Neural Computing and Applications (2022) 34:13305–13318
https://doi.org/10.1007/s00521-022-06949-4

S.I. : DEEP LEARNING FOR TIME SERIES DATA

Recurrent neural network model for high-speed train vibration prediction from time series

Jakub Siłka[1] · Michał Wieczorek[1] · Marcin Woźniak[1]

Received: 21 July 2021 / Accepted: 10 January 2022 / Published online: 29 January 2022

Abstract
In this article, we want to discuss the use of deep learning model to predict potential vibrations of high-speed trains. In our research, we have tested and developed deep learning model to predict potential vibrations from time series of recorded vibrations during travel. We have tested various training models, different time steps and potential error margins to examine how well we are able to predict situation on the track. Summarizing, in our article we have used the RNN-LSTM neural network model with hyperbolic tangent in hidden layers and rectified linear unit gate at the final layer in order to predict future values from the time series data. Results of our research show the our system is able to predict vibrations with Accuracy of above 99% in series of values forward.

Keywords Deep learning · Recurrent neural network · LSTM · NAdam algorithm

1 Introduction

Deep learning systems are widely applied in many fields of modern technology due to high precision and many possible applications. Intelligent transportation is related to the variety of developments in technology. We can read about many interesting models used to predict or simulate vibrations. In [8] was proposed a model of vibration analysis for elements joined by friction. Neural networks were used to predict ground vibration in [3]. As a result, warning system for geological activities was developed. Similarly in [4] was presented how to use heuristic model to help on prediction of blast-produced ground vibration. A model composed for open-pit mine vibrations prediction presented in [12] was also based on neural network.

Deep learning models are very often used in diagnostic purposes for variety of technical systems. In [11] was presented how to evaluate displacement from data to improve transportation systems. Model presented in [5] was developed to help on fault detection in residual generator from collected data. There are also very interesting models of machine learning devoted to the topic of vibration analysis in means of transport. In [10] was presented an idea to evaluate vibration of high-speed railway by using deep learning. The model proposed in [13] was oriented on effect of vibrations that come from trains to urban areas. This topic was explored in [17] to show various aspects of such vibrations. Results presented in [21] described how energy consumption analysis may help in prediction of interior noise in a high speed. A wide spectrum of numerical experiments for various urban trains was presented in [7]. In [18] was presented an analytical approach to use neural networks for simulation of dynamical elements in moving vehicles. Among models used in analysis of time series, Recurrent Neural Networks (RNN) are presented in many efficient applications. An idea discussed in [20] has been developed for application of RNN in traffic noise prediction. Results presented in [6] show how to model RNN to infer global temporal structure. Presented results show that this type of neural networks can well adapt to variety of examples. In [19] was discussed efficiency of RNN in stochastic time series analysis, while results of [16] show solution to improve long time series processing with fuzzy granulation.

✉ Marcin Woźniak
marcin.wozniak@polsl.pl

Jakub Siłka
kubasilka@gmail.com

Michał Wieczorek
michal_wieczorek@hotmail.com

[1] Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland

In this article, we want to discuss application of deep learning model to predict potential vibrations of high-speed trains from time series of recorded travels. In our work, we have prepared a novel lightweight solution for predicting the future train vibrations using time series data provided as an input. What is more, the proposed model performs not only well in terms of training and evaluation times but also in high Accuracy terms. Our proposed prediction model is developed using Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) neurons. We have examined which training model would help to better train this system, and as a result, we propose to use NAdam training algorithm. Tests were also done for different configurations of the developed classifier. We have tested how many readings in a time proposed classifier can need to work with highest potential efficiency. We have also tested how long would it be necessary to train the classifier to work with assumed efficiency. Because we are working with time series data, we have decided to use LSTM neurons in our architecture as they have the best ability to remember past values and thus predicting trends. Other neuron types are ineffective because of great regression in performance in both training time and Accuracy standpoint. Results of our findings are compared and discussed to draw conclusions for future work on efficient predictors of long time series of sensor readings.

2 Deep learning model

The research in our project is oriented on prediction of possible vibrations in high-speed trains by using deep neural network model. We have developed a model of Recurrent Neural Network to analyze signal from the high-speed train. In Fig. 1 is presented a sample explanation of our research assumptions. We assume that high-speed train is moving at regular speed on the track. During motion vibration signal is read from suspension sensors and processed by developed LSTM-RNN to decide if the train is stable as visible in Fig. 2. If the system discovers malfunction or difference from regular situation on the track, it sends warning to the operator. As a result, driving of such high-speed train becomes more safe and easier.

2.1 Data

In order to develop our system, we have used the dataset provided by Enfang Cui [2]. This collection consists of 200 hours of metro train vibration energy harvesting data collected at intervals of 2 minutes. After simple data analysis, we have spotted that values range is quite similar across the whole range and the differences are small. Thus, we have decided to use standard Minimum-Maximum scaling algorithm for our data preprocessing to fit them into the 0-1 range. This preprocessing helped the network results to be rescaled to the original shape. For our experiments, this collection was split into the train/test subsets with the ratio of 70:30.

