Building fall detection models using LSTM, GRU, and BERT on SmartWatch Dataset

|  |  |
| --- | --- |
| Thi Thuy Tien Tran  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  ttran15@lakeheadu.ca | Azeem Abdul  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  aabdul13@lakeheadu.ca |
| Darcus Angeline Peter  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  dpeter3@lakeheadu.ca | Nimasha Warnakulasuriya  *Department of Computer Science*  *Lakehead University*  Thunder Bay, Ontario, Canada  nwarna@lakeheadu.ca |

*Abstract*—

*Index Terms*—fall detection, deep learning, BERT, LSTM, GRU, SmartWatch dataset

# INTRODUCTION

Fall has become a common problem for old peope all over the world. According to World Health Organization (WHO), every year, there are 28–35% of people aged 65 and above fall **[data]**. In long-term care institutions, 30%–50% experience falls yearly, with frequent recurrences **[system2]**. As human beings getting aged, the occurrence of falls rate and its health consequences increase **[data1]**. Falls contribute significantly to injuries, ranging from mild to severe, and even mortality. In fact, falls are responsible for 20%–30% of injuries and 40% of injury-related deaths **[system2]**. The treatment of unintentional injuries caused by falls is one of the most expensive **[reason1]**. Fall-related hospitalizations average cost for those aged 65 and above are $17,483 per hospital stay in 2004 in the US, and are projected to reach $240 billion by 2040 **[system2]**.

Internet of Things (IoT) refers to a network of internet-connected devices **[data4]**. IoT can be applied in health care for enhancing patient care, reducing costs, improving access to treatment, educating people on healthy lifestyles, enhancing quality of life **[data4]**. With the rapid development of IoT, using fall detection system to monitor daily behaviors and detect falls among the elderly is of great importance **[data5]**. It can potentially alleviate caregiver burden and decrease healthcare expenses by facilitating early intervention **[data3]**. Although a fall detection system does not prevent the fall, the information that provides is valuable and can be used by both carers and medical professionals **[system3]**.

Although deep learning has been used widely in building fall detection models, there is no article about applying BERT on Smart Watch dataset to build fall detection models.

This paper proposes a fall detection framework based on wearable-sensor fall detection dataset using deep learning models. The main contributions of this paper are:

- Developing a fall detection models based on SmartWatch fall datasets.

- Conduct a comparison analysis to verify the difference between state-of-the-art model BERT with others traditional deep learning models such as GRU and LSTM.

Our code can be found on this Github [repository](https://github.com/ttran15/fall_detection).

The rest of the paper is organized as follows: Section II reviews related works; Section III discusses methodology; Section IV presents the result; and finally, Section VI concludes the work.

# RELATED WORKS

The evolution of modern technologies developed by researchers to identify and prevent falls in elderly patients has progressed significantly over the past few years, so several sophisticated techniques have been presented to tackle the issue of senior people falling **[data3]**.

Current studies on fall detection can be grouped into three categories. The first one uses wearable sensors to detect the occurrence of a fall and uses data such as a three- axis accelerometer to detect a person′s posture. The second type uses environmental sensors to detect falls through vibration, infrared sensors, or audio. The third type adopts computer vision and uses video or image sequences taken by surveillance cameras to detect falls **[system4]**.

Regarding wearable sensors,

a fall detection system architecture using multiple sensors with four traditional machine learning algorithms (SVM, NB, Decision Tree and KNN) was studied. The paper is the first to propose using ANOVA analysis to evaluate the statistical significant of differences observed by varying the number of sensors and the choice of a particular machine learning algorithm. The main conclusion from this paper is that sensors placed close to the gravity center of the human body (i.e., chest and waist) are the most effective **[reference]**

**[data]** We experimented with both traditional (Support Vector Machine and Naive Bayes) and non-traditional (Deep Learning) machine learning algorithms for the creation of fall detection models using three different fall datasets (Smartwatch, Notch, Farseeing). Our results show that a Deep Learning model for fall detection generally outperforms more traditional models across the three datasets

Regarding environmental sensors,

Environmental-device fall detection approaches mainly use floor vibration, sound or infrared images as features and combine machine learning and neural networks for fall detection. This method is less accurate than the wearable-device fall detection methods, because the sensing range is larger and it is susceptible to environmental in- terference. The cost is also high as more equipment is required to be installed in the sensing environment. **[system4]**.

