Sensitivity Analysis of LSTM Networks for Fall Detection Wearable Sensors

João P. Matos-Carvalho*, Sérgio D. Correia*† and Slavisa Tomic*
*COPELABS, Universidade Lusófona, Campo Grande 376, 1749-024 Lisbon, Portugal
†VALORIZA, Instituto Politécnico de Portalegre, Campus Politécnico n.10, 7300–555 Portalegre, Portugal
Correspondence: joao.matos.carvalho@ulusofona.pt

***Abstract*—Falling detection concerns identifying and alerting when a person falls or experiences a sudden loss of balance. It is an important safety feature, particularly for elderly or vulnerable individuals, as falls can lead to severe injuries. One applicable technique is employing Long Short-Term Memory (LSTM) networks that can be used for falling detection by processing data from sensors, such as accelerometers or gyroscopes, to identify patterns associated with falls. This paper demonstrates that a single LSTM layer can be used for this purpose. Still, multiple LSTM layers can improve the system's performance with less processing time, reaching 99.13% of accuracy with 4.35% of loss values. Furthermore, this work thoroughly studies the sensitivity of the size of an LSTM cell, demonstrating that it is not necessary to obtain a size close to the timestamp of the features and thus save hardware resources.**

***Index Terms*—Artificial Intelligence, Embedded Computing, Fall Detection, LSTM, Machine Learning, Wearable Sensors**

I. Introduction

A. Motivation

One of the most significant health risks for seniors worldwide is falling. The cost of treating injuries sustained in the aftermath of falls and providing emergency services is substantial. If a fall is precisely anticipated or recognized in time and prevented by giving prompt assistance, fall risks and their effects can be significantly minimized [1]. The research community has proposed a wide range of methods to identify and prevent elderly persons from falling. These methods rely on combining machine learning, the Internet of Things (IoT), and image recognition, among other technologies. Fall prediction is the process of continuously monitoring an older person with wearable or non-wearable equipment to determine whether they are likely to fall and how likely that is to happen [2]. Fall prediction mainly focuses on identifying fall risk factors, though it requires a precise prediction mechanism that can react quickly. Although difficult to attain, a good prognosis will significantly aid in protecting older people from the consequences of falls. Several events, such as sitting down on a chair while still standing or stooping over to pick up something, might create the appearance of a fall. The procedure is anticipated to distinguish between genuine and fake falls and immediately transmit an alarm to pre-specified individuals or places. The goal is to assist older individuals as soon as possible following a fall so that any consequences can be kept to a minimum [3].

While technologies based on image processing mean that it is not necessary to place any device on the user's body, this remains attached to the monitored location, preventing the monitored person's freedom of movement. On the other hand, methodologies based on the measurement of magnitudes intrinsic to the user's body allow total freedom of movement and independence from the user's living environment. These imply specific data collection equipment placement, battery power, and communication capacity. Thus, in this work, the measurement of acceleration signals of the user's body to be monitored is considered the data package available for analysis.

B. Related Work

Artificial intelligence and the use of neural networks have gained a lot of significance in various fields of application [4]–[6]. Also, when it comes to the fall detection problem, machine learning algorithms have paved the way with promising results [7]. Although the main types of approaches for fall detection consist of the usage of wearables devices, camera-based and ambiance devices may also be considered, although with less interest due to their lack of portability. It should be noted that the disadvantage of portable fall detection systems is related to their energy supply and their need of a communication channel. Nevertheless, employing strategies such as data compression [8] or efficient power management when considering IoT devices can significantly reduce the power consumption footprint and allow its wider use [9], [10].

Different algorithms have been documented in the literature when it comes to the use of machine-learning algorithms for detecting falls. The authors of [11] shown that Artificial Neural Networks (ANNs) have a high tolerance for heterogeneous data with accurate results, making them suited for detecting falls, particularly in the fog-computing layer employed on smart devices to save the needed high computing resources. Also, support vector machines (SVMs) were reported in [12] to be suitable for small and large datasets. They might be considered for the edge-computing layer in the context of fall detection, especially on fall postures. Decision Trees can partition data into two classes: fall or non-fall, and their learning process is straightforward and easy, making them appropriate for examining the causes of falls, particularly physiological data, to identify the patterns and potential risk of falls [13]. Unlike typical machine learning methods, deep learning does

not require external feature extractors. However, training a large dataset takes a lengthy time, whereas prediction takes a short period. Like the one we observe in ANNs, the processing is seen as a black-box and is also challenging to grasp [14]. Long-Short-term memory networks (LSTM) are part of a particular class of deep algorithms that can be used for this purpose [15], [16]. The authors of [3] proposed a technique for detecting falls using 3D skeletal data from a Microsoft Kinect. The approach used the accelerated velocity of the center of mass of various body components and skeletal data as major bio-mechanical parameters and employed an LSTM to identify a fall. Unlike other comparable systems, it does not necessitate the installation of a sensor on any part of the older person's body, allowing them to maintain their privacy. The approach was tested and verified on an existing dataset and found to be successful in fall detection. The gadget may detect falls in older individuals at home because no specific sensor installation is required. As far as the author's knowledge, and considering deep LSTM networks as the current state-of-the-art for solving the fall detection problem, there are no studies concerning the size of the LSTM, that is, its cell size or the size of the cell state vector.

