Recurrent Neural Networks Applied to GNSS Time Series for Denoising and Prediction

Elena Loli Piccolomini
Department of Computer Science and Engeneering, University of Bologna, Italy

Stefano Gandolfi
Department of Engeneering, University of Bologna, Italy

Luca Poluzzi
Department of Engeneering, University of Bologna, Italy

Luca Tavasci
Department of Engeneering, University of Bologna, Italy

Pasquale Cascarano
Department of Mathematics, University of Bologna, Italy

Andrea Pascucci
Department of Mathematics, University of Bologna, Italy

Abstract

Global Navigation Satellite Systems (GNSS) are systems that continuously acquire data and provide position time series. Many monitoring applications are based on GNSS data and their efficiency depends on the capability in the time series analysis to characterize the signal content and/or to predict incoming coordinates. In this work we propose a suitable Network Architecture, based on Long Short Term Memory Recurrent Neural Networks, to solve two main tasks in GNSS time series analysis: denoising and prediction. We carry out an analysis on a synthetic time series, then we inspect two real different case studies and evaluate the results. We develop a non-deep network that removes almost the 50% of scattering from real GNSS time series and achieves a coordinate prediction with 1.1 millimeters of Mean Squared Error.

2012 ACM Subject Classification General and reference → General conference proceedings; Mathematics of computing → Time series analysis; Computing methodologies → Supervised learning by regression; Information systems → Global positioning systems

Keywords and phrases Deep Neural Networks, Recurrent Neural Networks, Time Series Denoising, Time Series Prediction

Digital Object Identifier 10.4230/LIPIcs.TIME.2019.10

1 Introduction

Nowadays global navigation satellite systems (GNSS), such as the GPS, are widely used tools for many monitoring applications. Thanks to the capability of these systems to provide continuously acquired data, which are converted in three-dimensional coordinates after data processing, the monitoring is usually performed through time series analysis. GNSS monitoring can have different purposes depending on the monitored object and the data processing varies consequently therefore this leads to very different time series in terms of signal to noise ratio and noise characteristics. Moreover, the analysis can have different goals: sometimes a signal-denoising is required in order to allow more accurate characterization of the signals, whereas in other cases the goal is a reliable prediction of the coordinates that will be obtained from the incoming data. For GNSS time series analysis many forecasting models have been proposed, based on ARMA, ARIMA and Kalman methods [2],[20],[16],[18], [17] and many denoising models, based on moving averages, sequential filtering and, recently, based on deep learning [15],[12],[5]. Deep Neural Networks (DNNs) have been investigated by many researchers during last years due to the development of computational resources (e.g GPUs) and the raise of Big Data, in order to find mathematical models inspired by the information flowing and processing in human brain. DNNs are widely employed to solve different data science tasks such as pattern recognition, classification, regression, anomaly detection, signal-denoising and density estimation. The class of DNNs algorithms provides a family of networks, suited for sequential data processing, called Recurrent Neural Networks (RNNs). Among them Long Short Term Memory (LSTM) model, introduced at the beginning to solve vanishing gradient problems which affect all the RNNs [1],[9],[4], has showed its strength for almost all of the time dependent problems. The LSTM is characterized by a gated structure which allows it to store past sequence features in its memory block, bringing out them in the output and preserving long term dependences. LSTM approach achieves grateful results in speech recognition [7] and [8], hand-writing recognition [6] and recently in Traffic Flow Prediction [19], Real Time Autonomous Vehicle Navigation [13]. The aim of this work is to develop a deep learning method suitable for prediction and denoising of GNSS time series. Our model add to the LSTM layer used in [12] an activation hyperbolic tangent ($tanh$) layer and a Full Connected layer. In Section 2, first we introduce basic notions about Deep Feed Forward Neural Networks, Recurrent Neural Networks and LSTM Network, then we present the proposed network architecture and its forward equations. In Section 3 we present three different time series used for the experiments and finally in Section 4 we discuss the results.

2 Framework

In this section we carry out a brief review about DNNs and their developments. In the first part we introduce Feed Forward Neural Networks (FFNNs) and we point out their structure. In the second part we remark Recurrent Neural Networks (RNNs), a powerful tool for sequential data processing. Finally we provide our model forward equations.