Our system is robust to identify fall downs from other possible similar events, like jumps, door close, objects fall down, etc. Such a smart system can also be connected to smart commercial devices (like GOOGLE HOME or AMAZON ALEXA) for emergency notifications. Our approach represents an advance in smart technology for elder people who live alone. Evaluation of the system shows it is able to detect fall downs with an acceptance rate of 95.14% (distinguishing from other possible events), and it identifies people with one or two steps in a 97.22% (higher accuracy than other methods that use more footsteps) **[reference1]**

Our proposal is using very-low-resolution thermal sensors for classifying a fall and then alerting to the care staff. Also, we analyze the performance of three recurrent neural networks for fall detections: long short-term memory (LSTM), gated recurrent unit, and Bi-LSTM. **[reference2]**

Regarding computer vision,

Computer-vision fall detection approaches use image processing and a proba- bilistic model to detect human contour and applied machine learning or neural network for movement classification. It does not need high-cost equipment, it is susceptible to environmental interference such as light, clothing, or overlapping portraits. There are also concerns about privacy with this approach. Skeleton recognition is less likely to be disturbed by the environment than human contour recognition, with an improved privacy protection. **[system4]**

The proposed approach presented here extracts motion information using best-fit approximated ellipse and bounding box around the human body, produces projection histograms and determines the head position over time, to generate 10 features to identify falls. These features are fed into a multilayer perceptron neural network for fall classification. Experimental results show the reliability of the proposed approach with a high fall detection rate of 99.60% and a low false alarm rate of 2.62% when tested with the UR Fall Detection dataset. **[reference3]**.

This paper proposes a fall detection framework based on wearable-sensor fall detection dataset using deep learning models.

# METHODOLOGY

## Dataset

SmartWatch dataset can be download via this [link](https://userweb.cs.txstate.edu/~hn12/data/SmartFallDataSet/SmartWatch/).

The Smartwatch dataset was collected from seven volunteers each wearing a MS Band watch. These seven subjects were all of good health and were recruited to perform simulated falls and ADLs. Their ages ranged from 21–55, height ranged from 5 ft to 6.5 ft. and the weight from 100 lbs to 230 lbs. Each subject was told to wear the smartwatch on his/her left hand and performed a pre-determined set of ADLs consisting of: jogging, sitting down, throwing an object, and waving their hands. This initial set of ADLs were chosen based on the fact there are common activities that involved movement of the arms. These datasets were automatically labeled as “NotFall”. We then asked the same subject to perform four types of falls onto a 12-inch-high mattress on the floor; front, back, left, and right falls. Each subject repeated each type of fall 10 times. We experimented with the sampling rates of 4 Hz, 1.25 Hz, and 62.5 Hz supported by the smartwatch, and settled with 31.25 Hz. We found that the data sampling frequency of 4 Hz, although it consumes fewer resources, missed too many critical sample points within the critical phase of a fall. On the other hand, the use of the higher sampling frequency of 62.5 Hz provided by the watch was flooding the application with too much data and incurred a high computation cost which is impractical for real-time prediction of falls. We implemented a data collection service on the smartphone to have a button that, when pressed, labels data as “Fall” and otherwise “NotFall”. Data was thus labeled in real time as it was collected, by the researcher holding the smartphone. However, the pressing of the button can introduce errors such as the button is being pressed too late, too early, or too long for a fall activity. To mitigate these errors, we post-processed the collected data to ensure that data points related to the critical phase of a fall were labeled as “Fall”. This is done by checking that for each fall data file, the highest peak of acceleration, and data points before and after that point, were always labeled as “Fall”. **[data]**

TABLE 1

SmartWatch dataset description

|  |  |  |  |
| --- | --- | --- | --- |
|  | Fall | Not Fall | Sum |
| Training | 4,550 | 29,470 | 34,020 |
| Testing | 2,275 | 14,954 | 17,229 |
| Sum | 6,825 | 44,424 | 51,249 |