C. Contribution

Despite the fact that some works have demonstrated that recurrent neural networks, namely LSTMs, are adequate for detecting patterns on time series data, there has yet to be a consensus regarding the number of recursive layers that should be placed in cascade, as well as their size. Motivated by the aforementioned matter, a sensitivity analysis on LSTM networks for fall detection wearable sensors will be presented in this paper. As such, the main contribution is related to some complexity analyses, as the number of layers of LSTM increases, as well as the role that this type of hyperparameter has on the algorithm's performance. Secondly, it will be shown that reducing the cell size leads to obtaining models with lower memory footprint, a feature most desired on wearable electronics embedded designs.

The remaining paper is organized as follows: Section II describes the LSTM unit cell and how it works. Section III presents the proposed method as the architecture hyperparameters followed by the metric evaluation. Section IV describes the results and Section V the conclusions of this paper and its future work.

II. Long Short-Term Memory Networks

Long-term and short-term memory networks (LSTM) are part of a group of architectures called recurrent neural networks (RNN), which are a family of neural networks widely used for models whose problems are characterized by data sequences [17]. These two types of networks differ from other neural networks since the network parameters are shared along several points of the sequence under analysis. The outputs of the previous elements influence each one of the output elements. This particularity of persistence of the characteristics present in the data allows the model to look at a sequence as a whole and not just at an individual element. Unlike other networks, RNNs use feedback from activation's from previous time slots as input from the network to make a decision for the current input. It should be noted that the sequences that the RNN analyzes correspond to a time interval δ, which may not refer precisely to the phenomenon of time duration in the real world.

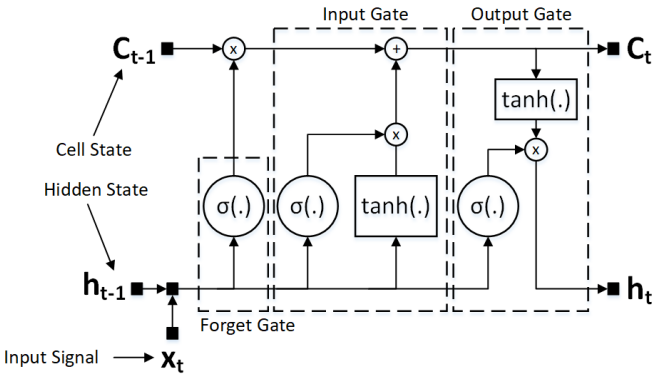


Fig. 1: Internal structure of the LSTM cell.

Given the example of an LSTM cell and the complete internal structure illustrated in Figure 1, where the input gate, the output, and the forget gate are represented by $\boldsymbol{i^{(t)}}$, $\boldsymbol{o^{(t)}}$ and $\boldsymbol{f^{(t)}}$ respectively, and $\boldsymbol{x}$, $\boldsymbol{C}$ and $\boldsymbol{h}$, correspond to the input, state and output of the current cell and the previous cell, that is, the time stamp $t-1$. The variable $\tilde{\boldsymbol{C}}^{(t)}$) represents the vector of new values that can be added to the state $\boldsymbol{C}$. Then the final state $\boldsymbol{C}$ of the previous LSTM cell may or may not be affected by the output of each of the three gates, creating a new state of the current cell.

A. Forget gate

This first gate decides which part of the final state information from the previous cell will be rejected. The sigmoid function of the forget gate layer makes the decision. This function receives as input the concatenation between the input of the current cell and the output of the previous cell. Its corresponding mathematical equation can be expressed as

$$\boldsymbol{f^{(t)}} = \sigma(\boldsymbol{U_f x^{(t)}} + \boldsymbol{W_f h^{(t-1)}} + \boldsymbol{b_f}) \tag{1}$$

where:

- $U_f \in \mathbb{R}^{h \times d}$ corresponds to the input weights;
- $W_f \in \mathbb{R}^{h \times h}$ are the recurrent weights;
- $b_f \in \mathbb{R}^{h}$ are the bias;
- σ is the sigmoid function, computed as

$$\boldsymbol{\sigma^{(t)}} = \frac{1}{1 - \exp^{-x}} \tag{2}$$

- $tanh$ is the hyperbolic tangent
- the operator $\circ$ denotes the element-wise product.
- The cell output corresponds to the last cell hidden state, that is $y_t = h_t$, when $t = h$.