2.1 Feed Forward Neural Networks

The main goal of a FFNN algorithm is to approximate some function f^* in a suitable functional space. A FFNN describes an approximation map, called net, defined as follows:

$$y = f(x, \theta_1, \dots, \theta_k), \tag{1}$$

where x, y are the input and the output of the net, respectively, and $\theta_1, \dots, \theta_k$ are the parameters that the net has to learn respect to a given training set. The map (1) is the composition of k functions and it can be written as follows:

$$f(x, \theta_1, \dots, \theta_k) = f_k \circ f_{k-1} \circ \cdots \circ f_1, \tag{2}$$

setting $x_1 = x$ and $y_j = f_j(x_j, \theta_j)$ $\forall j = 1 \dots k$, where $x_j = y_{j-1}$ $\forall j = 2 \dots k$. Each of the functions f_k is called the k-th hidden-layer. The depth of a FFNN is described by the parameter k. The universal approximation theorem [11] states that a FFNN with a linear output layer and at least one hidden layer can approximate as good as depth increases any Borel measurable function from a finite dimensional space to another. For this reason deeper FFNNs showed better results in many tasks; the drawback is the computational cost. Hereafter we provide the forward equation of a 1-hidden layer network:

$$x = input \tag{3}$$
$$a = Wx + b \tag{4}$$
$$y = \Phi(a) \tag{5}$$

where W is the weight matrix, b is the bias vector, Φ is the pointwise activation function. The goal is to find the weights and the bias by minimizing a certain loss function over a given training set (e.g the mean squared error). This problem can be handled by using iterative gradient based method such as stochastic gradient descent (SGD) or ADAM method [14], while the Back Propagation Algorithm [3] is used to compute the gradients.

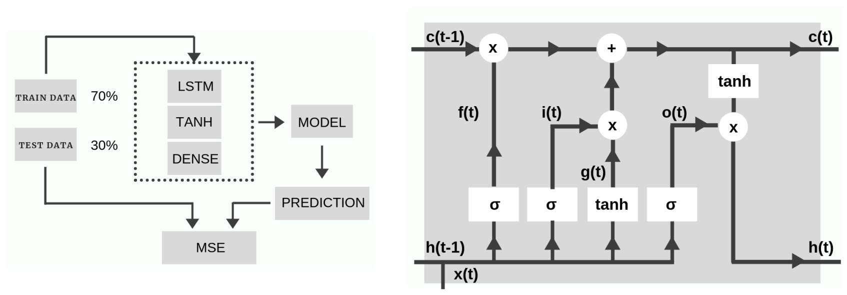
2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a family of Neural Networks suited for time series processing. In RNNs the output at previous time steps affects the output at the current time step and therefore RNNs are able to catch long term dependencies in sequential data. Given a starting hidden layer vector h_0, then for each time step t and for each input vector x_t the forward equations of a general RNN are:

$$h_t = W_{hh}h_{t-1} + U_{xh}x_t + b_h \tag{6}$$
$$y_t = \Phi(h_t) \tag{7}$$

where W_{hh} denotes the hidden-to-hidden weight matrix, U_{xh} the input-to-hidden weight matrix, b_h the bias vector and Φ is a pointwise non-linear activation function. The main difference between FFNN (3)-(5) and RNN (6)-(7) forward equations, is that the latter have a temporal structure. As for FFNNs, gradient based algorithms are used to optimize a loss function respect to the unknown weights and the Back Propagation Through Time (BPTT) algorithm is employed to compute the gradients. One drawback of RNNs is that BPTT algorithm computes gradients that tend to vanish or explode due to the fact that we are composing many times the same non linear function [1],[9]. The most effective model used in practical applications are gated RNNs such as Long Short-Term Memory (LSTM) nets. A LSTM network [10] is made up of LSTM units showed in Figure 1(b). Generally, a LSTM unit is composed of a cell which is able to record the main information over long time intervals [4] and three different gates: the forget gate, the input gate and the output gate. These gates have the task to supervise the flow of information and prevent vanishing gradient problems. The forget gate uses a sigmoid function to decide which information has to be taken into account in the previous cell state. The input gate decides which new information has to be stored in the cell memory. The output gate decides the contents of the output vector. Each gate takes the same input: x_t the current input and h_{t-1} the previous hidden layer vector. The LSTM unit forward equations are presented below:

$$f_t = \sigma(U_f x_t + W_f h_{t-1} + b_f) \tag{8}$$
$$i_t = \sigma(U_i x_t + W_i h_{t-1} + b_i) \tag{9}$$
$$g_t = tanh(U_g x_t + W_g h_{t-1} + b_g) \tag{10}$$
$$c_t = i_t * g_t + f_t * c_{t-1} \tag{11}$$
$$o_t = \sigma(U_o x_t + W_o h_{t-1} + b_o) \tag{12}$$
$$h_t = tanh(c_t) * o_t \tag{13}$$



