APPLICATION OF LONG SHORT TERM MEMORY NEURAL NETWORKS FOR GPS SATELLITE CLOCK BIAS PREDICTION

PIOTR GNYŚ* AND PAWEŁ PRZESTRZELSKI**

Department of Computer Science
Polish-Japanese Academy of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
*pgnys@pjwstk.edu.pl
**pprzestrzelski@pjwstk.edu.pl

(received: 28.07.2021; revised: 13.10.2021;
accepted: 20.10.2021; published online: 30.12.2021)

Abstract: Satellite-based localization systems like GPS or Galileo are one of the most commonly used tools in outdoor navigation. While for most applications, like car navigation or hiking, the level of precision provided by commercial solutions is satisfactory it is not always the case for mobile robots. In the case of long-time autonomy and robots that operate in remote areas battery usage and access to synchronization data becomes a problem. In this paper, a solution providing a real-time onboard clock synchronization is presented. Results achieved are better than the current state-of-the-art solution in real-time clock bias prediction for most satellites.

Keywords: neural networks, LSTM, time series prediction, clock bias, GNSS, machine learning
DOI: https://doi.org/10.34808/tq2021/25.4/a

1. Introduction

The research presented in this paper aims to develop an algorithm that will predict the bias of GPS satellites onboard atomic clock ensembles bias. An algorithm must be applicable in an environment with low computation power available and where battery charge is a highly limited resource.

1.1. Motivation

During the development of an autonomic marine agent, one of the problems that had to be solved was precise navigation. As robot task is to measure the quality of water in lakes and small streams it is expected that it will operate for long periods in regions where services like the cellular network may not be available. Additionally, plans are to develop a model that will be able to operate

on open sea and navigation issues that robots face in such conditions are even more restrictive on algorithms [1]. With that said main localization technique implemented will be Global Positioning System which in turn bought the issue of the limited precision of civilian variant of GPS as well as its requirement for synchronization with a time reference. For that reason, it was decided that there is a need for an onboard clock bias prediction to limit requirement for synchronization.

1.2. Contribution

In the following paper, a new approach for GPS clock bias prediction based on a Long Short Term Memory neural network is presented. For 20 out of 29 satellites that were analyzed in this work, prediction results were better than the current state of the art and for 6 of them, results were significantly better. Results of the presented research can be used in an offline GPS receiver as an alternative for IGU provided products.

2. Clock bias in GNSS

Due to the nature of Global Satellite Navigation Systems (GNSS) precision time measurement is crucial for accurate localization. In this section information on a basic explanation of how GNSS services work will be presented as well as a more in-depth description of clock ensemble implementation and bias modeling. The current state of the art will be presented as well however no details about underlying the mathematical model will be shared as this is beyond scope of this paper.

2.1. Basics of satellite-based localization systems

All GNSS are variants of beacon-based localization systems [2]. Such systems require information about beacon position and distance between localized objects and beacons. With that information, it is possible to calculate the position of an object in the same reference frame as that of beacons. Both of those tasks are much more difficult in GNSS due to the nature of the beacons. Unlike in the case of stationary beacons, GNSS satellites move with high speed so their position must be calculated based on satellite ephemeris CITEVallado2008. Another problem is distance measurement which without specialized equipment must be done with a time of arrival (ToA) instead of the angle of arrival (AoA) or received signal strength (RSS) [3]. When measuring distance by ToA 3 properties of a signal must be known:

- t_o the time of origination;
- t_a the time of arrival;
- v the velocity.

In the case of the GNSS, the system signal is an electromagnetic wave, therefore, its speed is equal to the speed of light $c \approx 3 \cdot 10^9$ m/s. The time of arrival is recorded when the data frame wavefront reaches the receiver, this means

that the receiver time is used. The origination time is recorded on the satellite according to its local clock and included in the data frame. Thanks to that, the distance can be calculated from the following equation:

$$d = c \cdot (t_a - t_o) \tag{1}$$

However, t_a and t_o use different reference frames, so, should a comparison be possible, they must be transformed into a common reference frame. This is referred to as synchronization of the clocks and it is very important, as a desynchronization at the level of a single nanosecond results in about 30 cm of a positioning error [4].