B. Input gate

In this stage, the proportion of new information that will be stored in the $\boldsymbol{C^{(t)}}$ state of the cell is decided. This stage is divided into two parts: the first $\boldsymbol{i^{(t)}}$ which determines, through the sigmoid function, which cell input values will be updated; and the second $\tilde{\boldsymbol{C}}^{(t)}$ which creates the new vector of candidate values to replace values from the input vector, through the function $tanh$, with output values in the interval $[-1, 1]$. The result of each part is combined to update the $\boldsymbol{C}$ state of the cell. The equations that build up this gate are:

$$\boldsymbol{i^{(t)}} = \sigma(\boldsymbol{U_i x^{(t)}} + \boldsymbol{W_i h^{(t-1)}} + \boldsymbol{b_i}) \tag{3}$$

and

$$\tilde{\boldsymbol{C}}^{(t)} = tanh(\boldsymbol{U_g x^{(t)}} + \boldsymbol{W_g h^{(t-1)}} + \boldsymbol{b_g}). \tag{4}$$

C. Output gate

In the output gate, the information that will be at the exit of the cell is determined. This output will be selected by the updated state $\boldsymbol{C^{(t)}}$ of the cell, which, through the multiplication operation between the concatenation of the input $\boldsymbol{x^{(t)}}$ of the current cell and the output of the previous cell by the sigmoid function, allows choosing what the final output of the cell will be. The cell gate and output equations are as follows:

$$\boldsymbol{o^{(t)}} = \sigma(\boldsymbol{U_o x^{(t)}} + \boldsymbol{W_o h^{(t-1)}} + \boldsymbol{b_o}) \tag{5}$$

and

$$\boldsymbol{h^{(t)}} = \boldsymbol{o^{(t)}} \circ tanh(\boldsymbol{C^{(t)}}) \tag{6}$$

The final state $\boldsymbol{C^{(t)}}$ of the cell is updated, converting the state of the previous cell through operations with the results of the gates through the following equation:

$$\boldsymbol{C^{(t)}} = \boldsymbol{f^{(t)}} \circ \boldsymbol{C^{(t-1)}} + \boldsymbol{i^{(t)}} \circ \tilde{\boldsymbol{C}}^{(t-1)} \tag{7}$$

After understanding the concept of the LSTM cell, it is now possible to describe the proposed method of this paper in the following section. It should be noticed that from all the equations (1) to (7), a set of hyperparameters arises that must be determined considering a training stage. Its dimension, that is, the number of hyperparameters that describes the LSTM network, depends on the cell size h.

III. Proposed Method

The proposed method will analyze the network's topology to study the models' sensitivity for a time series classification problem. Therefore, two network models will be deeply studied in this paper:

1) Model I, as described in Figure 2 a), will consist of one LSTM layer as feature extraction, one fully connected layer to classify the output (Fall or Daily Activity), and one SoftMax layer to convert the vector of numbers from the fully connected layer into a vector of probabilities;
2) Model II, as shown in Figure 2 b), is similar to Model I but has two LSTM layers, for the feature extraction, instead of one.