(a) (b)

Figure 1 (a) Complete process used in this study. (b) LSTM unit.

where $*$ represents the pointwise product, f_t, i_t, o_t are respectively the forget gate, the input gate and the output gate vectors at time t, c_t is the cell vector at time t and the terms U and W in each equation are the weight matrices.

2.3 Proposed Network Architecture

Given an input sequence x_t, for $t = 1 \ldots$ T, and an initial hidden vector h^0, the model proposed in this paper is a RNN model with the following forward equations:

$$h_t^1 = \Phi(x_t, h_{t-1}) \tag{14}$$
$$h_t^2 = tanh(h_t^2) \tag{15}$$
$$y_t = w_{h^2} * h_t^1 + b_{h^2} \tag{16}$$

where $*$ represents the scalar product, Φ is taken as the LSTM unit with the forward equations (8)-(13). We add after the classic LSTM unit: an activation *tanh* layer and then a full connected layer that realizes a dot product between the weights w_{h^2} and the output of the previous layers. We will call our model LSTM-Full in the following. A crucial point of this architecture is the choice of the h_t^1 vector dimension which influences the number of the weights to train. We remark that for each time t, the hidden layer vector h_t^1 has the same dimension. In the following we indicate with H the length of these hidden layer vectors.

3 Materials

Global Navigation Satellite Systems (GNSS) is a technology that uses data sent by artificial satellites to get the position of the receiver. Using the same data but changing the receiver quality and the data processing it is possible to obtain a wide range of accuracy that ranges from 10/15 meters in real-time positioning of a smartphone to a few millimeters (mm) using a geodetic class receiver that acquires data for 24-hour to estimate the daily average position. A GNSS receiver acquires data continuously and after a data processing phase provides a time series of positions useful for geodynamics or geological applications such as tectonic plate motion research, landslides and subsidence studies. We consider three sets of data in order to evaluate the efficiency of the designed LSTM-Full network in GNSS time series analysis. The first time series is a synthetic one, which allows to compare the output of the network to a known "ground truth" and has been created taking into account the well known characteristics of a geodetic time series of daily positions. The second time series derives from a real GNSS permanent station that daily processes the data using a static approach and it is characterized by a comparatively high signal to noise ratio. The third time series comes from a monitoring GNSS station used for early warning monitoring and it is characterized by a low signal compared to the measurements.

3.1 Synthetic Time Series

To create a Synthetic Time Series that simulates properly a real GNSS one, it is important to understand which are the main characteristics of the GNSS positioning. Using GNSS signals a receiver can estimate its position, with different levels of accuracy, measuring the transmitting time of GNSS signals emitted from four or more GNSS satellites and knowing the position of each satellite during the time. The signal crosses the atmosphere with a speed close to the speed of light depending on the physical characteristics of the atmosphere that change continuously. Therefore a GNSS receiver installed on a building roof or on a stable rock can measure during the time some periodical effects due to the seasonal thermal dilatation or solid earth and ocean tides. For all these reasons a GNSS time series can be composed by some periodical terms and, as literature shows, a noisy component which is the sum of a white noise that follows a normal distribution and a random walk noise. The synthetic time series is constructed as follows:

$$x(t) = mt + q + Asin(2\pi ft) \tag{17}$$

and it is characterized by the following parameters: $m = 0, q = 0.01, A = 0.01, f = \frac{1}{365}$ and considering $t \in [1, 4000]$. We assume that x, q, A are expressed in meters (m) whereas t is expressed in days (d) and f is expressed in d^{-1}. In order to represent a realistic case study we add: w gaussian noise, with mean equal to zero and standard deviation $p = 2$ mm and r random walk noise or red noise generated using a normal distribution with mean zero and standard deviation equal to 0.03 mm. Hence we construct the raw time-series as follows:

$$\tilde{x}(t) = x(t) + w + r. \tag{18}$$

It is showed in Figure 2(a) using the green dots. Since in our model we use the activation *tanh* layer it is suggested to scale all the time series values in $[-1, 1]$. This kind of action increases the net performances.

