

Sampling Frequency Evaluation on Recurrent Neural Networks Architectures for IoT Real-time Fall Detection Devices

F. Luna-Perejon^a, J. Civit-Masot^b, L. Muñoz-Saavedra, L. Duran-Lopez, I. Amaya-Rodriguez, J. P. Dominguez-Morales^c, S. Vicente-Diaz^d, A. Linares-Barranco^e, A. Civit-Balcells^f and M. J. Dominguez-Morales^g

Robotics and Computer Technology Lab., University of Seville, Seville 41012, Spain

Keywords: Fall Detection, Recurrent Neural Network, Deep Learning, Internet of Things.

Abstract: Falls are one of the most frequent causes of injuries in elderly people. Wearable Fall Detection Systems provided a ubiquitous tool for monitoring and alert when these events happen. Recurrent Neural Networks (RNN) are algorithms that demonstrates a great accuracy in some problems analyzing sequential inputs, such as temporal signal values. However, their computational complexity are an obstacle for the implementation in IoT devices. This work shows a performance analysis of a set of RNN architectures when trained with data obtained using different sampling frequencies. These architectures were trained to detect both fall and fall hazards by using accelerometers and were tested with 10-fold cross validation, using the F1-score metric. The results obtained show that sampling with a frequency of 25Hz does not affect the effectiveness, based on the F1-score, which implies a substantial increase in the performance in terms of computational cost. The architectures with two RNN layers and without a first dense layer had slightly better results than the smallest architectures. In future works, the best architectures obtained will be integrated in an IoT solution to determine the effectiveness empirically.

1 INTRODUCTION


Approximately 28%-35% of elderly people, over 65 years old, suffer at least one unintentional fall per year (Organization et al., 2008). Major injuries can provoke temporal or permanent disabilities, even death. Regarding to this risk, the early assistance is considered a relevant factor (Noury et al., 2007). Due to the increase in the population of the affected cohort (Werner, 2011), this issue is increasingly relevant.


For an early assistance, Fall Detection Systems (FDS) can be a crucial tool as they allow us to monitor the user and quickly alert health centers which they are connected with, when a fall or some risk events are detected. Among all the different FDS types, wearable devices allow a continuous monitor-


ing without dependence to the environment. Due to the need to be portable and with great autonomy, these systems not only should be effective distinguishing between falls and activity of daily living, but also have low computational cost in order to be implemented in IoT devices with low-power consumption requirements. Recurrent Neural Networks (RNN) have shown to be very effective algorithms to analyze sequences of values (Lipton et al., 2015; Aceto et al., 2019; de Jesús Rubio, 2009). Due to previous studies (Musci et al., 2018; Luna-Perejon et al., 2019), these RNN architectures has demonstrated to detect falls with acceptable results. However, such Deep Learning algorithms are computationally complex, which affects energy consumption and execution time, disadvantages that can make its implementation in IoT devices not viable (Torti et al., 2018).


In this context, the present work shows the performance results obtained when the best architectures from (Luna-Perejon et al., 2019) are trained with a smaller width of input data, by reducing the sampling frequency down to 25Hz.


The rest of the paper is structured as follows: Section 2 presents the basics of RNN layers used, the


^a  <https://orcid.org/0000-0002-4352-8759>


^b  <https://orcid.org/0000-0003-3306-3537>

^c  <https://orcid.org/0000-0002-5474-107X>

^d  <https://orcid.org/0000-0001-9466-485X>

^e  <https://orcid.org/0000-0002-6056-740X>

^f  <https://orcid.org/0000-0001-8733-1811>

^g  <https://orcid.org/0000-0001-5669-9111>

dataset structure, the RNN architectures tested in this study and the evaluation metrics. Then, Section 3 presents the results and discussion over the obtained values. Finally, Section 4 shows the conclusions.

2 MATERIAL AND METHODS

The tests carried out in this study aim to analyze the performance of different RNN architectures when working with a lower output width, that is, less amount of samples per classification.