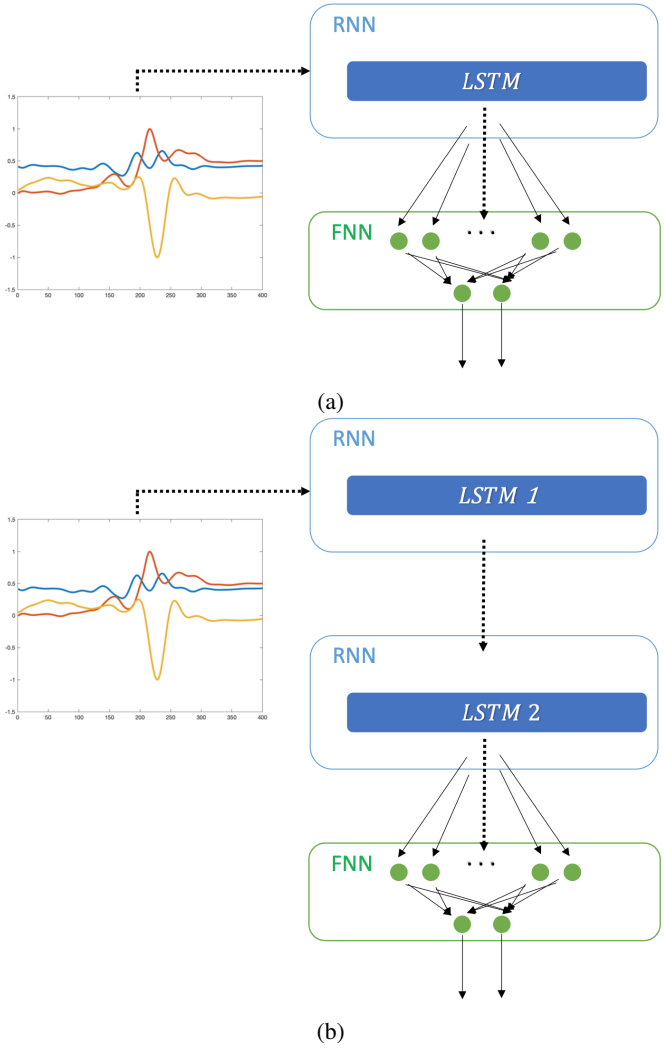
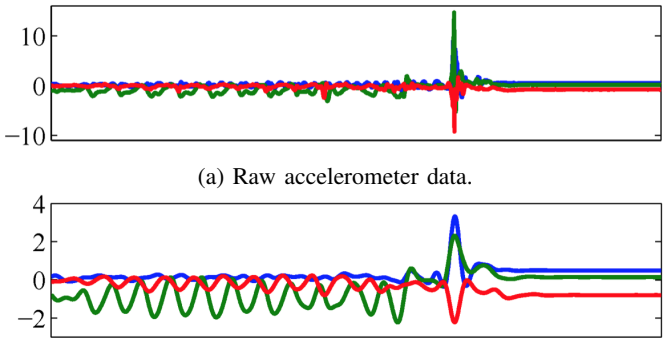


Fig. 2: Proposed system models. Inspired from [18]–[21].

Initially, for both models, the first step consists of annotating the data that will be used for training purposes. The data to be applied here is known as the SisFall dataset [22], [23]. This dataset incorporates a wide range of experimental samples concerning the size and age of the volunteers (38 participants, including 19 males and 19 females between 19–75 years), a considerable duration of samples (4.505 samples between $10s$ and $180s$, comprising 2.707 activities of daily living (ADLs) and 1.798 falls) and a large diversity of emulated movements (19 classes of ADLs including basic and sporting activities, and 15 classes of falls).

The authors in [22] suggest scaling first the accelerometer values using the scale factor of $\pm 16G$ the accelerometer resolution value of 13-bit. Then, a 4^{th} order low-pass digital Butterworth filter with a cutoff frequency of $5Hz$ is applied to the data samples to remove high-frequency noise. The results

are shown in Figure 3.



(a) Raw accelerometer data.

(b) Filtered accelerometer data.

Fig. 3: Fall example: trip and fall while jogging. Adapted from [23].

Next, a $2s$ window (length of 400 points in SisFall dataset sampled at $f_s = 200$Hz) is taken from the total number of samples centered on the fall event when using Falls samples and centered on the interval when using ADLs samples.

The final step is to train the two proposed models. A range of cell units, between 0 and 400, was used to define the LSTM size to study their sensitivity. Besides that, the Batch Size was equal to 31 with 15 epochs, and the validation frequency was 46. The trained models used a binary cross-entropy loss and an Adam optimizer [24] with a 1×10^{-4} learning rate and a 1×10^{-6} decay rate. The logistic regression loss can be represented as Equation 8 for the classification problem under study.

$$J(\mathbf{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right], \quad (8)$$

where w denotes the model weights parameters, N the number of samples, y_i denotes the target label, and $\hat{y}_i$ the projected label. Equation 9 expresses the accuracy metric as

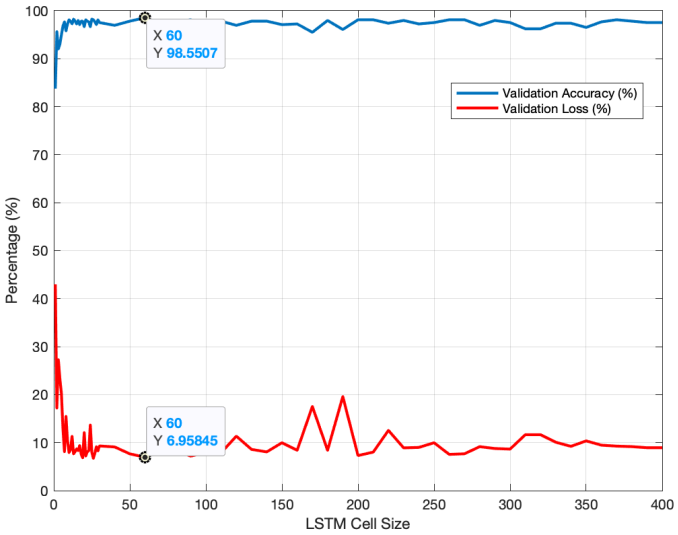
$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{N}\sum_{i=0}^{N} 1(\hat{y}_i = y_i). \quad (9)$$

Through the table of Algorithm 1, a pseudo code is presented to support the whole process training of both models.