3.2 Real Static Time Series

The first real time series, hereafter called "static" due to the GNSS data processing used, is 11 years long and each time step represents a daily solution. It has three different components: North, East and Up components. The North component is depicted in Figure 3 (a) using green dots. It is for sure the most accurate time series obtainable by GNSS technology because each solution is the mean of the observations recorded with a sampling time of 30 seconds during 24 hours. Since these kind of time series are generally used for tectonic plate motion or landslide monitoring it is important to have accurate and no noise solutions. The analysis that we conduct refers to those applications where the goal is the characterization of the periodical signals and in this case a signal denoising can improve the results.

3.3 Real Kinematic Time Series

The second real GNSS time series, hereafter called "kinematic", derives from higher frequency (1Hz) coordinates solutions, that were estimated using a kinematic approach during the data processing. The time span considered for the test is about 30 days and every sample represents a solution per second. As for the static time series, our data set is made up of North, East and Up components. In Figure 4 (a) we depict the East Component of this time series using the green dots. This is a completely different scenario respect to the previous one because we have a solution every second and for this reason, the redundancy of the system and the accuracy are lower. This approach can be suitable for early warning applications where it is much more important to have as soon as possible information about unexpected changes respect to the normal movement. Since this kinematic time series is used for structure monitoring such as bridge monitoring or critical landslides where a fast movement of the object can be critical for the safety of life, here the ultimate goal of our analysis is to develop a method capable to give an accurate prediction of the incoming coordinates to improve the capability to detect some anomalies.

4 Results and Discussion

In this section we carry out an objective analysis in order to investigate the properties of the proposed LSTM-Full method and demonstrate its effectiveness. All the experiments are performed on a PC Intel(R) Core(TM) i5-6200U CPU @ 2.30 GHz 2.40GHz with 8.00Gb RAM using Python 3.6 libraries, such as Keras and Pandas. In the first part we analyze the synthetic static GNSS time series depicted as in Section 3.1, then we inspect the two real GNSS time series, described in Section 3.2 and 3.3 respectively.

4.1 Data preparation

Since the LSTM-Full method is a supervised learning algorithm we need to define the form of the input and the form of the label vectors. We can represent the whole raw time series as a sequence of values $\tilde{x}_1, \dots, \tilde{x}_{\mathrm{T}}$, where T is the number of analysed samples. Then we construct the input as vectors defined as follows:

$$X_i = [\tilde{x}_i, \dots, \tilde{x}_{i+s}], \quad i \in [1, \mathrm{T} - s], \tag{19}$$

where s is the sliding window parameter representing the length of each vector. We underline that the data set has a time order respect to the variable i and the net has to be fed following this order. We can say that in time series analysis the time is an hidden feature decoded by the data set time order. The label vector is built as a vector y and its components are defined as shown below:

$$y_i = [\tilde{x}_{i+s+1}], \quad i \in [1, \mathrm{T} - s]. \tag{20}$$

In the following analysis the data set is divided in two disjoint subsets: the training set (taken as 70% of the whole data set) and the test set (30% of the whole data set). The first is used to optimize the model parameters, whereas the other is employed to evaluate model accuracy. The Mean Squared Error (MSE) value is used to evaluate the prediction accuracy of the proposed model whereas the standard deviation (STD) is used to evaluate the scatter of the solution. The general mathematical expressions of MSE and STD are:

$$\mathrm{MSE}(u, z) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (u_i - z_i)^2}, \tag{21}$$

$$\mathrm{STD}(v) = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (v_i - \tilde{v})^2}, \tag{22}$$

where N is the length of the vectors u,z, M and $\tilde{v}$ are the length and the mean of the vector v, respectively.

Table 1 All the results are computed setting $H = 30$. Columns 1 and 2 show respectively the MSE(x,$\tilde{y}$) and STD(x,$\tilde{y}$) values, where x is the theoretical signal and $\tilde{y}$ is the predicted signal, for different length of the input, both in millimeters. The raw time series has a MSE equal to 2.327 millimeters and a STD equal to 1.329 millimeters. Column 3 shows the computational training time, in seconds, choosing different input size.

s	MSE(x,$\tilde{y}$) (mm)	STD(x,$\tilde{y}$) (mm)	time (s)
7	1.276	0.718	7.862
14	1.212	0.687	9.953
30	1.245	0.693	15.115
183	1.198	0.690	59.482
365	1.206	0.678	105.21

4.2 Synthetic time series analysis

The purposes of this analysis are to evaluate:

- if the model proposed can predict the theoretical time series $x(t)$ defined in (17) using only the raw signal $\tilde{x}(t)$ defined in (18),
- if the model proposed can filter the raw data,
- the sensitivity of the proposed method to the choice of the parameters H (defined in Section 2.3) and s (defined in Section 4.1).