2.2 Recurrent neural network

Because we are working with the time series data, we have decided to use Recurrent Neural Network in order to allow it to learn the dependencies of values in time and that way give us better results. In our RNN model, we have used one Input layer followed by 8 LSTM layers and finally 2 standard layers. For all hidden layers, we have used Hyperbolic Tangent activation function, and for the output

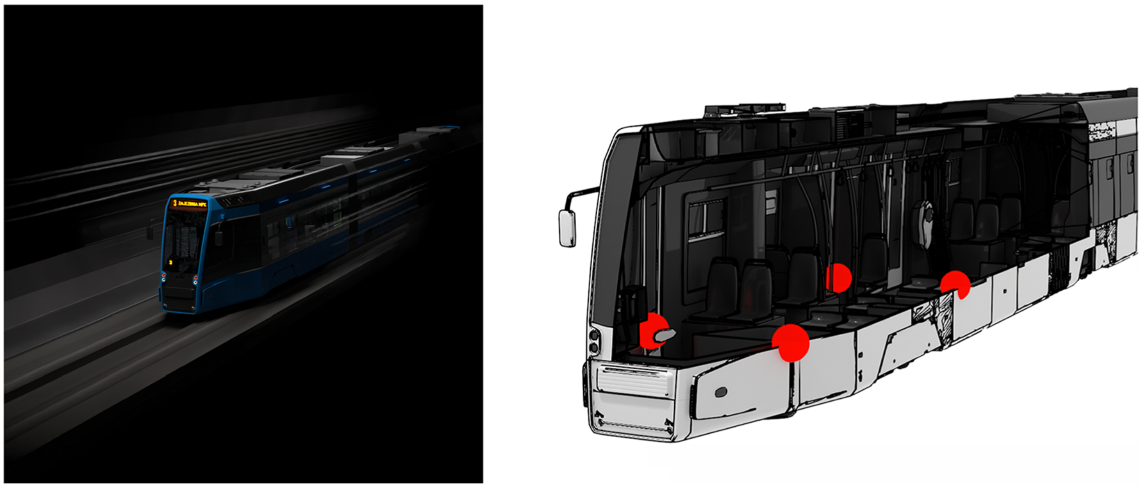


Fig. 1 In our research, we assume that high-speed train or metro is moving on the truck and sensors are reading vibrations from suspension system. Recordings are analyzed by our developed deep learning model to help operator during driving

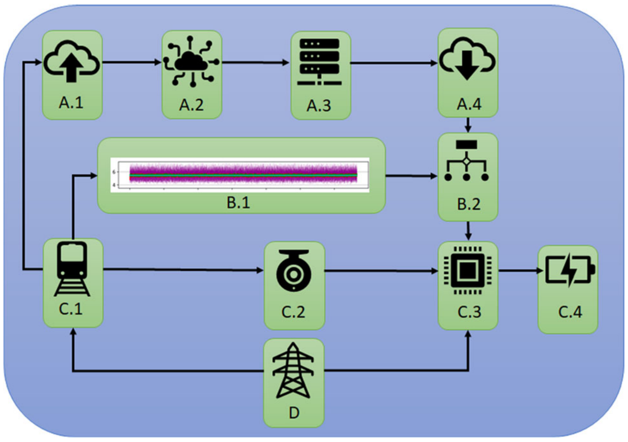


Fig. 2 Sample diagram of the data processing in a system. Power (D) is sent to the train, and then, data (B.1) from the train (C.1) are sent to the neural network (B.2) which transmits predicted values to the control system (C.3). It decides how much power can be recovered from recuperation (C.2). Secondly, it also decides what energy reserves are stored in batteries (C.4). In the meantime, the data from the train sensors go to the cloud (A.1) where it is processed (A.2) so that they can be added to the database to enable improved learning of the next version of the neural network classifier. After approval by the technical laboratory (A.3), data are loaded to train (A.4) and replace the old version of system knowledge

layer we have decided to use ReLU function as we are dealing with the value prediction which can be higher than the maximum value of the training set so we do not want to be restricted by -1 to 1 range that the Tanh function gives us. In some cases, better results could be given by using leaky ReLU or ELU activation functions; however, in this

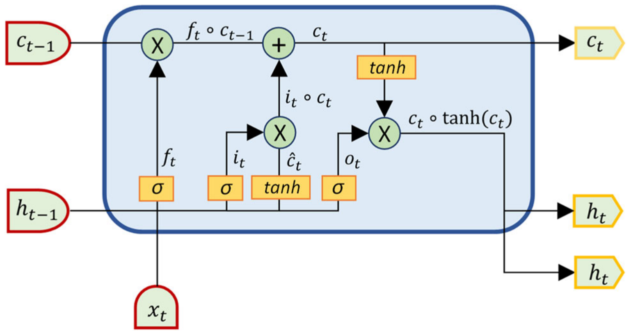


Fig. 4 Applied Long Short-Term Memory (LSTM) neuron model

dataset it is not needed because vibrations could not be negative, especially when using ReLU is much more computational heavy and one of our goals was to reduce the training times to the minimum. The final architecture is shown in Fig. 3, while LSTM neuron model is presented in Fig. 4.