2.2. Clock modelling

Clocks are devices that provide a reference time by measuring the repetition of a periodic process. One of the most well-known examples of such a process is the pendulum, and even before mechanical clocks, humans would use the rotation of the earth and the resulting sun procession on the sky. Nevertheless, those methods do not provide measurements that would be precise enough for beacon-based location, which is why, atomic clock ensembles are used in the case of GPS. In the case of this research a discreet clock model is used where the clock readout is described as

$$t_c(i) = t_r(i) + b(i) \tag{2}$$

where t_c refers to the time measured by the analyzed clock, t_r is the time given by a reference clock which we assume to give perfect readouts and b is the clock bias. Each of those values are indexed by measurement i and a value of the bias for each step must be predicted in order to correct the clock readouts. There are many approaches to modeling the bias, however, this is beyond the scope of this article, as the method used here relies on adjusting the arbitrary model to fit the already recorded bias data. More information about knowledge-driven bias modeling can be found in the literature related to the frequency stability analysis [5].

2.3. IGU products

The most widely used source of precise clock corrections are products provided by the International GNSS Service (IGS) [6].

Table 1. Variants of IGS products

Type	Accuracy	Latency	Sample interval
Broadcaster	5ns	real time	daily
Ultra rapid – predicted	3ns	real time	15 min
Ultra rapid – observed	150ps	3-9 hours	15 min
Rapid	75ps	17-41 hours	5 min
Final	75ps	12-18 days	30 s

The values shown in Table 1 refer to the satellite clock bias only, IGS products provide other information, a full description of which is available at the online repository. IGS products can be easily divided into two categories:

- real time consisting of transmitted and ultra-rapid predicted half,
- high latency consisting of ultra-rapid observed half as well as rapid and final products.

Solutions that have a high latency are not usable in real-time navigation and as such will not be considered in this work. The observed ultra-rapid types will be used as a source of the reference time, thus, if a bias prediction error is equal to zero, it means that it is the same as provided by the ultra-rapid product observed. As can be seen in Table 1, all real-time solutions provide precision in a range of nanoseconds. This work aims to show that LSTM networks can provide better results than those solutions, while still working at a real-time response latency.

2.4. Data source

This work focuses only on GPS satellites which are divided into the following blocks: I, II, IIA, IIR ,IIR-M ,IIF, III. Blocks I and II were fully retired before research described in that paper began and block III satellites were active for a too-short time to generate enough data. That is why those blocks were not used at all, on the other hand, a single satellite from block IIA was used in the second phase of experiments however it was retired in the meantime and a decision was made to not use it in the next experiments and as such it is not listed in the final satellite pool. This results in a total of 30 satellite clocks analyzed with almost all of them are equipped with Rubidium clock ensemble with exception of two satellites from generation IIF that use Cesium clocks instead. Each satellite have an assigned space vehicle number (SVN) and pseudo random noise (PRN). In this work a PRN will be used as a identifier as it is unique for every active satellite, although it can be used again after said satellite gets retired, and ranges from 1 to 32. Association between satellite PRN, clock and block is shown in Table 2.

Table 2. Bias prediction error in relation to regularization and dropout level

Generation	clock type	satellites
IIA	Rb	18
IIR	Rb	2 11 13 14 16 19 20 21 22 23 28
IIR-M	Rb	5 7 12 15 17 29
IIF	Rb	1 3 6 9 10 25 26 27 30 32
IIF	Cs	8 24

For satellites with rubidium based clock ensembles bias have a very distinct constant drift that makes data appear linear, it can be seen for satellites 01 and 08. On the other hand in the case of cesium based clock ensembles for which constant drift is much smaller other sources of bias are visible like seen for satellite 24. There is also a single satellite for which, during observed period, constant drift was almost not present. This was satellite 14 and while no official source of information describes this behaviour.

