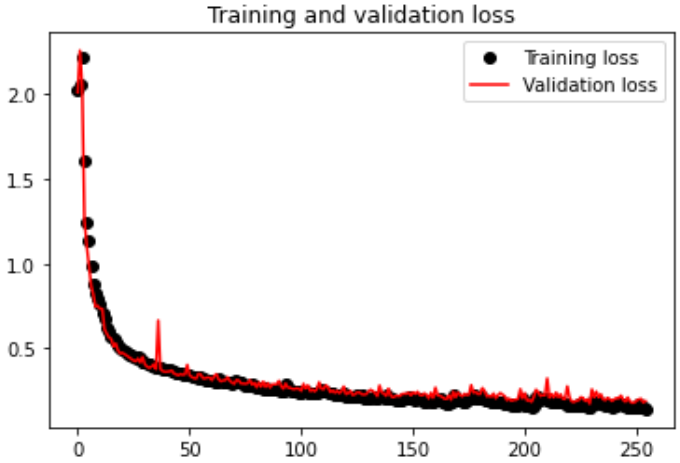


Figure 9. Training and validation loss of LSTM model

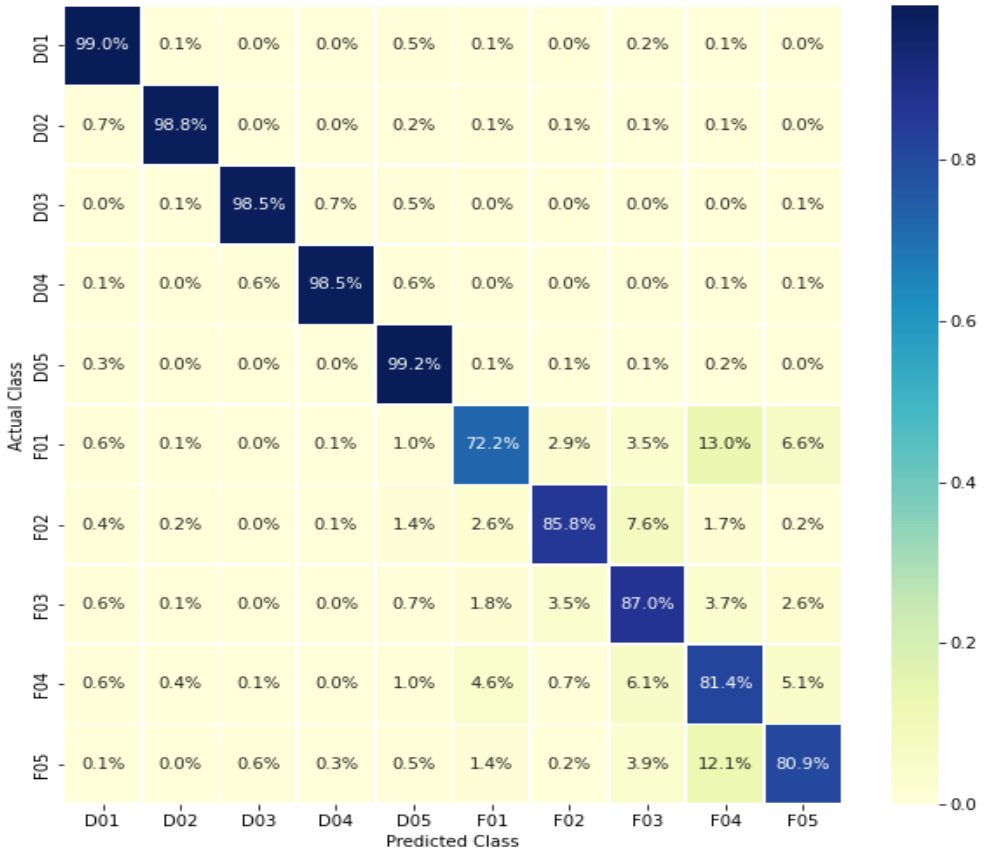


Figure 10. Confusion Matrix using LSTM

We used the CNN-LSTM hybrid model to classify falls and ADLs events in the second experiment. After many tests with different epochs and batch sizes, we decided to use an epoch of 300 and a batch size of 128. The experiment was conducted with other optimizers such as RMSprop [16], Adam [17], and stochastic gradient descent (SGD). We decided to choose an Adam optimizer with a learning rate of 0.001. Choosing the learning rate value could be frustrating; a low learning rate value could take a very long time to train or fail at training; in contrast, a high learning rate value can lead to difficulties in converging loss. This is the trade-off we should consider when picking the learning rate value. The same applied to the dropout rate but with a different scenario. A high dropout value could cancel out many weights and biases at training time and result in undesirable performance. It is worth noting that the training time is a linear function of the hyper-parameter. It is observed from the results that there is a trade-off between training time and the hyper-parameters used during the training process. The validation accuracy must be stable to prevent the model from overfitting towards the training data. The training process needs a computer with a high-performance graphics card; because of this limitation, the model was trained online on Kaggle website with a GPU accelerator and implemented using Python 3.7 environment, several experiments (average training time of ~ 8 hours) were performed. Experimental results showed an accuracy of 98.18%. Figures 11 and 12 show the training and validation loss and confusion matrix of our CNN-LSTM model, respectively.