Each of LSTM neurons works with memory recall ability mathematically modeled as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_t), \quad (1)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \quad (2)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \quad (3)$$

$$\hat{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \quad (4)$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \hat{c}_t, \quad (5)$$

Fig. 3 Applied Recurrent Neural Network model. We have set 8 LSTM layers for 1 input layer and 7 hidden layers. For all hidden layers, we have used hyperbolic tangent activation function, and for the output layer we have decided to use ReLU

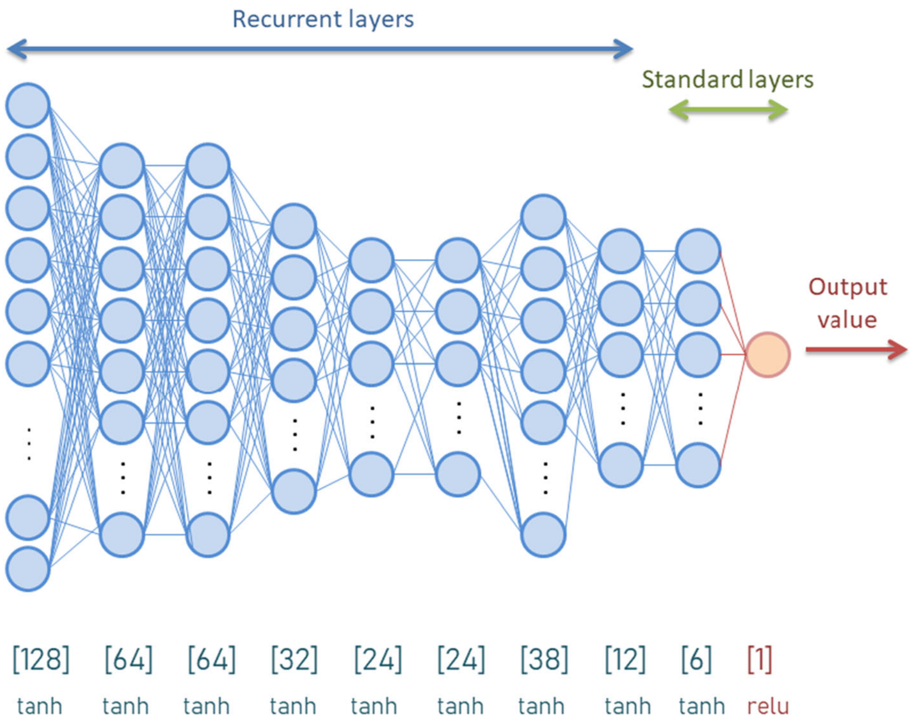


Table 1 Comparison of results for using different neural network architectures

Architecture	Accuracy (%)
Bi-LSTM	99.14
LSTM	99.12
ANN	88.11
Gru-RNN	99.081

$$h_t = o_t \circ \tanh(c_t), \tag{6}$$

where we assume: x_t as input time series array, f_t forget gate activation function, i_t input/update gate activation value, o_t output gate activation value, h_t hidden state, $\tilde{c}_t$ cell input activation, c_t cell state, while other symbols are W,b weights matrices and bias, σ sigmoid activation function, tanh hyperbolic tangent activation function. For calculations, we assume $c_0 = 0$ and $h_0 = 0$.

Algorithm 1 NAdam training model

```
1: Generate random weights,
2: while $global\ error\ value\ \varepsilon < error\_value$ do
3:    Shuffle data in training set TS,
4:    for each mini-batch inside TS do
5:       Step = Step + 1,
6:       Calculate gradient vector on this mini-batch,
7:       Update vector $a$ eq. (7),
8:       Update vector $b$ eq. (8),
9:       Rescale vector $\hat{b}$ eq. (10),
10:      Rescale vector $\hat{a}$ eq. (13),
11:      Update variable $\hat{x}$ with new values eq. (14),
12:   end for
13:   Calculate $global\ error$ eq. (15),
14: end while
```

3 Model training

After conducting experiments to get the best possible Accuracy of our predictor, we have used NAdam optimization algorithm. Because of high performance and short training times, it is widely used in machine learning research. To improve model Accuracy and reduce training even more, we have used learning rate decay. It allowed