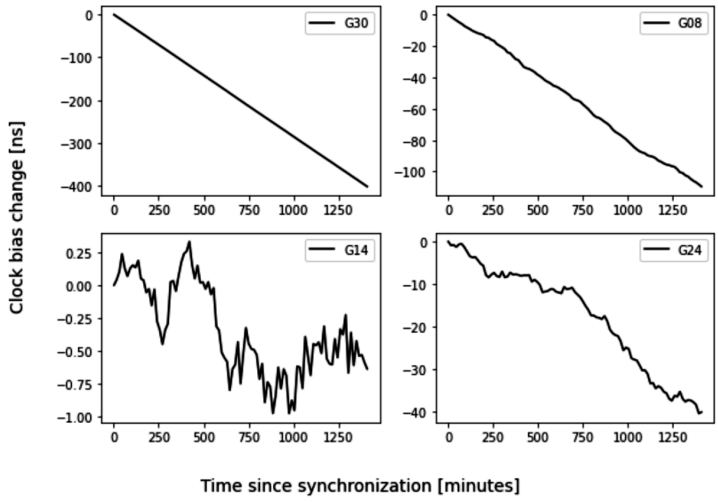

Figure 1. Raw clock bias

2.5. Preprocessing

IGU provides raw clock bias, this poses two problems for the approach used in this work. The first one is that constant clock drift is such a major source of bias that it overshadows other sources as seen in Figure 1 in visualization data seems to be linear. Second issue is non-stationary nature of series, this is a problem as neural networks work best fro stationary data with mean at 0 and values in between -1 and 1. To solve those problems firs series is Differentiated which returns data where other noises besides constant drift are visible as seen on Figure 2.

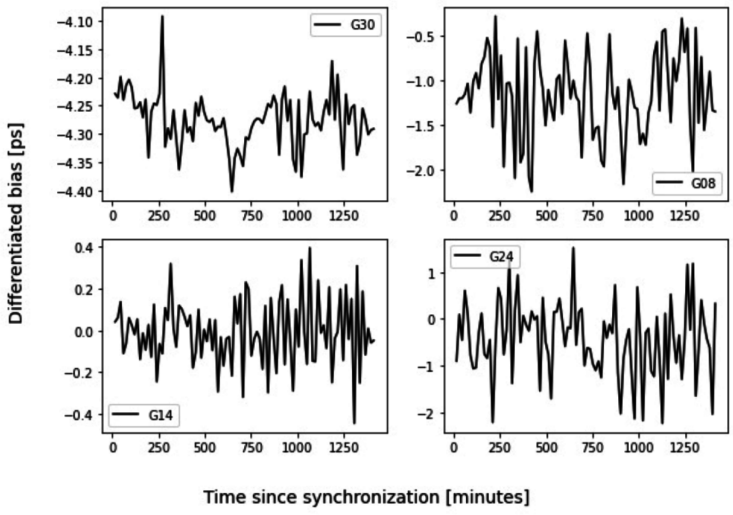

Figure 2. Differentiated clock bias

Constant drift is still present as a shift at the y-axis, to remove it a mean shift must be performed. Finally, data must be scaled so that there will be no values with an absolute value above one. It is, of course, possible for future prediction inputs to have absolute value above one however this will not be a problem as a network can deal with such inputs especially if they appear rarely in series. Another issue is whether to use the same preprocessing for all satellites or should each of them have their parameters. Analysis of data on Figure 2 and Figure 3 shows that constant shift, as well as value range for mean, shifted data can vary radically between satellites. Because of that, separate preprocessing parameters are used for each satellite.

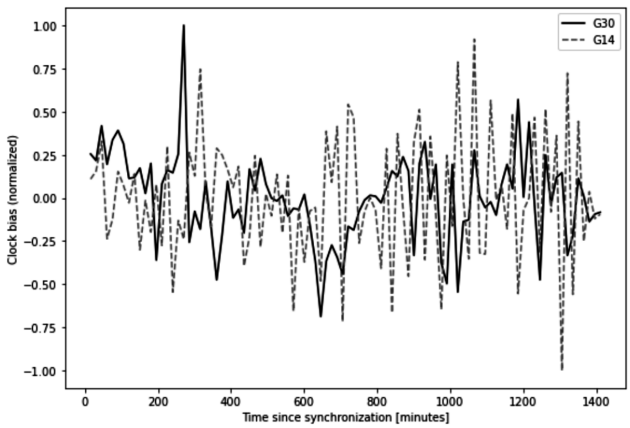

Figure 3. Comparison of diffed clock bias

3. Neural networks

Machine learning (ML) approaches based on artificial neural networks (ANN) are well established as efficient pattern detectors [7–10]. They have also been used in GNSS systems especially since the advent of the deep learning algorithms [11–13]. One of its uses is the prediction of the clock bias [14, 15], which, as mentioned in the previous section, is an essential value in positioning calculations.