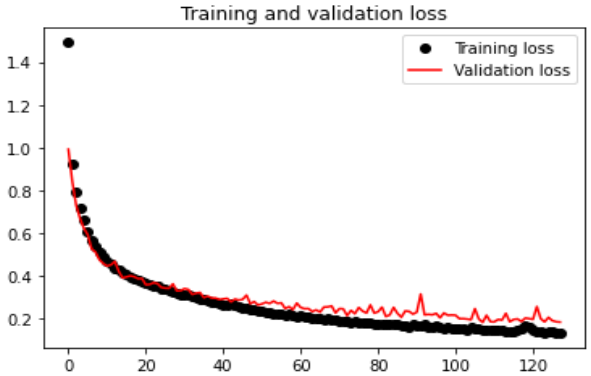


Figure 11. Training and validation loss of CNN-LSTM model

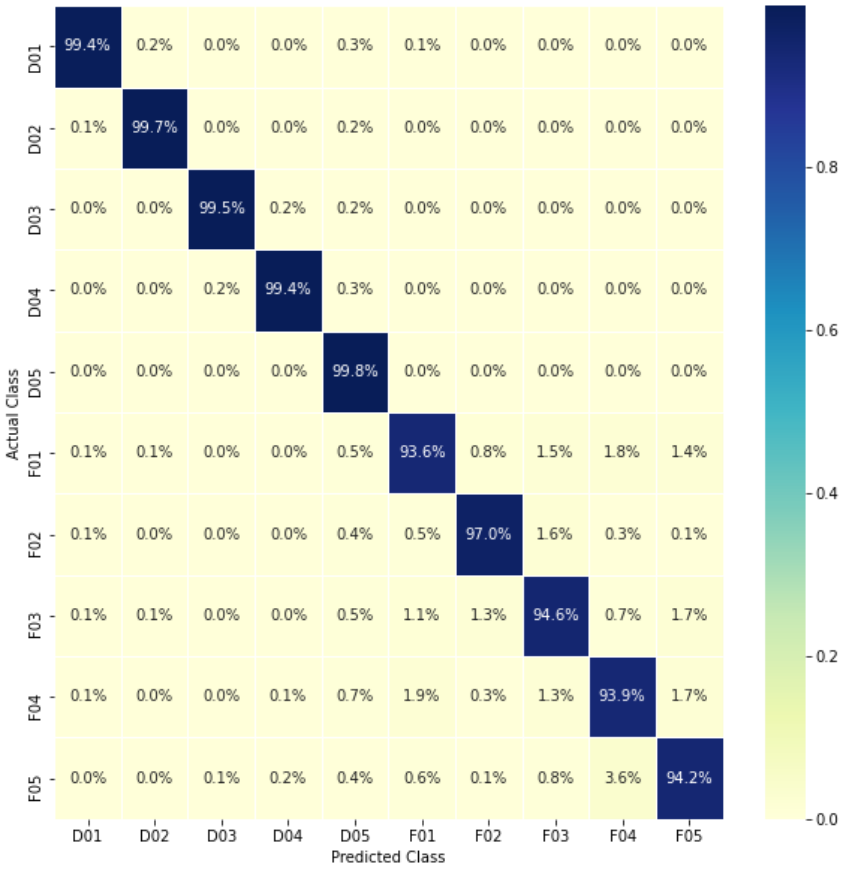


Figure 12. Confusion matrix using CNN-LSTM

4.1. Classification performance

The classification performance for both LSTM and CNN-LSTM models is represented by different metrics, including sensitivity, specificity, and accuracy; these metrics can be calculated by equations 1, 2, and 3, respectively

$$\text{Sensitivity} = \frac{TP}{TP + FN} \quad (1)$$

$$\text{Specificity} = \frac{TN}{FP + TN} \quad (2)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (3)$$

Where true positive (TP), a fall has occurred, and the model correctly predicts that. False-positive (FP), the model predicts normal ADL action as a fall. True negative (TN), a fall occurred, but the model predicts it as ADL. False-negative (FN) fall occurred, but the model does not predict it. Sensitivity measures the model's ability to predict all actual falls; specificity measures the ADL prediction rate; accuracy, which is the proportion of accurate prediction results in this model. Table 3 summarizes the classification performance.

Table 3. Classification results of LSTM and CNN-LSTM model

	LSTM	**CNN-LSTM**
Sensitivity (%)	89.65%	97.11%
Specificity (%)	99.25%	99.78%
Accuracy (%)	93.11%	98.18%