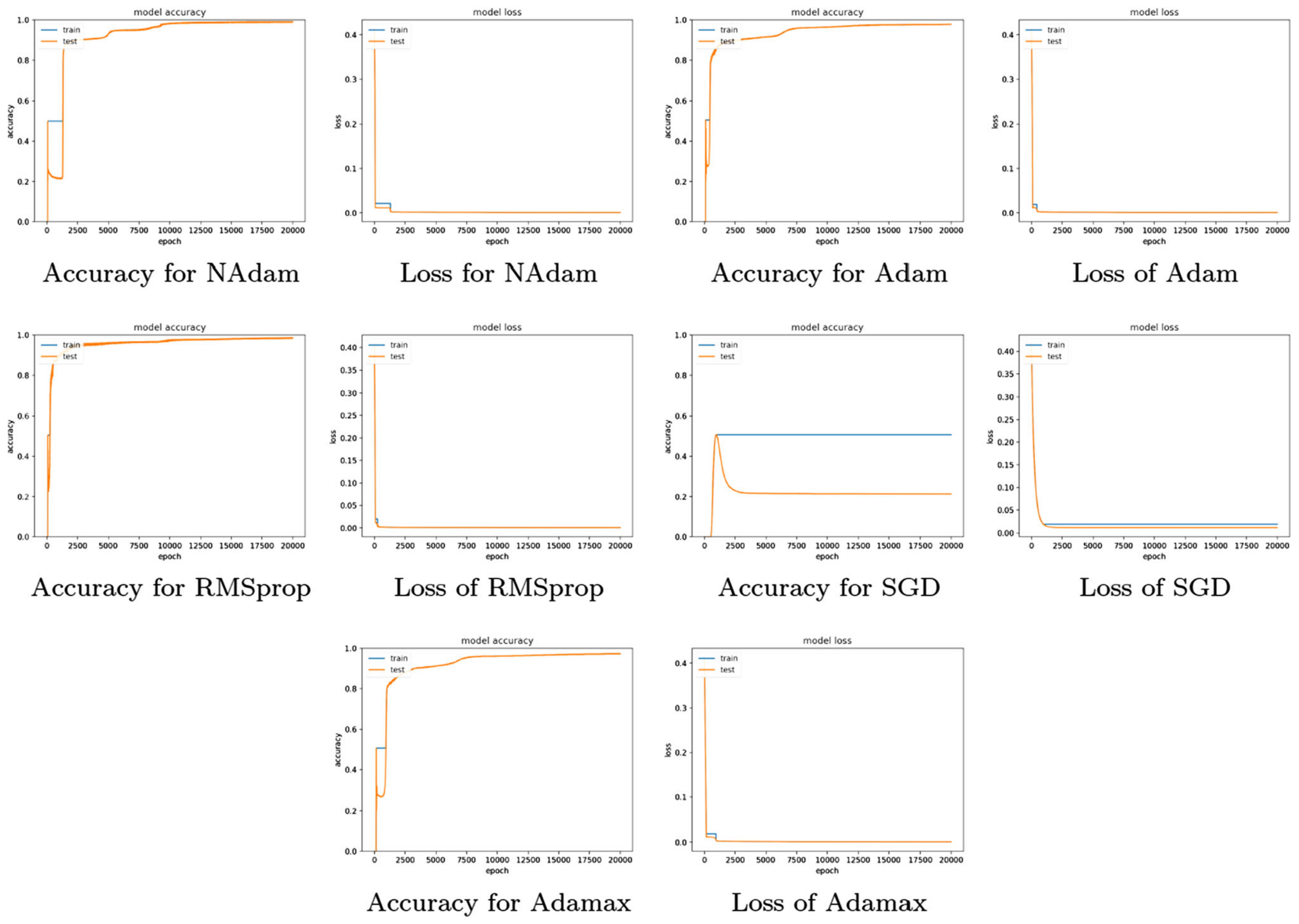


Fig. 5 Comparison of training results between different tested algorithms. For each of the algorithms, we can see results of Accuracy and Loss function change in training. In blue line, we can see results on train data, while in orange we can see results on test data

Table 2 Comparison of using different time step (10 is equal 10 readings) during RNN training (calculated after $\frac{1}{3}$ of NAdam iteration training)

Time Step	Accuracy (%)
10	96.41
11	89.27
12	97.26
13	90.48
14	49.41
15	91.22
16	97.59
17	89.94
18	97.56
19	49.44
20	98.29
21	93.07
22	98.20
23	50.50
24	50.44
25	91.71
26	98.17
27	50.49
28	50.21
29	50.28

deep neural network to first quickly adapt to the given problem and then slowly polish the final efficiency. NAdam formula can be described as follows. First we need to compute values a_i and b_i which are later used for computing correlations $\hat{a_i}$ and $\hat{b_i}$:

$$a_i = \theta_1 a_{i-1} + (1-\theta_1)z_i, \tag{7}$$

$$b_i = \theta_2 b_{i-1} + (1-\theta_2)z_i^2, \tag{8}$$

where θ_1 and θ_2 are constant hyper-parameters and z is a current gradient value of the error function. Correlations are described as follows:

$$\hat{a_i} = \frac{a_i}{1-\theta_1^i} \tag{9}$$

$$\hat{b_i} = \frac{b_i}{1-\theta_1^i} \tag{10}$$

After that the formula for updating weights is defined as follows:

$$x_{i+1} = x_i - \frac{\eta}{\sqrt{\hat{b_i}} + \gamma}\hat{a_i}, \tag{11}$$

where γ is a small, constant value and η is a learning rate. Next we apply NAG formula to classical Adam using equations below:

$$x_i = x_{i-1} - \eta\frac{\theta_1 a_{i-1}}{\theta_2 b_{i-1} + (1-\theta_2)z_i^2 + \gamma} - \eta\frac{(1-\theta_1)z_i}{\sqrt{\theta_2 a_{i-1} + (1-\theta_2)z_i^2 + \gamma}} \tag{12}$$

Finally, we modify Adam algorithm update rule; thus, we need to change equations for $\hat{a_i}$ and x_i:

$$\hat{a_i} = (1-\theta_1)z_i + \theta_{1\,i+1}a_i \tag{13}$$

$$x_i = x_{i-1} - \eta\frac{\hat{a_i}}{\sqrt{\theta_{2i}} + \gamma} \tag{14}$$

We have used Mean Squared Logarithmic Error as fitness function for our model, since it gave well understanding of the training process for this type of data.