One of the major sensors used in fall detection on a smartwatch is an accelerometer, which measures the acceleration of an object. Acceleration is the change in velocity with respect to time and velocity represents the rate at which an object changes its position. Acceleration data is commonly used in fall detection because accelerometer sensors are found in most smart devices, and a distinct change in acceleration happens when a fall occurs. **[system1]**

Firstly, elderly patients with a smartwatch and three-axis accelerometer readings are used to measure daily body motion in x-y-z coordinates **[data3]**

Figure 1 illustrates the data pre-processing phase, in which signal values are segmented for training and detection. For model training, data segments were labeled using two classes: Fall and ADL (activities of daily living, i.e., non-fall). This shows how the data signals of human activities were pre- processed easily and transferred on time. **[data2]**

A graph showing a number of different colored lines

Description automatically generated with medium confidence

Figure 1. the data pre-processing phase

The dataset is of binary nature having labels of ”Fall” and ”Not Fall”. The name of columns in the original dataset is accelerometer\_x, accelerometer\_y, accelerometer\_z, and outcome. **[data1]**

## Techniques

One of the disadvantages of traditional machine learning algorithms is the need for a priori feature extraction from the data. Feature extraction and selection are tasks that need to be performed before any learning can occur. The types of features to be extracted have to be manually specified and thus their effectiveness heavily depends on the ingenuity of the researcher. [data]

On the other hand, the deep learning methods have ample way of tuning the hyperparameters. Hence, deep learning methods are in a position to address the limitations associated with the conventional machine learning methods [28]. [data1]

In this research, we use three kinds of deep learning models including LSTM, GRU, and BERT.

### LSTM

The breakdown of the LSTM model is first to import all the necessary libraries. For LSTM used numb ,torch, matplotib,and a few libraries from sklearn.metrics for getting the numeric values of the model performance. Then the dataset is loaded and the fall and not-fall dataset was combined and converted from numpy arrays to PyTorch tensors as using tensors would be beneficial when processing as batches later in the model .The next step would be to initialize the model it defines an LSTM network with an LSTM layer, a fully-connected layer and a ReLU activation .Following it would be to set up the parameters starting off by creating an instance of LSTMNet and criterion to define the loss function along with optimizer with learning rate of 0.001.Then the lists are made to keep track of training and validation losses and accuracies for each epoch. We have set the epoch=10 for the model and all the different metrics are calculate for each one. Now to proceed with the training loop and iterate over the training data we calculate the forward pass that compute the model output, the loss using criterion and compute the gradients and update the model parameters and store the track metrics. The validation section starts by setting the model to evaluation mode and iterating over the validation data and store the validation metrics. Finally print the training and validation losses and accuracy for each epoch and provide the metrics on the final validation set which includes accuracy, precision,Recall,F1 score, classification report and the ROC-curve.

### GRU

### BERT

## Experience designs

**[data]** RNNs are traditionally trained with backpropagation through time (BPTT), so it is necessary to specify how many steps n in the past the network should be trained on. This parameter is important since our falls and activities occur over a period of time. If we train the network on only a few steps in the past, it will not capture the full scope of the activity. However, if we train it over too many steps, the network may take into account past accelerometer data that is not relevant. We settled on using n ≈ 40 steps for this model. This means each prediction the model makes takes into account ≈1.28 s of data. This is enough time to capture the aspects of a fall, as well as to rule out some activities as falls.

Model predictions (i.e., predictions produced by the neural architecture) begin once the number of sensor data points acquired is equal to the number of steps. Every model prediction thereafter will only require one additional data point, as the model will slide one data point at a time, reusing all of the previous data points except for the least recent. However, before producing a final prediction, we generate a heuristic value based on the probabilities produced by several model predictions. We compute the average value of 10 consecutive probabilities, and compare this with a pre-defined threshold value. If the average probability breaches this threshold, then it is considered a fall prediction. This helps to avoid isolated positive model predictions from triggering a false positive. Figure 5 outlines this schematic.

Prediction scheme is shown in Figure 2.

A diagram of a model output

Description automatically generated

Figure 2. Prediction scheme

Data is split into training and testing dataset with ratio 70:30. Training dataset is used to train different models including LSTM, GRU and BERT. Testing dataset is used to evaluate model performance. The best model with the highest F1-score on testing dataset are chosen.

Model parameters with different algorithms are presented in Table 2.