3.1. Overview

Like all digital signal processing applications, software neural networks operate in a discreet time and a discreet amplitude domain. This is the reason why a time series clock model was chosen. A basic neuron model (neural layer) was created by McCullough and Pitts in 1943 [16]. This model described the response of a neural layer to multiple signals with the equation $y = \chi(W \cdot x + b)$ where y is the response, x the input, W the weights and b the bias. An algorithm for automated adjustment of weights in relation to the data was proposed in 1958. While this model and its successors were inspired by a biological neuron, they were

much more simplified. One of those simplifications is the lack of a time domain in a model which means that the response of a layer depends on its current input only. This is in contrast with biological networks that are sensitive not only to a signal value but also to its changes over time.

3.2. Long Short Term Memory networks

A simple solution to the problem of the time independence is to concatenate the response of the neural layer from the previous cycle to its input $x'(t) = [x(t)|y(t-1)]$. Such a solution results in the signal propagating through time and influencing the responses of future cycles, if this is the only modification to the feed-forward model, such layer is called a simple recurrent unit (SRU). While this solution makes the model time aware, it has its problems, mainly the signal vanishing issue. Since the input signal from the cycle, n has direct influence only on a response of this cycle and for each subsequent cycle, it is the only trough feedback loop. The influence of input n on the response of cycle $n+k$ grows inverse proportional to k. This means that in this model, it is only those regularities that appear over short periods that can be detected. Making the weights on feedback bigger will not eliminate the problem and instead will replace it with signal an explosion that causes a response to reach maximum value, if a strong signal appeared on the input at least once. One of the possible solutions to this issue is an addition of a long term memory which will regulate the forward and loop back path influence on the neuron response. Such a solution is used in the long-short term memory (LSTM) networks [17] as presented on Figure 4.

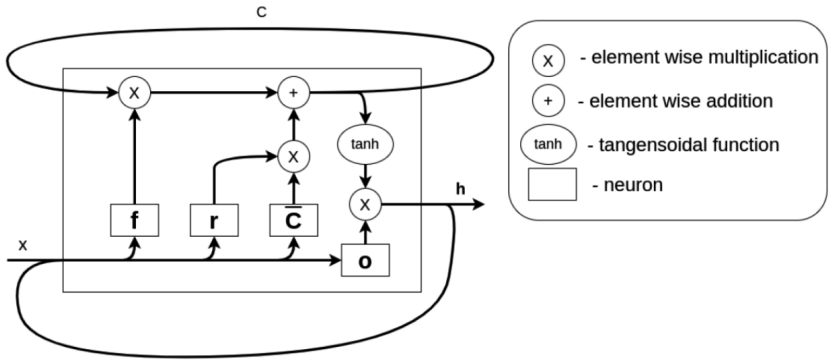

Figure 4. LSTM layer

3.3. Overfitting

Overfitting takes place when the estimator function is adjusted to training data to such a high degree that it can no longer function as a general predictor. For example, if a network that is supposed to recognize cats will be trained on a set that contains only sphinxes, it may be unable to classify other breeds as cats. Overfitted networks provide very high-quality results as long as the input

overlaps with the test set otherwise, the quality of the results drops sharply. In the case of satellite clocks, the predictor can overfit with regard to the following parameters:

- clock type,
- location (orbit),
- epoch.

When selecting a solution a decision must be made what level of generalization the model should represent. Limiting the predictions to the same epochs that were used for training is in contradiction with the network's main goal, predicting future biases. However, as satellites use different models of atomic clocks and are placed on different orbits, attempts at generalization for those proprieties risk a high precision trade-off. Therefore, a separate network will be used for each satellite that will be unable to generalize its predictions for others.

4. Experiments

The main aim of the experiments was to determine if a small LSTM network could achieve results comparable with the IGU rapid predictions considered as the state of the art. The first tests were made on a single satellite and the prediction results were compared against the polynomial regression as well as IGU. As the results had already been better than IGU in the first attempts, all the following tests were carried out on an almost complete set of satellites. This set did not include those satellites that were activated or retired during the experiment period.

4.1. Overview

The experiments were divided into three phases.