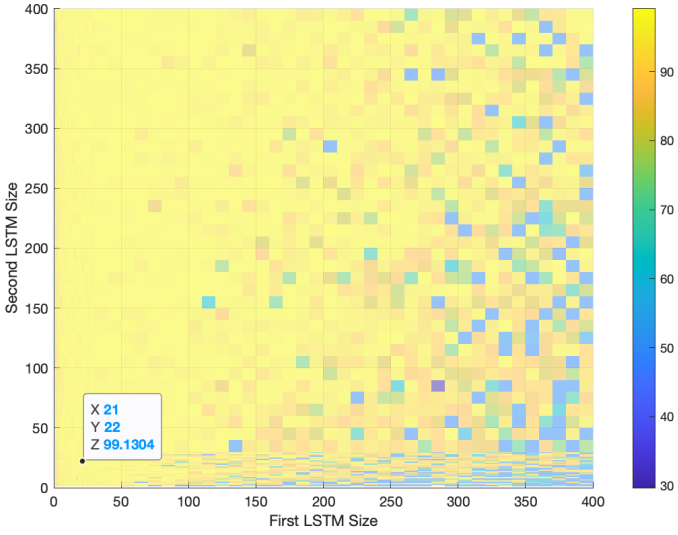
Both models were implemented using MatLab R2022b. They were also trained on a laptop with NVIDIA GeForce GTX 1060, 16GB RAM, and Intel Core i7-8750H CPU@2.20GHz x12. As can be seen, the proposed system is quite simple and also inexpensive for real-world environment implementations. The MatLab scripts are publicly available here [25].

IV. Results

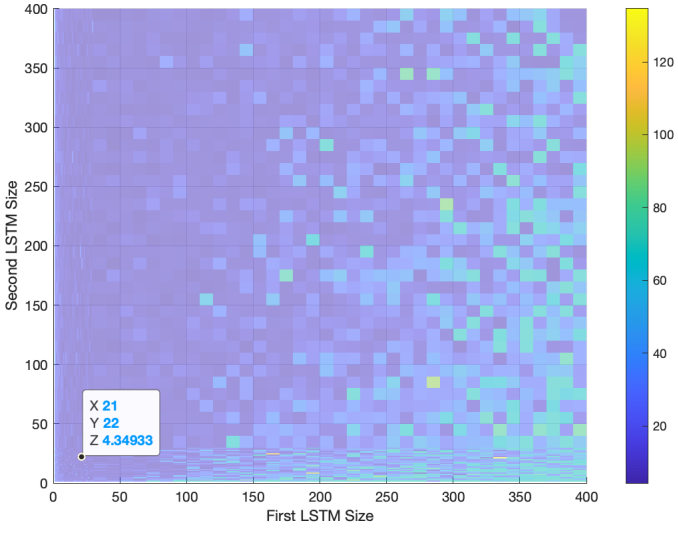
This section presents the performance validation of the proposed models through numerical simulation. To analyze the sensitivity that the dimension of LSTM would have on the



(a) Model I validation accuracy and loss results.



(b) Model II with validation accuracy results.



(c) Model II with validation loss results.

Fig. 4: Accuracy and loss validation results for Model I and Model II.

Algorithm 1 The proposed training model phase.

Require: $\boldsymbol{e} = 1, ..., N$, $\boldsymbol{bs} = 20, 100, 200, 300, 400$, $\boldsymbol{LSTM_1} = 1, ..., 400$, $\boldsymbol{LSTM_2} = 1, ..., 400$

for $\boldsymbol{e} = 1, ..., N$ **do**
  for $batch \leftarrow bs$ **do**
    for $l_1 \leftarrow \boldsymbol{LSTM_1}$ **do**
      for $l_2 \leftarrow \boldsymbol{LSTM_2}$ **do**
        //Build the Deep Learning Model
        $\boldsymbol{model}[0] \leftarrow \boldsymbol{input}$
        $\boldsymbol{model}[1] \leftarrow \boldsymbol{l_1}$
        $\boldsymbol{model}[2] \leftarrow \boldsymbol{l_2}$
        $\boldsymbol{model}[3] \leftarrow \boldsymbol{FNN}$
        //Compile Model
        $model \leftarrow$ Loss = Categorical Cross Entropy
        $model \leftarrow$ Metric = Accuracy
        $model \leftarrow$ Optimizer = Adam
        //Train Model
        $model \leftarrow$ Dataset
        //Score model evaluation and save it
        $results \leftarrow model.save$
      end for
    end for
  end for
end for

achieved performance, as mentioned in secion III, a range of unit cell size was used in the LSTM layers for both models.