We also compared our results with several other algorithms evaluated on Sisfall public dataset, it should be noted that some of the algorithms are binary classification method, comparison of the performance is given in Table.3 below.

Table 3. Similar work on falls and ADLs recognition

Research	Dataset	Model	Classification accuracy
Liu et al 2018 [18].	Sisfall	SVM	97.6%
Musci et al 2018 [8].	Sisfall	RNN-LSTM	97.16%
Ours	**Sisfall**	**CNN-LSTM**	**98.18%**

From the compression table above, our proposed model shows the higher accuracy using CNN-LSTM model and raw acceleration data. Traditional machine ;learning like SVM and KNN needs a lot of feature engineering related to time-domain and frequency-domain nature of acceleration data, dealing with traditional machine learning is obsolete in most cases these days. CNN model make it is easier by its ability for automatic feature extraction using convolution filters.

4.2. Results of real-time CNN-LSTM fall and activities of daily living (ADL) recognition

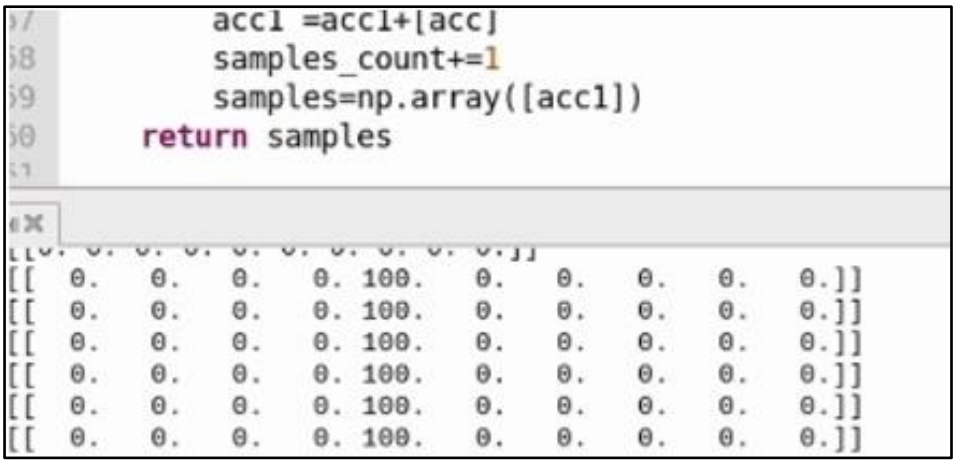
The implementation of the real-time falls and ADLs recognition will be presented here. The prototype is tested on an individual of 30 years old for three classes, each class for 15 seconds. The real-time device setup for the participant and the results are shown in figures 13 and 14, respectively.