2.1 Gated Recurrent Neural Networks

The original RNN model emerged as an ANN-based way to attack classification problems related with sequential data with a strong dependence on the order of values. This made them suitable for temporal signals. However, its application field was very limited due to the influence of the vanishing gradient problem (Hochreiter, 1998). This effect means that the gradient that is propagated back through the network either decays or grows exponentially. As a consequence, traditional RNNs are hard to train using backpropagation through time (Williams and Zipser, 1995). Gated Recurrent Neural Networks have been one of the most effective solutions to this problem to date. They introduce some memory-like cells in the architecture that hold information separated from the rest of the neural network. The information is managed through a set of gates. During the training of the network, the cells learn to close or open their gates according to the relevance of the information that comes from the sequence and the information currently stored. This information is used in the learning process of the RNN.

Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) were the first proposed Gated RNN. They contain three gates, two of which evaluate the update of the information stored, and the last gate controls what information is provided to the RNN in each step. Gated Recurrent Units (GRU) (Cho et al., 2014) are more recent cells similar to LSTM, that lack of the last gate mentioned so that the whole information stored is used during the entire execution. Both alternatives have shown to be similarly effective (Chung et al., 2014), although GRUs are slightly more economical in terms of computation cost.

2.2 Dataset

We used the SisFall dataset (Sucerquia et al., 2017), consisting of a set of activities performed by users and

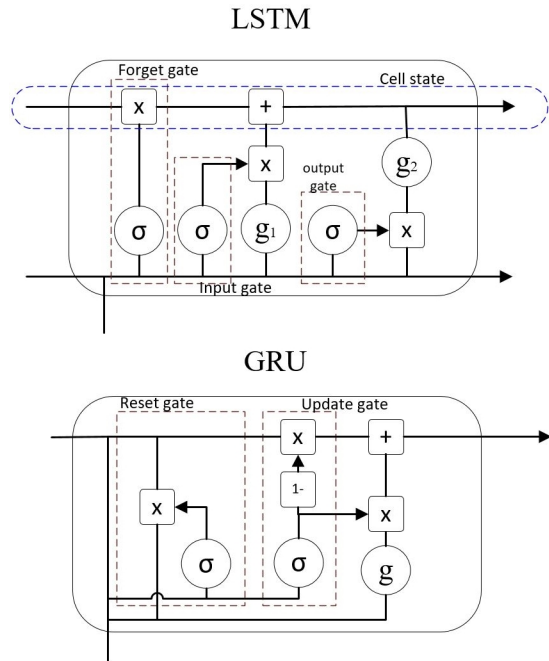


Figure 1: LSTM and GRU cells. First contains three gates, while the second have only two.

registered using a device with accelerometers, fixed to their waist. This set includes activities of daily living (ADL) and falls, acquired at 200 Hertz. To adapt the dataset to this kind of algorithms, each recording was segmented with a fixed number of samples, called "width" from now on. They were classified using the criteria proposed by (Musci et al., 2018) and considering three classes: Fall, Alert and Background (BKG), that is, a fall event, a fall hazard status and the rest (an ADL or inactivity after the fall). Each recording was segmented with a width of 256 and a stride of 128.

2.3 Methodology

For a more robust result, a 10-fold cross validation was carried out. The subsets were created randomly by users, so that the data of each user was contained in a unique fold. Additionally, the distribution of adult and elderly users in each subset was done in such a way that a balance was obtained looking for an equitable distribution.

2.3.1 RNN Architectures

The RNN architectures considered in this work were the ones proposed in a previous study (Luna-Perejon et al., 2019) but only those that have batch normalization 3, due to the great improvement in the effectiveness with almost no repercussion on the computational complexity. The smallest architecture consist

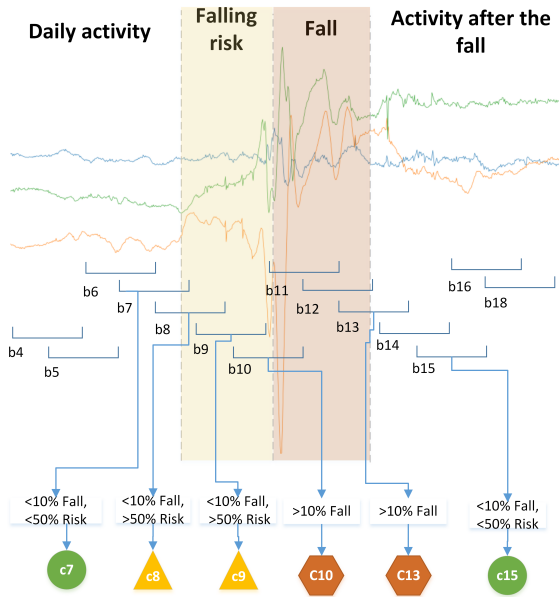


Figure 2: Recording segmentation and labeling process.

of batch normalization, a RNN layer and dense layer as output. The other architectures contain additionally a first dense layer, a second RNN layer previous to the final layer, or both. We tested all the architectures with the two types of RNN layers introduced in 2.1, that is, LSTM and GRU. It should be noted that the RNN used are non-bidirectional.