In the following we indicate with $x, \tilde{x}$ and $\tilde{y}$ the theoretical time series, the raw time series and the solution constructed with the LSTM-Full algorithm, respectively. All of them represent a 730 days solution. In the first column of Table 1 we report the results of MSE$(x, \tilde{y})$, over different choices of the sliding window parameter s. The choice of the sliding window parameter is a crucial point and in practice it should be a trade off between the training time and the prediction performance. Here we suggest, when possible, to choose s as the lag with the highest Autocorrelation Function (ACF) value of raw data (see Figure 2(c)); this seems necessary if we want to compute a less scattered solution. In Figure 2(b) (red-dot line) we depict the solution for $s = 365$, since that the time series considered has an annual seasonality. In the second column of Table 1 we report the STD and we observe that the scattering is reduced while the sliding window dimension increases. The best improvements in terms of MSE (48% less than raw data) is reached choosing $s = 183$ whereas the best improvements in denoising is reached choosing $s = 365$ (49% less than raw data). As we expect the computational training time increases when the input size, that is s, increases. In Figure 2(d) we report the ACF plot of the differences between the theoretical time steps and the predicted time steps. The plot highlights that there is no relevant correlation within prediction errors, as a good prediction is expected to be. All the previous results are computed setting the hidden vectors length equal to 30. In Table 2 we report the summary of the analysis carried out choosing different values for the parameter H and setting $s = 365$. As it emerges from the first and the second column of Table 2 the best results in terms of MSE and of STD are obtained setting $H = 5$ (50 % less for MSE and 51 % less for STD respect to raw data). In the third column we report the computational training time that again increases while the parameter H increases.

4.3 Static time series analysis

So far we have proved the effectiveness of the proposed LSTM-Full method on a test problem, we now want to evaluate our model on the real time series described in Section 3.2. Since the ultimate goal for this time series is to remove the noisy components, we set the hidden

Table 2 All the results are computed setting $s = 365$. Columns 1 and 2 show respectively the MSE(x,$\tilde{y}$) and STD(x,$\tilde{y}$) values, where x is the theoretical signal and $\tilde{y}$ is the predicted signal, for different length of H. The raw time series has a MSE equal to 2.327 millimeters and a STD equal to 1.329 millimeters. Column 3 shows the computational training time, in seconds, choosing different values of H.

H	MSE(x,$\tilde{y}$) (mm)	STD(x,$\tilde{y}$) (mm)	time (s)
5	1.169	0.646	109.174
30	1.205	0.678	115.049
100	1.309	0.727	224.441
300	1.262	0.708	2801.995

Table 3 Column 1 shows the STD values for the North Component, in millimeters, for different size of the sliding window parameter. The raw time series have a STD value equal to 0.975 millimeters. Column 2 shows the computational training time in seconds choosing different input size.

s	STD($p_s - \tilde{y_s}$) (mm)	time (s)
7	0.563	8.026
14	0.481	10.153
30	0.432	15.348
183	0.432	66.314
365	0.423	183.304

vectors length parameter H equal to 5. In the following we indicate with $\tilde{x}$ and $\tilde{y_s}$ the raw time steps and the filtered solution respect to the sliding window parameter s, with p_0 and p_s the 5th-degree regression polynomials respect to $\tilde{x}$ and $\tilde{y_s}$. In order to evaluate the scattering of raw data and of $\tilde{y_s}$, we consider the value STD($p_s - \tilde{y_s}$). In the first column of Table 3 we compute the STD values of the filtered time series over different choices of the sliding window parameter s. The best result is achieved using s equal to 365, which is the lag with the highest autocorrelation function value, shown in Figure 3(b). In Figure 3(c) we plot the filtered time series values obtained using $s = 365$ compared with the raw North Component of the static time series. In the second column of Table 3 we report the computational training time, which increases while s increases, as it is for the synthetic case. In Table 4 we report the STD values for the North Component, East Component and Up Component of our real static time series, using a sliding window parameter equal to 365.