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i=0}^{N}(log(y_i+1) - log(\hat{y}_i+1))^2 \tag{15}$$

4 Experiments

Specification of our computer is as follows:

- *CPU* AMD Ryzen Threadripper 2950X 16 cores/32 threads
- *GPU* 2× NVidia RTX 2080 8GB
- *RAM* 64GB

Because in our work we are dealing with time series data, we have chosen empirically to use Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) neurons. However, to ensure that our choice was correct we have done a comparison between RNN-LSTM, classical artificial neural network (ANN) and other types of RNN models. Results are shown in Table 1. We can see that in our tests models which use Long Short-Term Memory neurons are having the best Accuracy. Our proposed model is giving result similar to Bi-LSTM; however, from our experiments we see that training of our model was much shorter, what gave us a big advance and justified our choice to present this idea as a main result of our research in this project.

4.1 Finding the best training algorithm

In our experiments, we searched for the best training algorithm for this dataset. To do this, we have conducted series of experiments running the same network architecture training by different methods of optimization. Results are found in Table 4 and more detailed description can be found below:

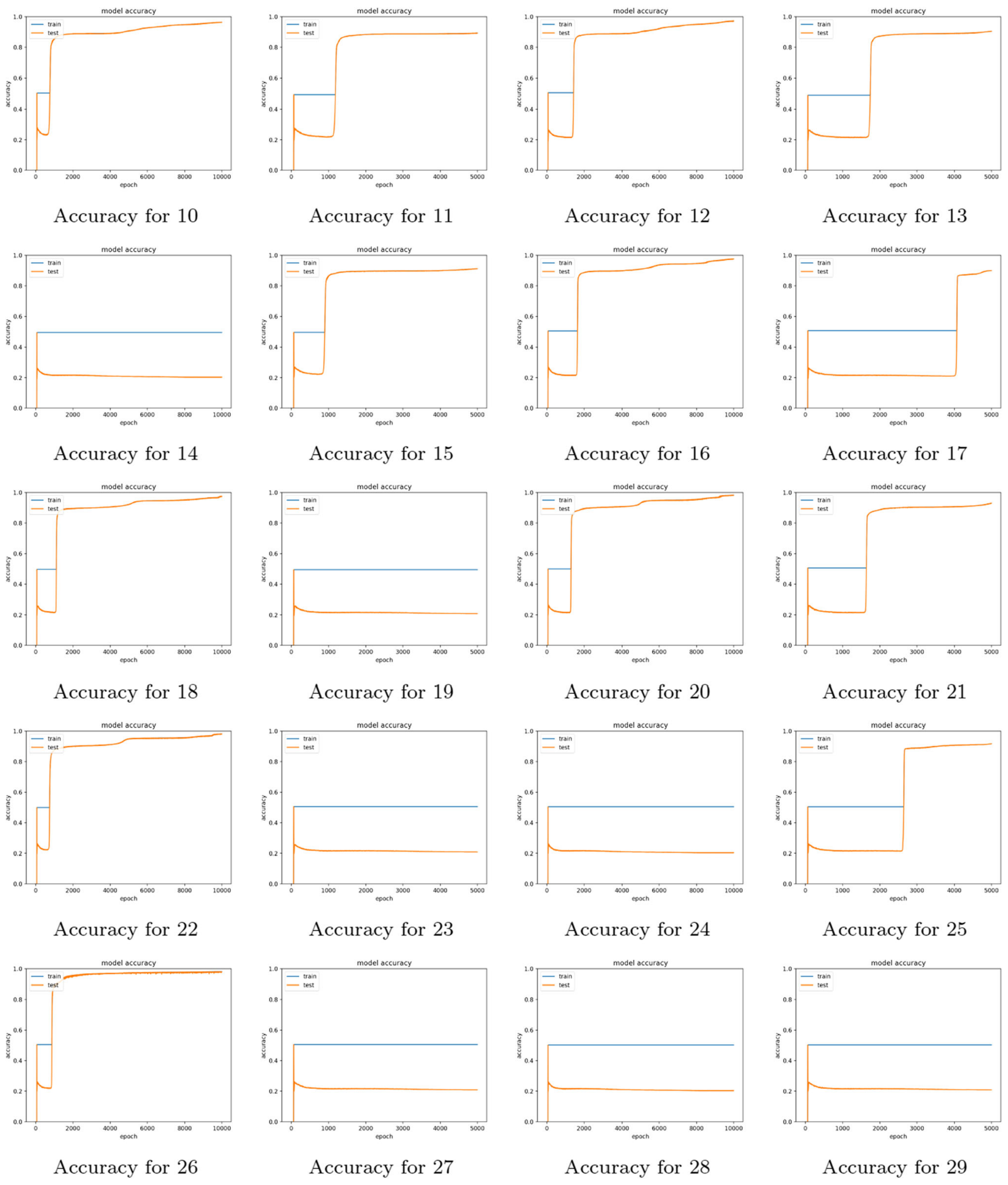


Fig. 6 Results of training measured in changes to Accuracy using different time steps for proposed RNN architecture. In blue line, we can see results on train data, while in orange we can see results on test data