To analyze the performance of these architectures with a lower width, the sampling frequency of the dataset was reduced. This was done artificially by subtracting the even elements of each sample. In this way, we obtain the dataset equivalent to 100Hz from that at 200Hz, we get that one at 50Hz from 100Hz, and same to get it at 25Hz. The widths of the inputs were 256, 128, 64 and 32 samples, respectively. Each architecture was trained and validated with all the output widths.

The hyperparameters used to train all the models were a learning rate of 0.001 and a batch size of 32. Dropout were not applied. We used Adam as optimizer, hyperbolic tangent as activation function and sigmoid as recurrent activation in RNN layers. Finally, we used the weighted loss function proposed by (Musci et al., 2018).

2.3.2 Metrics

The dataset used is highly unbalanced, therefore the overall classification accuracy is not an appropriate way to measure the effectiveness of the system. We compared the effectiveness employing the macro and micro F1-scores (Sokolova and Lapalme, 2009) for each class and average, that measures the rela-

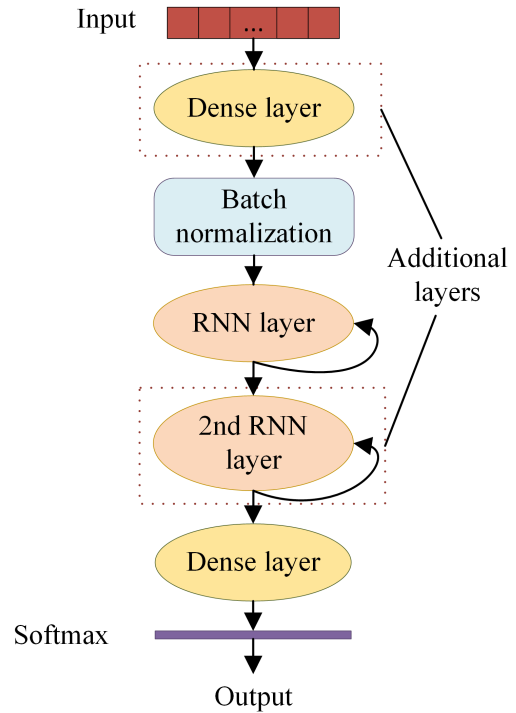


Figure 3: Scheme representing all architectures trained in the study, differentiated between them by having none, one or both of the highlighted layers.

tions between data's positive labels and those given by a classifier through a combination of (micro or macro, respectively) precision and recall. The computational complexity were estimated with the number of parameters that has each architecture and the averaged execution time during the training. We used a graphic processor unit NVIDIA GTX 1080 Ti and the CuDNN versions of this RNN layers provided implemented in the framework Keras, so the results were substantially lower that expected in IoT devices. Therefore, we show a comparison of the acceleration with respect to the trained architecture with greater execution time.

3 RESULTS AND DISCUSSION

The resulting dataset was unbalanced due to the short duration of fall and risk events. Table 1 shows the distribution for each fold in cross validation.

The results showed similar F1-score (macro) results (Figure 4), for both mean and standard deviation. The aspect of greater relevance is that with a lower sampling rate and with a lower number of inputs, the effectiveness of the algorithm does not seem to decrease; in fact, it seems to grow. These results suggests that it is possible to reduce substantially

Table 1: Dataset distribution in each fold for cross validation.

	Total	BKG	ALERT	FALL
fold 1	9485	9010	151	324
fold 2	9485	9057	90	338
fold 3	9485	9074	92	319
fold 4	9485	9025	85	375
fold 5	9330	8887	123	320
fold 6	9441	9011	94	336
fold 7	9892	9424	144	324
fold 8	9485	9044	93	348
fold 9	9486	9067	102	317
fold 10	9093	8574	198	321

the computational cost of running this architectures without reducing the effectiveness, and facilitating the task of monitoring in real time using microcontrollers for IoT systems.