4.4 Kinematic time series analysis

In the case of the kinematic time series described in Section 3.3, the aim of the LSTM-Full model is to give an accurate prediction for the incoming time steps, since we want to apply this model for structural monitoring. Since we are not seeking for a filtered solution, we fix the dimension of the LSTM hidden vectors at 100. In order to evaluate the efficiency of the proposed method in this case we use the MSE value between the raw data $\tilde{x}$ and the predicted time steps $\tilde{y}$. The lag with the highest value of the ACF is 86164 (the number of seconds in a sideral day), shown Figure 4 (b), and it is impossible to use it as sliding window parameter, due to the computational cost of the algorithm. In the first column of Table 5 we report the MSE values over different arbitrary size of the sliding window parameter increased until the machine used for this experiment can afford the computational cost. The best results are reached using $s = 60$ or $s = 300$. In Figure 4 (a) and in Figure 4(c) we can see that the predicted time series (red dots) model seems to well reproduce the data (green

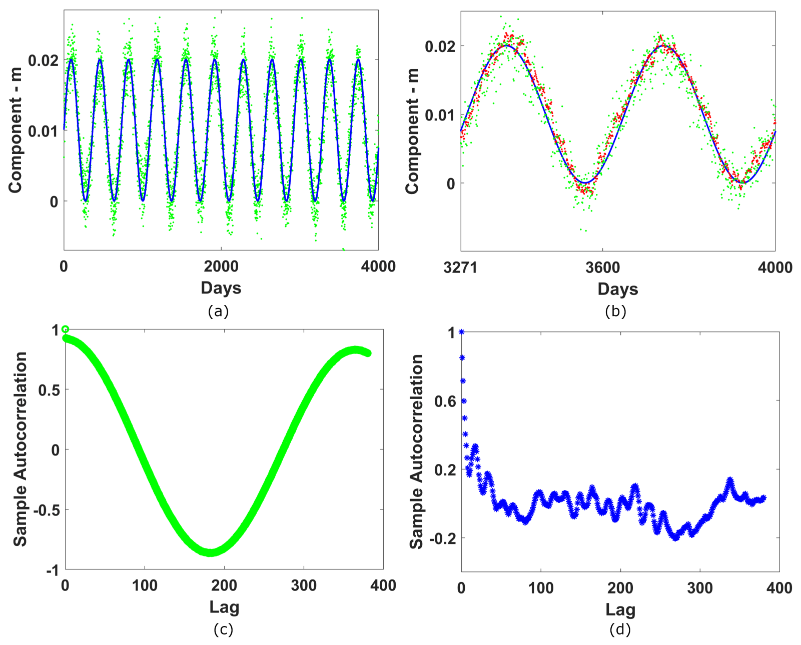


Figure 2 (a) Plot over 4000 days of the raw synthetic time series (green dots) and the theoretical sinusoidal signal (line blue). (b) Predicted time steps over 730 days (red dots) compared with the raw signal (green dots) and the theoretical sinuisoidal signal (blue line).(c) ACF over 400 lag of the raw time series. (d) ACF of the prediction error over 365 lags.

Table 4 In the first column: STD values in millimeters for the North, East and Up component of the real static time series, using a sliding window parameter equal to 365. In the second column: the STD values of the raw signals for each component.

Component	STD($p_s - \tilde{y_s}$) (mm)	STD($p_0 - \tilde{x}$) (mm)
N	0.423	0.975
E	0.559	1.031
U	1.221	2.808

dots). The computational training time, reported in the second column of Table 5, increases while the slinding window parameter increases as for the other case studies. In Table 6 we write down the MSE values for the North, East and Up Component of our time series setting the slinding window parameter equal to 300.

5 Conclusions

The proposed LSTM-Full Network architecture is suitable for denoising and prediction tasks for GNSS time series. Its performance depends on the choice of the sliding window parameter and of the hidden vectors size of the LSTM layer. From the experiments we suggest to choose the sliding window parameter as the lag with the highest value of the autocorrelation function, when the computational resources allow it. The hidden vectors size should be choosen small if the task is denoising otherwise it should be higher to give an accurate prediction on incoming data.