- *Adamax* Network did learn to predict values with an Accuracy of 96.29-,
- *Adam* Network reached 97.78% Accuracy; however, it struggled to correctly predict the highest and the lowest values,

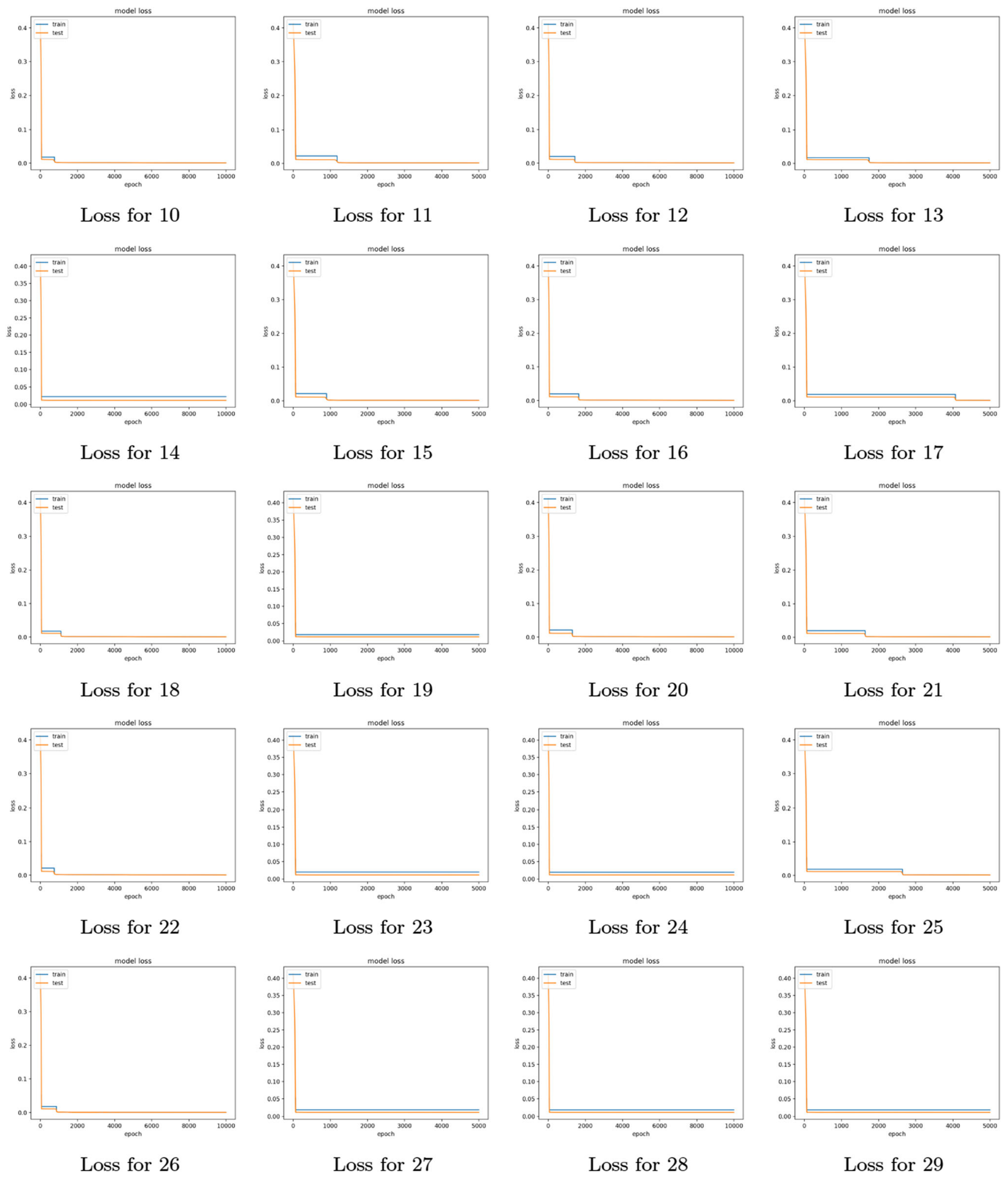


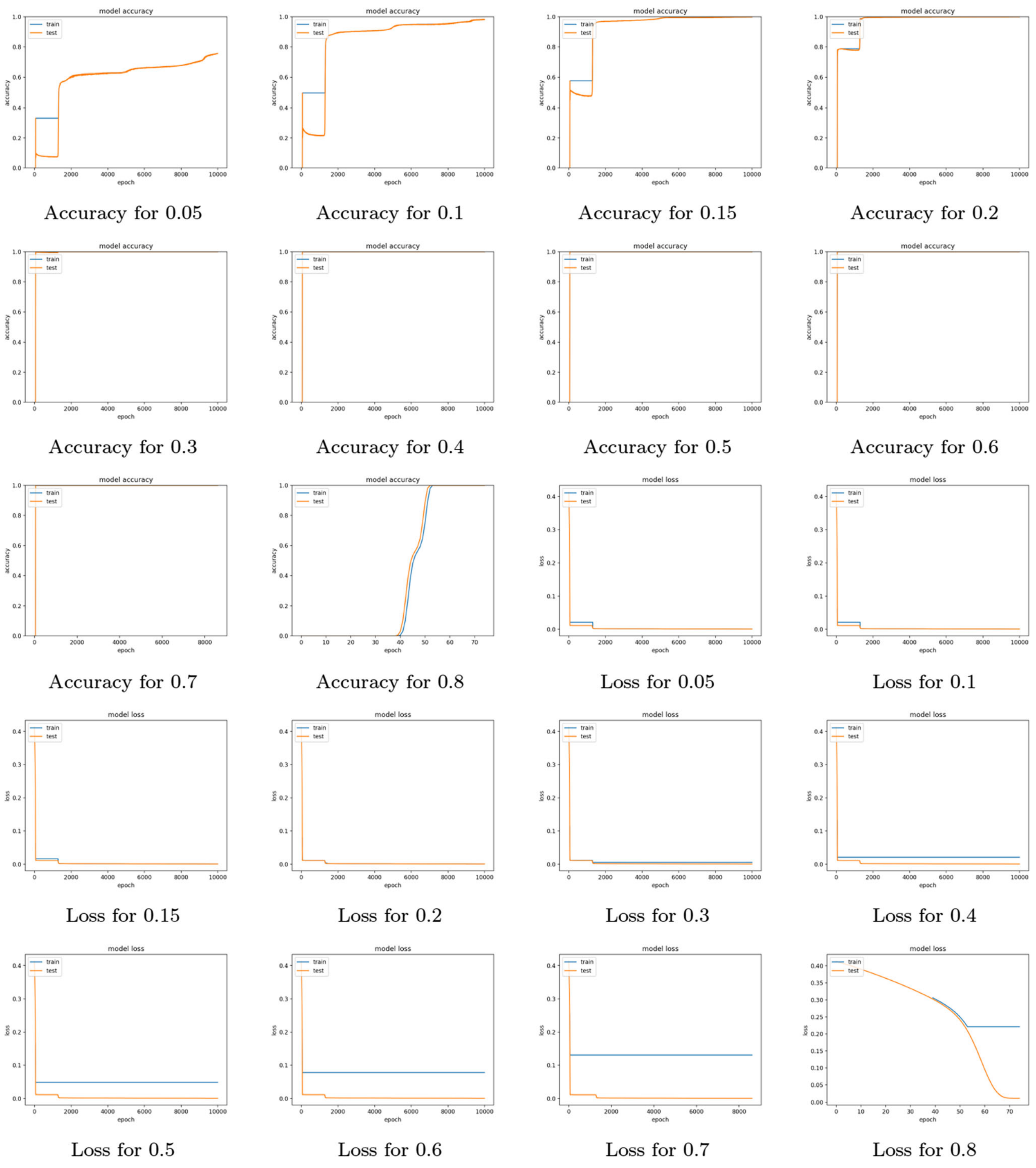
Fig. 7 Results of Loss function using different time steps for the RNN architecture. In blue line, we can see results on train data, while in orange we can see results on test data

- *RMSprop* Results were better than Adam and reached 98.60%; however, the network had the same troubles with the highest and the lowest values as the Adam algorithm.

- *NAdam* Predictor reached over 99.12% making it the best fitting algorithm for our task.

More results containing charts of changes to Loss functions and Accuracy in time domain are found in Fig. 5. As we can see, some did not learn any valuable dependencies