Figure 13. The setup of the real-time prototype

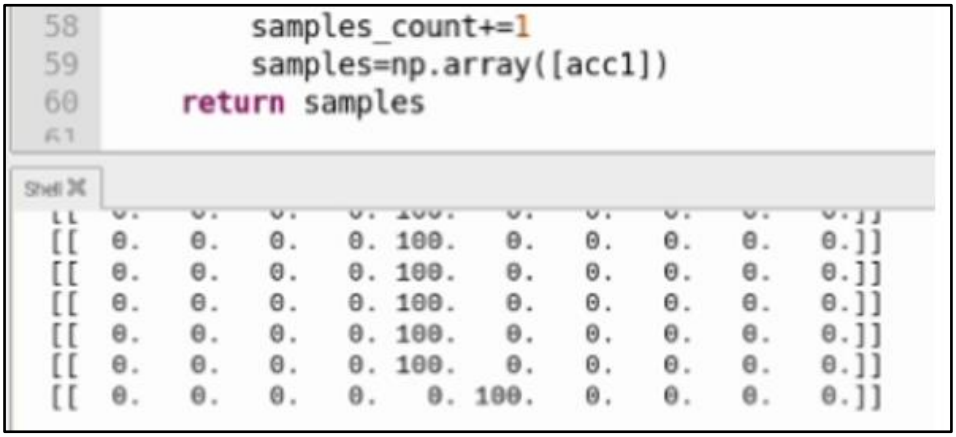
```
        samples=np.array([acc1])
    return samples

D01 D02 D03 D04 D05 F01 F02 F03 F04 F05
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

(a)



(b)



(c)

Figure 14. Results of the real-time falls and ADLs recognition prototype,
(a) the individual performs "standing," and the model predicted that class correctly,
(b) the individual performs "strolling," the model predicted as "walking upstairs and downstairs,"
(c) the individual performs falling forward, and the model predicts this class correctly

After running the Python script responsible for the real-time recognition process, an array comprising ten elements is shown as the printed output. The array holds ten classes that we trained our model on. Figure.10 (a) and (c) both show that the recognition process for the real-time process is done correctly. Unfortunately, the model misclassified "walking slowly" as "walking upstairs and downstairs." This misclassification occurs in real-time and is based on the several experiments we have done for this. Hence, it is worth saying: first, the accelerometer sensor should be positioned in a fixed place to result in inappropriate readings. Secondly, further calibration of the accelerometer sensor should be done, and the real-time model is susceptible to variations in acceleration values. Besides these observations, the sampling rate of the collected data and the scaling process applied during training must be met during the real-time process. This means that the captured data from the accelerometer sensor should be sampled at 200 Hz, and Min-Max scaling should be applied too.

5. Conclusion

First, a multiclass LSTM model has been developed to discriminate various types of falls and ADLs. The LSTM model has achieved an overall accuracy of 93% at the testing phase. It has been observed that the LSTM has a slightly degraded performance on resolving various types of falls caused by a slip as opposed to falls caused by a trip. A combined CNN-LSTM model has been proposed and implemented to improve classifier efficiency. The proposed model has a superior performance over the LSTM network. We have reached an overall accuracy of 98.74% in the testing phase. On the hardware side, we have converted the trained CNN-LSTM model to TensorFlow Lite for implementation in Raspberry PI for on-device inference. The model has been tested against data collected using the ADXL345 accelerometer sensor, and we have successfully shown that the model provides acceptable results for real-time recognition.

References

[1] World Health Organization, “WHO global report on falls prevention in older age,” 2008.

[2] D. Wild, U.S. Nayak, B. Isaacs, "How dangerous are falls in older adults at home?", *Br. Med. J.*, vol.282, pp.266–268, 1981.

[3] D. Razum, G. Seketa, J. Vugrin, and I. Lackovic, “Optimal threshold selection for threshold-based fall detection algorithms with multiple features,” *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, pp. 1513–1516, 2018, doi: 10.23919/MIPRO.2018.8400272.