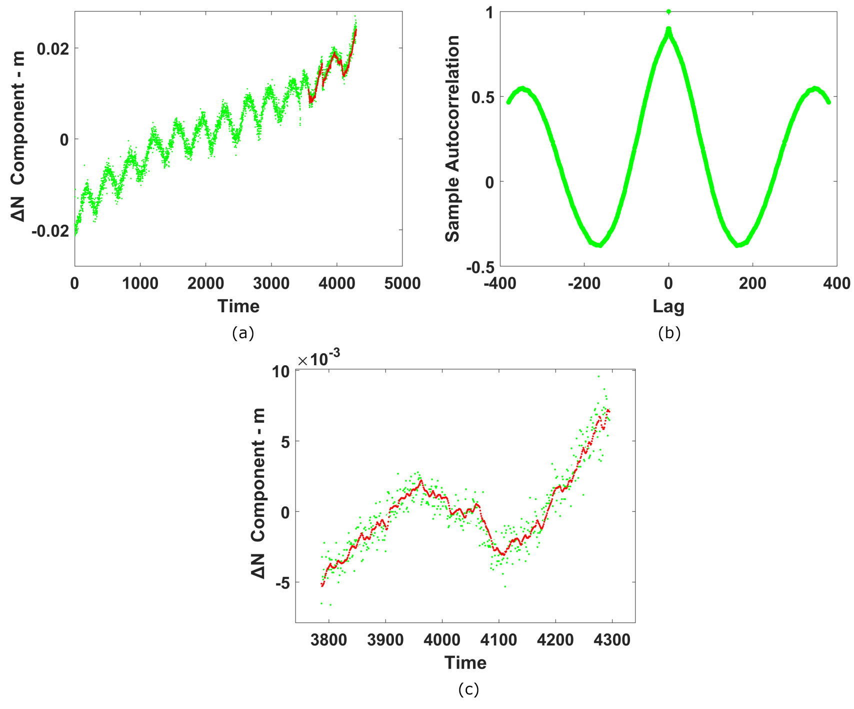


Figure 3 (a) Plot of the raw North Component of the Static Time Series (green dots) and the filtered time series (red dots). (b) ACF over 400 lag of the raw time series. (c) Plot of 509 filtered time steps (red dots), computed choosing s=365, compared with the raw time steps (green dots).

Table 5 Column 1 shows the MSE($\tilde{x}, \tilde{y}$) values for the East Component, in millimeters, for different size of the sliding window parameter. Column 2 shows the computational training time in seconds choosing different input size.

s	MSE($\tilde{x}, \tilde{y}$) (mm)	time (s)
60	1.188	76.28
300	1.184	313.53
1800	1.208	1811.2350
3600	1.220	2872.0116

Table 6 MSE($\tilde{x}, \tilde{y}$) values in millimeters for the North, East and Up component of the real kinematic time series, using a sliding window parameter equal to 300.