1. In the first phase a single satellite was selected and a prediction with a generic LSTM architecture was made. Then it was compared against the polynomial regression as well as the IGU rapid predictions. While achieving results better than the polynomial regression would be considered acceptable at this stage, the LSTM proved to be better than the IGU example which was considered the state of the art. For this reason, a decision was made not to adjust the network model at this stage, as was originally intended, but move to the next stage with an initial model.
2. In the second phase, the network developed in the first phase was tested onset of all active satellites and compared against the IGU rapid prediction. At this phase a comparison against the polynomial regression was dropped, as achieving results worse than IGU was no longer considered acceptable. In this phase LSTM achieved better results than IGU for 5 of 31 satellites.
3. As the second phase provided acceptable results only for a small group of satellites and alternative architectures were tested, more details about them

will be written in a dedicated section. In this phase results better than IGU were achieved for 13 of 30 satellites.

4. The final phase was dedicated to tuning the learning meta parameters and its results are described in more detail in the section dedicated to the experiments.

4.2. Phase 1 – approach validation

A topology with two hidden LSTM layers and a single dense layer as the output was used in the research. Most of the parameters as well as a general topology were set up based on suggestions from [18]. A rectifier (RELU), unipolar as well as bipolar functions were used as the activation function of the LSTM layers. A linear activation only was used for the dense (output) so that there should be no limits on the predicted value. The mean squared error (MSE), the mean average error (MAE) and the root mean square (RMS) were used as a loss function. Two optimizers would be tested, the Root Mean Square Propagation (RMSprop) [19] and the Adaptive Momentum Estimation (Adam) [20].

As the network learning process is stochastic by nature, 10 experiments were run for each configuration, and then the average results were compared. When comparing the results, RMSProp proved to be a better optimizer, as shown in Table 3 and therefore it was used in the following experiments. All the experiments were run on a dataset obtained on 7.22.2018.

Table 3. Optimizers and loss functions

Parameter	**Adam**		**Optimizer**		**RMSProp**	
	MAE	MSE	RMS	MAE	MSE	RMS
Avarage	1.10	1.38	1.15	0.80	0.79	0.87
σ	0.23	0.57	0.23	0.19	0.27	0.16
Min	0.82	0.78	0.89	0.48	0.37	0.61
Max	1.55	2.63	1.62	0.99	1.08	1.04

The repeatability of the results was much better in the Adam optimizer, however, that was due only to the tendency of this algorithm to be stuck in the same local minimum every time it was run.

In the next experiment a value of the loopback was adjusted, its initial value for the previous experiments was set to 12 as an educated guess.

As can be seen in Table 4 the best result was achieved for a loopback value of 32. Finally, a comparison between Adam and RMSProp was made again with all the other parameters set according to the previous experimental results.

When using the adjusted parameters, average errors become better for all the configurations and the results of RMSProp become less consistent due to a higher value of divergence as shown in Table 5. However, RMSProp is still an overall better solution than Adam, and therefore it will be used in the final configuration.

Table 4. Loopback values

Parameter	**Loopback**					
	1	4	12	32	64	96
Avarage	0.48	0.48	0.51	0.47	0.55	0.50
σ	0.01	0.01	0.03	0.01	0.02	0.02
min	0.47	0.46	0.47	0.46	0.52	0.47
max	0.51	0.49	0.55	0.48	0.59	0.54

Table 5. Optimizers and loss functions for adjusted parameters

Parameter	**Adam**		**Optimizer**		**RMSProp**	
	MAE	MSE	RMS	MAE	MSE	RMS
Avarage	0.87	0.95	0.94	0.66	0.63	0.76
σ	0.29	0.56	0.28	0.23	0.43	0.24
Min	0.44	0.30	0.55	0.43	0.29	0.54
Max	1.38	2.06	1.43	1.12	1.71	1.31

Table 6. Basic network configuration

Parameter	**Hidden layer**	
	First	Second
Neuron count	32	128
Activation function	ReLU	ReLU
Dropoff	0.2	0.5
Recurrent dropoff	0.2	0.5
Regularization	L2	L2
Statefullness	NO	NO

After running the experiments and comparing the results, a final set of the network parameters was set as described in a Table 6.

4.2.1. Comparison with other solutions

After preparing an optimal configuration of the prediction network, its predictions were compared against linear approximation, polynomial approximation of 2, 4, and 8 degrees as well as against an IGU rapid product predicted half. As the difference between the results of the polynomial approximation was relatively insignificant between the polynomials of different degrees, all the results will be presented as one rounded to two decimal points.