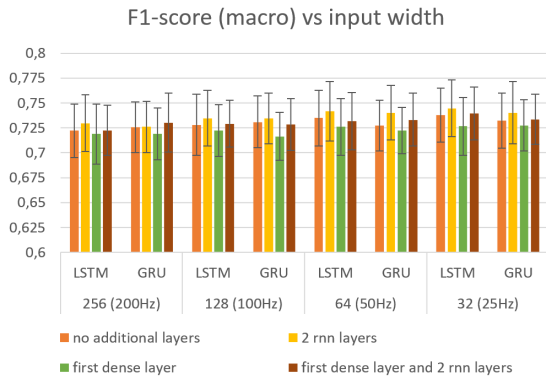


Figure 4: Macro f1-score results for each architecture and different frequency sampling. These values consist of the mean with the 10 folds for cross validation together the standard deviation.

Certain trained models had slightly better results in F1-score. Overall, the architectures without a first dense layer showed a best effectiveness. This is an indicator that the relationship of the values over time is a much more relevant that between the three axes of the accelerometer. The architectures with a second recurrent layer showed a slightly higher effectiveness. However, the number of parameters that these versions contain makes the computational complexity much greater in relation to architectures without this layer. This can make its implementation in IoT devices not viable for a real time monitorization. The architecture with highest execution time contained a first dense layer and two LSTM layers, and output width of 256. It was 0.78 milliseconds. The figure 5 shows the acceleration in relation with the slowest. Reducing the frequency to 25Hz, the execution times are reduced by 4.

Table 2: Number of parameters of each architecture.

	LSTM	GRU
Without additional layers	4847	3663
With a second RNN layer	13295	9999
With a first dense layer	8803	6691
With both additional layers	17251	13027

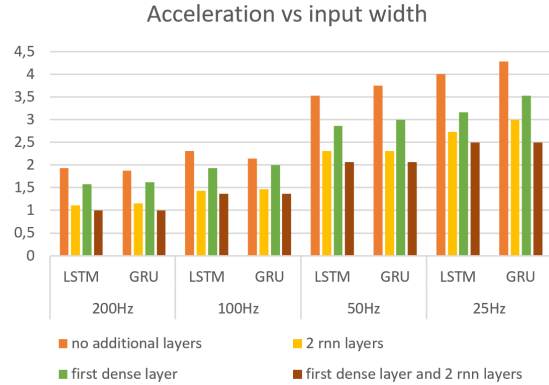


Figure 5: Acceleration in relation with the slowest architecture.

The Figure 6 shows high values for macro recall. This metric is also called sensitivity and, in multi-class problems, is equivalent to the weighted accuracy, that is, the average, over all the classes, of the fraction of correct predictions in this class. That indicates these algorithms have a very good ratio of true positives with respect of true positives plus false negatives for each class. The low value in F1-score is, therefore, due to the low precision of the algorithm, specifically in the class ALERT, below to 0.30. This is probably caused by the scarcity of the data and the variety of values for this class. However, we consider the study of this kind of events interesting in order to detect falling risk and prevent falls. For this, it will be necessary to record and label more data.

The values of these metrics seem not to be affected by the reduction in the number of samples over time. These results are promising, since it implies that a high sampling frequency is not necessary to maintain the effectiveness of these models. Therefore, the device where the trained model was integrated would require fewer microcontroller performance, in terms of sampling and processing speed. In addition, energy consumption would also be reduced, increasing the hours of portable use.

Micro F1-score (Figure 7) are much greater in comparison to macro score. This fact is due to micro precision and accuracy are calculated by combining the absolute successes and failures of all the classes so that the large number of hits in the BKG class hide the results for the other two classes.