Component	MSE($\tilde{x}, \tilde{y}$) (mm)
N	1.741
E	1.188
U	2.497

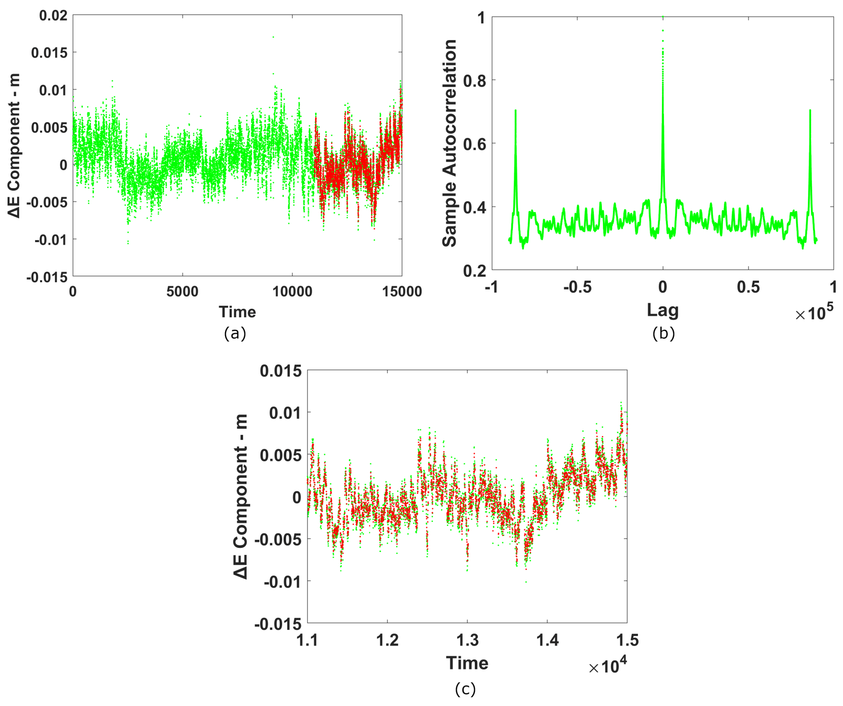


Figure 4 (a) Plot of the raw East Component of the Static Time Series (green dots) and the predicted time series (red dots). (b) ACF over 90000 lag of the raw time series. (c) Plot of the predicted time series (red dots), computed choosing s=300, compared with the raw time steps (green dots).

References

1 Y Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term dependencies with Gradient Descent is Difficult. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66, February 1994. doi:10.1109/72.279181.

2 Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting.* Springer New York, 1996.

3 David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back Propagating Errors. *Nature*, 323:533–536, October 1986. doi:10.1038/323533a0.

4 Claudio Gallicchio. Short-Term Memory of Deep RNN. In *Proceedings for European Symposium on Artificial Neural Networks*, February 2018.

5 Stefano Gandolfi, Luca Poluzzi, and Luca Tavasci. Structural Monitoring Using GNSS Technology and Sequential Filtering. In *Proceedings for FIG Working Week*, May 2015.

6 Alex Graves. Generating Sequences With Recurrent Neural Networks. *ArXiv: 1308.0850*, August 2013. arXiv:1308.0850.

7 Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings*, pages 273–278, December 2013. doi:10.1109/ASRU.2013.6707742.

8 Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 38, March 2013. doi:10.1109/ICASSP.2013.6638947.

9 Sepp Hochreiter. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 6(2):107–116, April 1998. doi:10.1142/S0218488598000094.

10 Sepp Hochreiter and Jürgen Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–80, December 1997. doi:10.1162/neco.1997.9.8.1735.

11 K.M. Hornik, M. Stinchcomb, and H. White. Multilayer feedforward networks are universal approximator. *IEEE Transactions on Neural Networks*, 2, January 1989.

12 Changhui Jiang, Shuai Chen, Yuwei Chen, Boya Zhang, Ziyi Feng, Hui Zhou, and Yuming Bo. A MEMS IMU De-Noising Method Using Long Short Term Memory Recurrent Neural Networks (LSTM – RNN). *Sensors*, 18:3470, October 2018. doi:10.3390/s18103470.

13 Hee-Un Kim and Tae-Suk Bae. Deep Learning-Based GNSS Network-Based Real-Time Kinematic Improvement for Autonomous Ground Vehicle Navigation. *Journal of Sensors*, 2019:1–8, March 2019. doi:10.1155/2019/3737265.

14 Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, December 2014.

15 José Lima and J Casaca. Smoothing GNSS Time Series with Asymmetric Simple Moving Averages. *Journal of Civil Engineering and Architecture*, 6, June 2012. doi:10.17265/1934-7359/2012.06.013.

16 X. Luo, M. Mayer, and B. Heck. Analysing Time Series of GNSS Residuals by Means of AR(I)MA Processes. In *VII Hotine-Marussi Symposium on Mathematical Geodesy*, pages 129–134. Springer Berlin Heidelberg, 2012.

17 Xiaoguang Luo. *GPS Stochastic Modelling - Signal Quality Measures and ARMA Processes*. Springer, January 2013. doi:10.1007/978-3-642-34836-5.

18 Ping-Feng Pai and Chih-Sheng Lin. A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega*, 33(6):497–505, 2005. URL: https://EconPapers.repec.org/RePEc:eee:jomega:v:33:y:2005:i:6:p:497-505.

19 Yuelei Xiao and Yang Yin. Hybrid LSTM Neural Network for Short-Term Traffic Flow Prediction. *Information*, 10:105, March 2019. doi:10.3390/info10030105.

20 Jingzhou Xin, Jianting Zhou, Simon Yang, Xiaoqing Li, and Yu Wang. Bridge structure deformation prediction based on GNSS data using Kalman – ARIMA – GARCH model. *Sensors (Basel, Switzerland)*, 18, January 2018. doi:10.3390/s18010298.