[4] A. Sucerquia, J. D. López, and J. F. Vargas-Bonilla, “SisFall: A fall and movement dataset,” *Sensors (Switzerland)*, vol. 17, no. 1, 2017, doi: 10.3390/s17010198.

[5] S. Wilhelm and R. Malekian, “Human Activity Recognition Using LSTM-RNN Deep Neural Network Architecture,” *IEEE*, 2019.

[6] I. Wayan Wiprayoga Wisesa and G. Mahardika, “Fall detection algorithm based on accelerometer and gyroscope sensor data using Recurrent Neural Networks,” *IOP Conference Series: Earth and Environmental Science*, vol. 258, no. 1, 2019, doi: 10.1088/1755-1315/258/1/012035.

[7] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, “UMAFall: A Multisensor Dataset for the Research on Automatic Fall Detection,” in *Procedia Computer Science*, 2017, vol. 110, pp. 32–39. doi: 10.1016/j.procs.2017.06.110.

[8] M. Musci, D. de Martini, N. Blago, T. Facchinetti, and M. Piastra, “Online fall detection using recurrent neural networks,” *arXiv*, no. Dl, 2018, doi: 10.1109/TETC.2020.3027454.

[9] T. H. Kim, A. Choi, H. M. Heo, K. Kim, K. Lee, and J. H. Mun, “Machine Learning-Based Pre-Impact Fall Detection Model to Discriminate Various Types of Fall,” *Journal of Biomechanical Engineering*, vol. 141, no. 8, pp. 1–10, 2019, doi: 10.1115/1.4043449.

[10] J. Xu, Z. He, and Y. Zhang, “CNN-LSTM Combined Network for IoT Enabled Fall Detection Applications,” *Journal of Physics: Conference Series*, vol. 1267, no. 1, 2019, doi: 10.1088/1742-6596/1267/1/012044.

[11] [Online] “TensorFlow – An open-source software library for Machine learning and deep learning models “URL: https://www.tensorflow.org [accessed: 2021-07-11].

[12] [Online] “TensorFlow Lite – A library for deep learning models on-device inference “URL: https://www.tensorflow.org/lite [accessed: 2021-07-11].

[13] M. Musci, D. De Martini, N. Blago, T. Facchinetti, and M. Piastra, “Online Fall Detection using Recurrent Neural Networks on Smart Wearable Devices,” IEEE Trans. Emerg. Top. Comput., vol. XX, no. XX, pp. 1–13, 2020, doi: 10.1109/TETC.2020.3027454.

[14] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.- c. Woo, “Convolutional lstm network: A machine learning approach for precipitation now casting,” in Advances in neural information processing systems, 2015, pp. 802–810.

[15] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, “Convolutional, long short-term memory, fully connected deep neural networks,” in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2015, pp. 4580–4584

[16] T. Tieleman and G. Hinton, “Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning,” Tech. Rep., Technical report, p. 31, 2012.

[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[18] K. Liu, C. Hsieh, S. J. Hsu and C. Chan, "Impact of Sampling Rate on Wearable-Based Fall Detection Systems Based on Machine Learning Models," in IEEE Sensors Journal, vol. 18, no. 23, pp. 9882-9890, 1 Dec.1, 2018, doi: 10.1109/JSEN.2018.2872835.