As seen in Table 7 the LSTM network yielded significantly better results than IGU predicted. What is an interesting observation is that the polynomial approximation appears to work better than IGU for a prediction period of 24 hours. The next comparison was based on a shorter prediction time and as seen in Table 8 LSTM gains more advantage over IGU predicted, the longer the prediction range is. What is more interesting LSTM errors actually drop over time.

Table 7. Prediction errors for 24h range

Algorithm	**Error value**		
	MAE	MSE	RMS
LSTM	0.47	0.36	0.60
IGU-predicted	1.60	2.76	1.66
Linear	1.73	3.21	1.79
Polynomial	1.33	1.87	1.37

Table 8. Prediction errors for 24h range

Algorithm	**RMS for prediction range**		
	6h	12h	24h
LSTM	1.02	0.76	0.60
IGU-predicted	1.26	1.31	1.66

The LSTM predictions were of a higher accuracy than linear and polynomial ones as well as IGU Rapid predicted which is recognized as the state of the art.

4.3. Comparison with the state of the art

Over the course of the experiments, multiple network architectures were tested. In this paper the main focus will be given to experiments from phase 4, where a general architecture was already selected. The final architecture is a 3-layer network with 2 hidden LSTM layers, the first one with a size equal to the input and the second one double that size. The third layer consists of a single densely connected neuron that outputs a single prediction step.

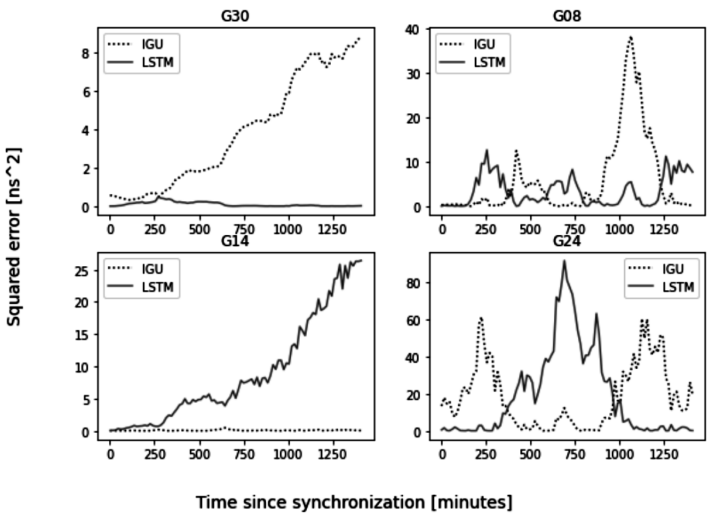

Figure 5. Squared error

As can be observed on the plots presented in Figure 5 there are satellites for which the results achieved by the LSTM network are significantly better than the current state of the art and that difference only deepens with the prediction time. Examples of such a situation are satellites G07, G10 or G30.