Fig. 9 Results of searching for how many future values should the ▶ network predict. In blue line, we can see results on train data, while in orange we can see results on test data



Accuracy for 0.05 | Accuracy for 0.1 | Accuracy for 0.15 | Accuracy for 0.2

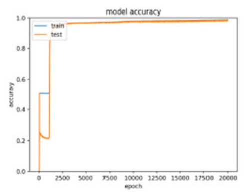
Accuracy for 0.3 | Accuracy for 0.4 | Accuracy for 0.5 | Accuracy for 0.6

Accuracy for 0.7 | Accuracy for 0.8 | Loss for 0.05 | Loss for 0.1

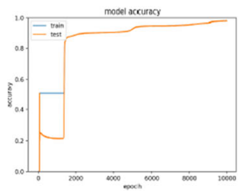
Loss for 0.15 | Loss for 0.2 | Loss for 0.3 | Loss for 0.4

Loss for 0.5 | Loss for 0.6 | Loss for 0.7 | Loss for 0.8

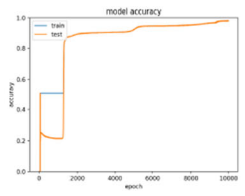
Fig. 8 Results of Accuracy and Loss function in training for prediction of using different error margins. In blue line, we can see results on train data, while in orange we can see results on test data