Figure 4 a) shows the validation results for Model I. It is noteworthy that, despite the window size used in the annotation is 400 (equivalent to 2 seconds for $f_s = 200$Hz), the optimal LSTM cell size for Model I does not need to be close to 400. In fact, the cell size value that achieved the best performance in terms of accuracy and loss results was a size of 60, as shown in Figure 4 a), with a validation accuracy and loss result of 98.55% and 6.958%, respectively.

Figures 4 b) and c) show the validation results for Model II. Similar to Model I, Model II does not need to have the sizes of the LSTMs close to the annotation window size. The sizes of the first and second LSTM layers of Model II that provide the system with better performance in terms of accuracy and loss are 99.13% and 4.349%, respectively. It is also apparent from Figures 4 b) and c) that the accuracy and loss results are more sensitive and affected when the size of the first LSTM layer is close to zero, regardless of the size of the second LSTM layer.

In addition to the individual results of models I and II, it is also possible to directly compare these models.

To summarize, Table I presents the best results achieved for the ideal configuration for both models.

According to the information provided in Table I, Model II achieves the most accurate results. In fact, by using this model, it is possible to increase the accuracy by 0.58%, reducing the validation on loss value by approximately 1.65%. This means that splitting the complexity into two distinct layers

TABLE I: Results.

Items	Model I	Model II	Difference between models
First LSTM layer Size	60	21	-
Second LSTM layer Size	-	22	-
Accuracy	98.55%	99.13%	0.58%
Validate on Loss	6.96%	4.35%	-2.61%
Processing Time (s)	0.425	0.418	-1.65%

can achieve better results than just using one LSTM layer (through this division, it is possible to achieve better results, which results in better performance). It is also essential to know the processing time of each model to predict an output. Moreover, Model II presents less processing time (-1.65%) when compared to Model I.

In addition to studying the accuracy and loss results between these two models, it is also important to analyze their stability during the training phase for the LSTMs sizes used in Table I.

Figures 5 a) and b) show that Model II exhibits more stability than Model I during the training phase. Herefore, it can be concluded that dividing the LSTM layer into two layers is superior to using only one LSTM layer in this situation.

Lastly, since occupancy memory is an important topic for wearable devices, it is imperative to know the cost memory of each model. Thus, knowing the structure of a LSTM and FC as mentioned in section II, the ocupancy memory can be estimated using the equation 10.

$$\begin{aligned} MS = \sum_{i=1}^{LS} \Big(& Sz_{Bias,i} \cdot 4 \cdot h_i + Sz_{InW,i} \cdot 4 \cdot h_i \cdot d + \dots \\ & + Sz_{RecW,i} \cdot 4 \cdot h_i^2 \Big) + \dots \\ & + Sz_{FC_W} \cdot 2 \cdot h_{LS} + SZ_{FC_{Bias}} \cdot 2 \end{aligned} \tag{10}$$

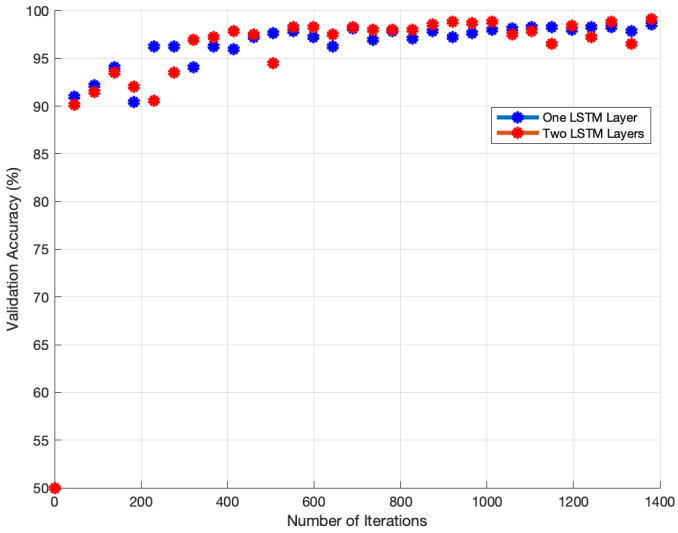
where $Sz_{LS_{Bias},i}$, $Sz_{LS_{InW},i}$, $Sz_{LS_{RecW},i}$, Sz_{FC_W}, and $SZ_{FC_{Bias}}$ corresponds to the number of bytes used for the quantity representation of layer i, for the LSTM bias, input weights, and recurrent weights. The sum limit LS is the number of LSTM (in this work, the authors only used one FC layer to classify the output). h_i is the number of units of the LSTM layer i, and d the number of inputs.