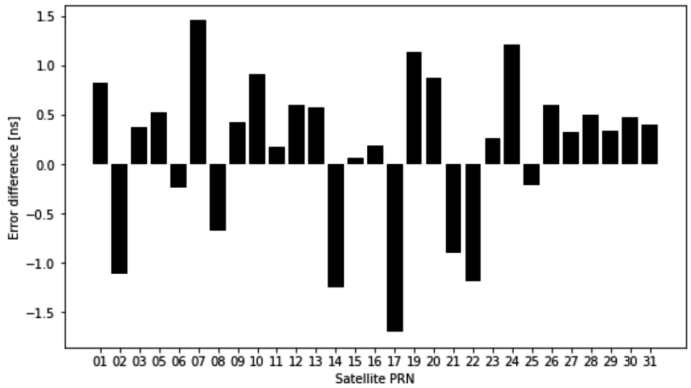

Figure 6. Comparition between absolute prediction errors

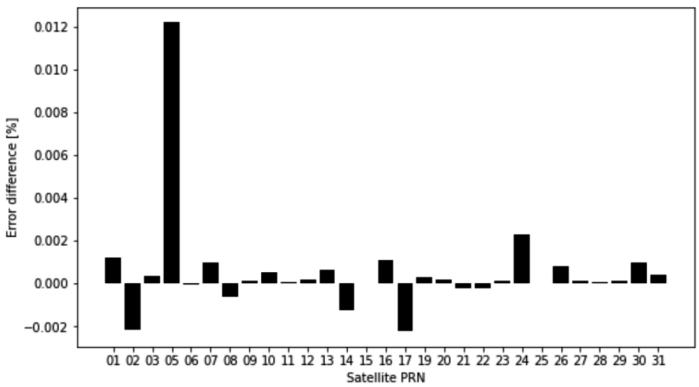

Figure 7. Comparition between relative prediction errors

Other groups represented by satellites G14, G15, or G17 are those for which results of LSTM are worse. Another group consists of satellites for which squared error is smaller for the initial period of prediction however it rises with time resulting in predictions worse than IGU. Examples of such behavior are visible in satellites G01 G09 and G29. The last of groups contain satellites for which prediction quality can vary over time like in the case of G08. As it is shown in Figure 8 in the case of some satellites period for which LSTM has an advantage over IGU predicted part is longer than 9 hours. This is an important value as after that period a synchronization with IGU observed product can be made.

IGU observed products have picoseconds level precision which is satisfactory for most robotic implementation.

As superiority over 9 hour period is enough to consider LSTM solution a preferable one to IGU rapid prediction for experiments in this paper success was achieved for 68% cases. The difference between LSTM prediction error and IGU error is visualized in Figure 6 where value represented on y-axis show by how many nanoseconds the solution proposed in this paper is more precise than state-of-the-art. The advantage of the LSTM solution is even more clear when analyzing relative error as in Figure 7. The proposed solution is capable of providing results comparable or better than state-of-the-art for most satellites.

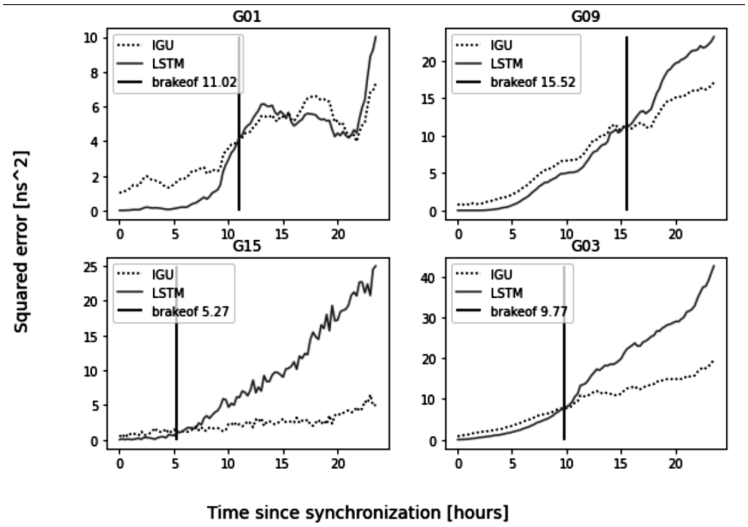

Figure 8. Comparison between LSTM and IGU predictions

5. Conclusions and future research

Experiments described in this paper have proven that even a relatively simple LSTM network can handle bias prediction well compared to the state of the art as shown in Table 9. That indicates the possibility of application of this system in low power embedded system which was a motivation for this research. While the proposed network is relatively simple it is still too complex for implementation in an embedded system. With 86913 parameters and 64-bit float representation, it takes almost 0.7 MB memory. High memory requirements, as well as a need for 64-bit architecture, limits the choice of hardware that is why the next step in this research will be the implementation of a 32-bit trainable LSTM network dedicated for embedded systems.

Table 9. Quality of results

Result category	satellites
Superior	07 10 12 19 26 30
Superior for acceptable time period	01 03 09 13 23 27 28 29 31
Varied	05 08 24 16
Superior for short time period	11 15
Inferior	02 06 14 17 21 22 25

References

[1] Cabrera-Gámez J et al. 2013 *An Embedded Low-Power Control System for Autonomous Sailboats*, Robotic Sailing doi: https://doi.org/10.1007/978-3-319-02276-5_6

[2] Blewitt G 1997 *Basics of the GPS Technique: Observation Equations*, Geodetic Applications of GPS 1–46

[3] Doberstein D 2012 *Fundamentals of GPS receivers: A hardware approach*, Springer New York doi: https://doi.org/10.1007/978-1-4614-0409-5