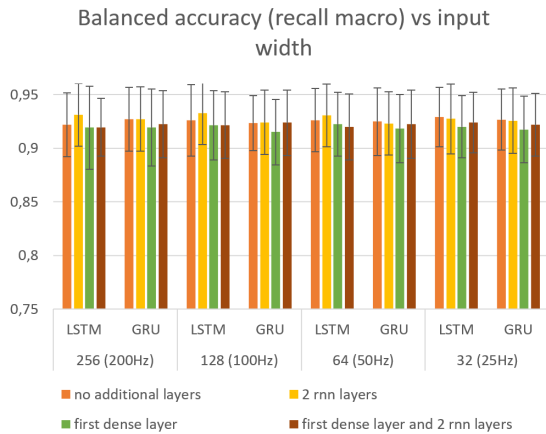


Figure 6: F1-score results for each architecture and different frequency sampling. These values consist of the mean with the 10 folds for cross validation together the standard deviation. In multi-class problems, this metric is equivalent to the weighted accuracy.

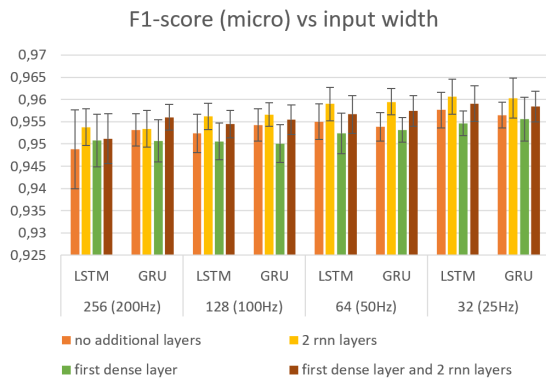


Figure 7: Micro F1-score results for each architecture and different frequency sampling. These values consist of the mean with the 10 folds for cross validation together the standard deviation. Although the results for this metric is very high, it is not representative of the reality of the problem, while the proportion of samples for the background class is much greater than that of the other classes.

4 CONCLUSIONS

The RNN architectures assessed seems to be effective to detect falls even with a small sample rate, without need of increment the acquisition time to obtain more samples as outputs. These results provide a ray of light to the possibility of executing these algorithms in microcontrollers. In future works we will carry out the integration of the best architectures in an IoT solution. The use of RNN architectures to the detection of such events is still recent. Other avenues of research may involve using additional biometric signals, such as heartbeat or galvanic skin response. The location

of the device in other parts of the body, such as the wrist, must also be studied with these algorithms. In the same way, the adjustment of the training and activation parameters of each layer can also be investigated to increase the effectiveness.

ACKNOWLEDGEMENTS

This work is supported by the Spanish government grant (with support from the European Regional Development Fund) COFNET (TEC2016-77785-P). F. Luna and I. Amaya are supported by the Empleo Juvenil with support from EU.

REFERENCES

- Aceto, G., Ciunzo, D., Montieri, A., and Pescapé, A. (2019). Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*.
- Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- de Jesús Rubio, J. (2009). Sofmls: online self-organizing fuzzy modified least-squares network. *IEEE Transactions on Fuzzy Systems*, 17(6):1296–1309.
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.
- Luna-Perejon, F., Civit-Masot, J., Amaya-Rodriguez, I., Duran-Lopez, L., Dominguez-Morales, J. P., Civit-Balcells, A., and Linares-Barranco, A. (2019). An automated fall detection system using recurrent neural networks. In *Conference on Artificial Intelligence in Medicine in Europe*, pages 36–41. Springer.
- Musci, M., De Martini, D., Blago, N., Facchinetti, T., and Piastra, M. (2018). Online fall detection using recurrent neural networks. *arXiv preprint arXiv:1804.04976*.
- Noury, N., Fleury, A., Rumeau, P., Bourke, A., Laighin, G., Rialle, V., and Lundy, J. (2007). Fall detection-principles and methods. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual*

- International Conference of the IEEE*, pages 1663–1666. IEEE.
- Organization, W. H. et al. (2008). Ageing; life course unit. who global report on falls prevention in older age. *World Health Organization*.
- Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.
- Sucerquia, A., López, J. D., and Vargas-Bonilla, J. F. (2017). Sisfall: A fall and movement dataset. *Sensors*, 17(1):198.
- Torti, E., Fontanella, A., Musci, M., Blago, N., Pau, D., Leporati, F., and Piastra, M. (2018). Embedded real-time fall detection with deep learning on wearable devices. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 405–412. IEEE.
- Werner, C. (2011). The older population: 2010. us census bureau.
- Williams, R. J. and Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. *Backpropagation: Theory, architectures, and applications*, 1:433–486.