In this paper, 16 bytes were used for the quantity representation of LSTM and FC layers, and d is equal to 3 (the 3 accelerometer axes). Therefore, the memory occupancy of each model is described in Table II (the models used are the ones that achieved the best accuracy and loss results with one and two LSTM layers).

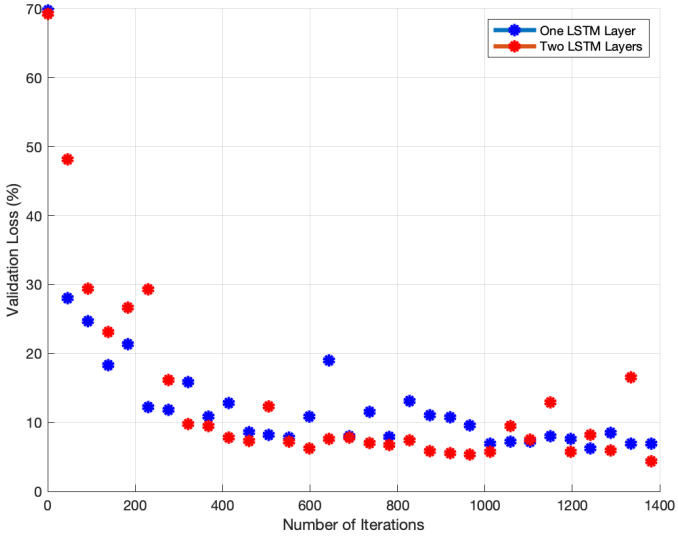
TABLE II: Occupancy memory results for Model I and Model II.

Items	Model I	Model II
Occupancy memory (bytes)	247712	70944

Table II shows that, although Model II has one more LSTM layer than Model I, Model II takes up less memory than



(a) Model I and II trainning accuracy results.



(b) Model I and II trainning loss results.

Fig. 5: Accuracy and loss training results for Model I and Model II.

Model I because the LSTM size in each layer is 3 times smaller than the LSTM layer of Model I. Thus, it is possible to conclude in this work that using more LSTM layers allows us to achieve better performance (with better accuracy and loss results) with less memory occupancy.

V. Conclusion and Future Work

Using multiple LSTM layers in a neural network can improve its ability to recognize and respond to complex patterns in sequential data. This paper proves that each LSTM layer of the network can have a different level of sensibility to certain features in the data, which can help the network learn to recognize a wide range of patterns, improving the network's performance.

At the same time, through this study, it was also possible to conclude that the application of two LSTM layers, instead of just one LSTM layer, allows us to achieve better performance with less processing time and less memory occupancy, reaching from 98.55% to 99.13% of accuracy with 6.96% and 4.35% of loss values respectively.

It is worth mentioning that the variable type and their respective accuracy are other issues that might be considered for future work studies. Since the required memory should be minimized, changing the variable type and width is crucial to reduce memory usage and increase accuracy.

Acknowledgment

This research was partially funded by by the European Union's Horizon Europe Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement No. 101086387, and Fundação para a Ciência e a Tecnologia under Projects UIDB/04111/2020, UIDB/50008/2020, ROBUST EXPL/EEI-EEE/0776/2021, and 2021.04180.CEECIND, as well as Instituto Lusófono de Investigação e Desenvolvimento (ILIND) under Project COFAC/ILIND/COPELABS/1/2022.

References

[1] R. Tanwar, N. Nandal, M. Zamani, and A. A. Manaf, "Pathway of trends and technologies in fall detection: A systematic review," *Healthcare*, vol. 10, no. 1, 2022.

[2] M. Hemmatpour, R. Ferrero, B. Montrucchio, and M. Rebaudengo, "A review on fall prediction and prevention system for personal devices: Evaluation and experimental results," *Advances in Human-Computer Interaction*, vol. 2019, p. 1–12, 2019.

[3] T. Xu, Y. Zhou, and J. Zhu, "New advances and challenges of fall detection systems: A survey," *Applied Sciences*, vol. 8, no. 3, p. 418, 2018.

[4] S. Sulemane, J. P. Matos-Carvalho, D. Pedro, F. Moutinho, and S. D. Correia, "Vineyard gap detection by convolutional neural networks fed by multi-spectral images," *Algorithms*, vol. 15, no. 12, 2022.

[5] S. D. Correia, S. Tomic, and M. Beko, "A feed-forward neural network approach for energy-based acoustic source localization," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, 2021.