[4] Enge P 2011 *Global Positioning System: Signals, Measurements, and Performance - Revised Second Edition*, International Journal of Wireless Information Networks

[5] Riley W J 2007 *Handbook of Frequency Stability Analysis*, NIST Special Publication 1065 (**31** Issue 1)

[6] Kouba J 2009 *A Guide to using international GNSS Service (IGS) Products*, Geodetic Survey Division Natural Resources Canada Ottawa

[7] Abiodun O I et al. 2019 *Comprehensive Review of Artificial Neural Network Applications to Pattern Recognition*, IEEE Access **7** 158820-158846 doi: 10.1109/ACCESS.2019.2945545

[8] Faraway J and Chatfield C 2019 *Time series forecasting with neural networks: a comparative study using the airline data*, Journal of the Royal Statistical Society: Series C (Applied Statistics) **47** (2) 231–250 doi: https://doi.org/10.1111/1467-9876.00109

[9] Khan A et al. *A Survey of the Recent Architectures of Deep Convolutional Neural Networks* `http://arxiv.org/abs/1901.06032`

[10] Miller A S 1993 *Vistas in Astronomy*, doi: https://doi.org/10.1016/0083-6656(93)90118-4, **36** 141

[11] Kim H U and Bae T S 2019 *Deep Learning-Based GNSS Network-Based Real-Time Kinematic Improvement for Autonomous Ground Vehicle Navigation*, Journal of Sensors 1–8 doi: https://doi.org/10.1155/2019/3737265

[12] Orus P R 2019 *Using TensorFlow-based Neural Network to estimate GNSS single frequency ionospheric delay (IONONet)*, Advances in Space Research **63** (5) 1607–1618 doi: https://doi.org/10.1016/j.asr.2018.11.011

[13] Wei J et al *The Satellite Selection Algorithm of GNSS Based on Neural Network*, 115–123Overview

[14] Indriyatmoko A et al. 2008 *Artificial neural networks for predicting DGPS carrier phase and pseudorange correction*, GPS Solutions **12** (4) 237–247 doi: https://doi.org/10.1007/s10291-008-0088-x

[15] Wang Y et al 2017 *Improving prediction performance of GPS satellite clock bias based on wavelet neural network*, GPS Solutions **21** (2) 523–534 doi: https://doi.org/10.1007/s10291-016-0543-z

[16] McCulloch W S and Pitts W 1943 *A logical calculus of the ideas immanent in nervous activity*, The Bulletin of Mathematical Biophysics doi: https://doi.org/10.1007/BF02478259

[17] Hochreiter S and Schmidhuber J 1997 *Long Short-Term Memory*, Neural Computation doi: https://doi.org/10.1162/neco.1997.9.8.1735

[18] Chollet F, Deep Learning Phyton
[19] Hinton G E et al 2012 *Lecture 6a- overview of mini-batch gradient descent*, COURSERA: Neural Networks for Machine Learning
[20] Kingma D P and Ba J L 2015 *Adam: A method for stochastic optimization*, 3rd International Conference on Learning Representations, ICLR 2015-Conference Track Proceedings
[21] Vallado D A and Crawford P 2008 *SGP4 orbit determination*, AIAA/AAS Astrodynamics Specialist Conference and Exhibit doi: https://doi.org/10.1007/978-3-662-50370-6_6

Piotr Gnyś is a PhD candidate at the Polish Japanese Academy of Information Technology. He is responsible for teaching various classes ranging from introduction to discrete mathematics to applications of computer vision in robotic systems. Piotr is also responsible for overseeing thesis projects of most of robotics specialization students as well as some students in intelligent data processing. His research profile focuses mostly on artificial intelligence in environments with low computational power as well as on self organizing systems. In addition to his academic career Piotr also is employed as a Data Engineer at Nobelica where he works on embedded machine learning projects.

Paweł Przestrzelski graduated from the University of Warmia and Mazury (UWM) in Olsztyn, Poland, where he obtained his Ph.D. degree at the Department of Satellite Geodesy and Navigation in 2017. He also graduated from the Polish-Japanese Academy of Information Technology (PJAIT) in Warsaw with a Computer Science Engineer's degree in 2019. Paweł Przestrzelski works as a software engineer in CloudFerro and runs undergraduate classes in PJAIT. His research interests cover algorithms and software development for GNSS positioning in obstructed areas and precise navigation.