TABLE 2

Model parameters

|  |  |
| --- | --- |
| Model | Parameters |
| GRU | input\_size=2, hidden\_size=256, num\_layers=2, lr = 0.0001, n\_epochs = 10 |
| LSTM | input\_size=2, hidden\_size=256, num\_layers=2, lr = 0.0001, n\_epochs = 10 |
| BERT |  |

## Environment Setup

In this project, we use Python and Jupyter Notebook for code implementation. To be specific, the environment setup are:

* Platform: Pytorch
* Libraries:

## Evaluation Metrics

We measured the performance of the proposed framework by calculating accuracy, precision, recall, and F1-score **[data2]**.

Accuracy is the rate of correct classification **[data7]**.

A black text on a white background

Description automatically generated

A precision score of 100% for binary classification means that every item labeled as belonging to the positive class does indeed belong to the positive class **[data7]**.

A black text on a white background

Description automatically generated

Sensitivity (or true positive rate) is the probability of a positive test result amongst those having the target condition **[data7]**.

A black text on a white background

Description automatically generated

Specificity (or true negative rate) is the probability of a negative test result amongst those without the target condition **[data7]**.

A black text on a white background

Description automatically generated

Since recall and precision are not effective in imbalanced class problems, we have used the harmonic mean of these two metrics namely weighted F1-score **[data1]**.

A black text on a white background

Description automatically generated

# RESULTS

LSTM

GRU

BERT

# CONCLUSION

Our contributions in this project are regarding building state-of-the-art model on Smart Watch dataset and comparing performance of different deep learning methods.

In the future, this project can be extended by integrating building model into an interactive website so that users can conveniently calculate their risk of heart attack.

# REFERENCES

[1] Imen Boudali, Sarra Chebaane, and Yassine Zitouni, “A predictive approach for myocardial infarction risk assessment using machine learning and big clinical data,” *Healthcare analytics*, vol. 5, pp. 100319–100319, Jun. 2024, doi: https://doi.org/10.1016/j.health.2024.100319.

[2] C. D. Fernando, P. T. Weerasinghe, and C. K. Walgampaya, “Heart Disease Risk Identification using Machine Learning Techniques for a Highly Imbalanced Dataset: a Comparative Study,” *KDU Journal of Multidisciplinary Studies*, vol. 4, no. 2, p. 43, Dec. 2022, doi: https://doi.org/10.4038/kjms.v4i2.50.

[3] I. Rojek, M. Kozielski, J. Dorożyński, and D. Mikołajewski, “AI-Based Prediction of Myocardial Infarction Risk as an Element of Preventive Medicine,” *Applied Sciences (2076-3417)*, vol. 12, no. 19, p. 9596, Sep. 2022, doi: https://doi.org/10.3390/app12199596.

‌[4] M. Oliveira, J. Seringa, Fausto José Pinto, R. Henriques, and T. Magalhães, “Machine learning prediction of mortality in Acute Myocardial Infarction,” *BMC Medical Informatics and Decision Making*, vol. 23, no. 1, Apr. 2023, doi: https://doi.org/10.1186/s12911-023-02168-6.

‌[5] K. I. Shakhgeldyan, N. S. Kuksin, I. G. Domzhalov, V. Y. Rublev, and B. I. Geltser, “Interpretable machine learning for in-hospital mortality risk prediction in patients with ST-elevation myocardial infarction after percutaneous coronary interventions,” *Computers in Biology and Medicine*, vol. 170, p. 107953, Jan. 2024. doi: https://doi.org/10.1016/j.compbiomed.2024.107953.

[6] Z. Chen et al., “Prediction of Myocardial Infarction From Patient Features With Machine Learning,” *Frontiers in Cardiovascular Medicine*, vol. 9, Mar. 2022, doi: https://doi.org/10.3389/fcvm.2022.754609.

‌[7] G. R. Chaudhari et al., “Deep learning augmented ECG analysis to identify biomarker-defined myocardial injury,” Scientific Reports, vol. 13, no. 1, Feb. 2023, doi: https://doi.org/10.1038/s41598-023-29989-9.

[8] Centers for Disease Control and Prevention. “Behavioral Risk Factor Surveillance System.” CDC. https://www.cdc.gov/brfss/index.html (accessed May. 12, 2024).