[6] R. Santos, J. P. Matos-Carvalho, S. Tomic, M. Beko, and S. D. Correia, "Applying deep neural networks to improve uav navigation in satellite-less environments," in *2022 International Young Engineers Forum (YEF-ECE)*, 2022, pp. 63–68.

[7] A. Ramachandran and A. Karuppiah, "A survey on recent advances in wearable fall detection systems," *BioMed research international*, vol. 2020, 2020.

[8] S. D. Correia, R. Perez, J. Matos-Carvalho, and V. R. Q. Leithardt, "µjson, a lightweight compression scheme for embedded gnss data transmission on iot nodes," in *2022 5th Conference on Cloud and Internet of Things (CIoT)*, 2022, pp. 232–238.

[9] E. Pinheiro and S. Correia, "Hardware architecture of a low-cost scalable energy monitor system," *Int. J. Eng. Trends Technol*, vol. 61, no. 1, pp. 1–5, 2018.

[10] N. Martins, S. D. Correia, P. Romano, and D. N. K. Jayakody, "Indoor air quality monitoring and iot platform for smart building management," *strategies*, vol. 9, p. 11, 2022.

[11] H. Kerdegari, S. Mokaram, K. Samsudin, and A. R. Ramli, "A pervasive neural network based fall detection system on smart phone," *Journal of Ambient Intelligence and Smart Environments*, vol. 7, no. 2, p. 221–230, 2015.

[12] M. Yu, Y. Yu, A. Rhuma, S. M. R. Naqvi, L. Wang, and J. A. Chambers, "An online one class support vector machine-based person-specific fall detection system for monitoring an elderly individual in a room environment," *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 6, p. 1002–1014, 2013.

[13] S. S. Kambhampati, V. Singh, M. S. Manikandan, and B. Ramkumar, "Unified framework for triaxial accelerometer-based fall event detection and classification using cumulants and hierarchical decision tree classifier," *Healthcare Technology Letters*, vol. 2, no. 4, p. 101–107, 2015.

[14] M. M. Hassan, M. Z. Uddin, A. Mohamed, and A. Almogren, "A robust human activity recognition system using smartphone sensors and deep learning," *Future Generation Computer Systems*, vol. 81, p. 307–313, 2018.

[15] D. Pedro, J. P. Matos-Carvalho, J. M. Fonseca, and A. Mora, "Collision avoidance on unmanned aerial vehicles using neural network pipelines and flow clustering techniques," *Remote Sensing*, vol. 13, no. 13, p. 2643, 2021.

[16] J. P. Matos-Carvalho, F. Moutinho, A. B. Salvado, T. Carrasqueira, R. Campos-Rebelo, D. Pedro, L. M. Campos, J. M. Fonseca, and A. Mora, "Static and dynamic algorithms for terrain classification in uav aerial imagery," *Remote Sensing*, vol. 11, no. 21, p. 2501, 2019.

[17] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[18] R. Santos, J. P. Matos-Carvalho, S. Tomic, M. Beko, and S. D. Correia, "Applying deep neural networks to improve uav navigation in satelliteless environments," in *2022 International Young Engineers Forum (YEF-ECE)*, 2022, pp. 63–68.

[19] J. Nakama, R. Parada, J. P. Matos-Carvalho, F. Azevedo, D. Pedro, and L. Campos, "Autonomous environment generator for uav-based simulation," *Applied Sciences*, vol. 11, no. 5, p. 2185, 2021.

[20] D. Pedro, A. Mora, J. Carvalho, F. Azevedo, and J. Fonseca, "Colanet: A uav collision avoidance dataset," in *Technological Innovation for Life Improvement*, L. M. Camarinha-Matos, N. Farhadi, F. Lopes, and H. Pereira, Eds. Springer International Publishing, 2020, pp. 53–62.

[21] A. B. Salvado, R. Mendonça, A. Lourenço, F. Marques, J. P. Matos-Carvalho, L. Miguel Campos, and J. Barata, "Semantic navigation mapping from aerial multispectral imagery," in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, 2019, pp. 1192–1197.

[22] A. Sucerquia, J. López, and J. Vargas-Bonilla, "Sisfall: A fall and movement dataset," *Sensors*, vol. 17, no. 12, p. 198, 2017.

[23] M. Johnson, "Wjmatthew/sisfallanalysis: Analysis of the sisfall fall detection dataset with readings from accelerometer and gyroscope." [Online]. Available: https://github.com/WJMatthew/SisFallAnalysis

[24] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[25] S. D. Correia and J. P. Matos-Carvalho, "MatLab Source Code for Sensitivity Analysis of LSTM Networks for Fall Detection Wearable Sensors," 2023. [Online]. Available: https://github.com/SCorreiaPT/LSTM_SensitivityFallDetection