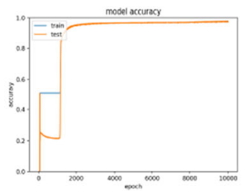
Accuracy for 10



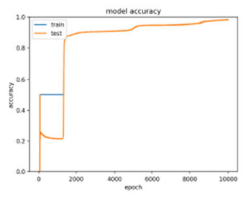
Accuracy for 100



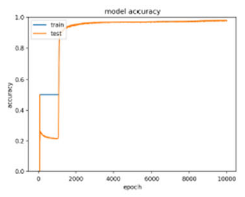
Accuracy for 1000



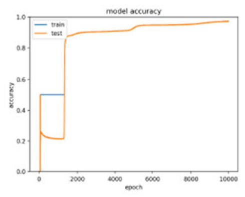
Accuracy for 15



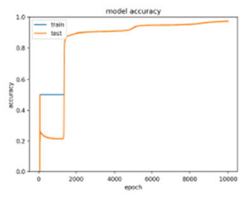
Accuracy for 1500



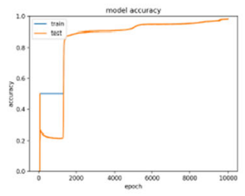
Accuracy for 20



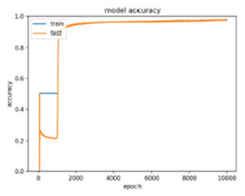
Accuracy for 25



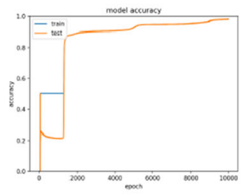
Accuracy for 250



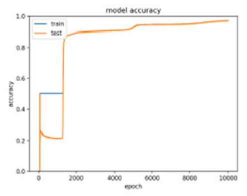
Accuracy for 2500



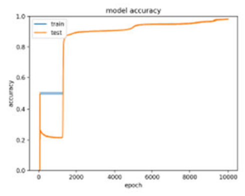
Accuracy for 5



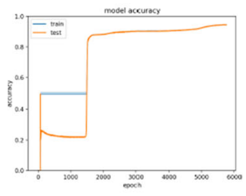
Accuracy for 50



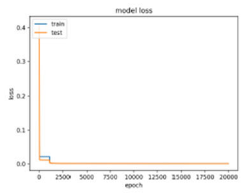
Accuracy for 500



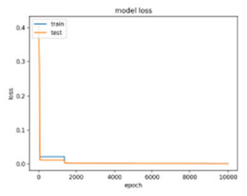
Accuracy for 5000



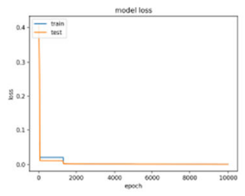
Accuracy for 10000



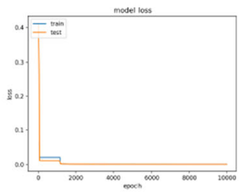
Loss for 10



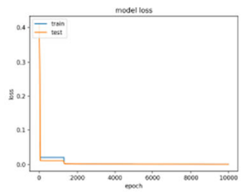
Loss for 100



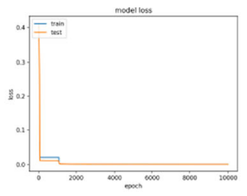
Loss for 1000



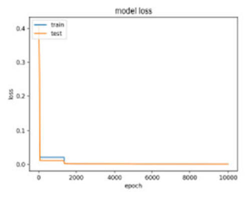
Loss for 15



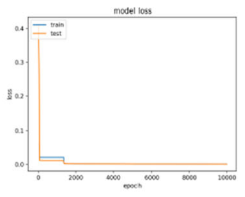
Loss for 1500



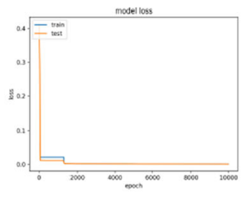
Loss for 20



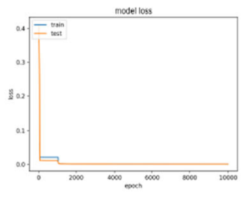
Loss for 25



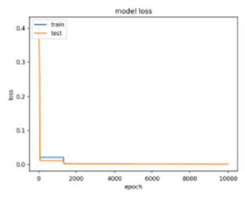
Loss for 250



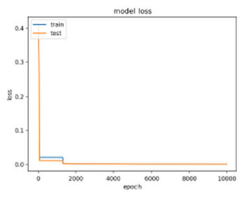
Loss for 2500



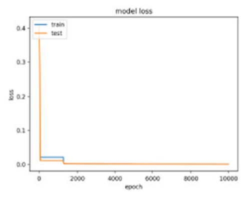
Loss for 5



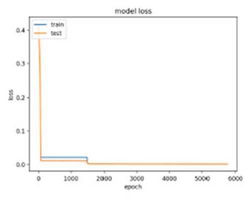
Loss for 50



Loss for 500



Loss for 5000



Loss for 10000

Table 3 Comparison of using different error margins (for NAdam)

Error margin	Accuracy (%)
0.05	75.66
0.1	99.12
0.15	99.91
0.2	99.99
0.3	100.0
0.4	100.0
0.5	100.0
0.6	100.0
0.7	100.0
0.8	100.0

Table 4 Comparison of Training Times and Accuracy for each algorithm

Algorithm	Accuracy	Training Time
Adamax	97.29%	56 min 3.90 s
Adam	97.78%	56 min 7.82 s
RMSprop	98.60%	48 min 32.06 s
NAdam	99.12%	48 min 35.86 s

transforming all data into the same, constant values. In some cases, it worked; however, the Accuracy was very high reaching over 95%. Finally, we have chosen NAdam as the training algorithm due to the best Accuracy. In numerical experiments, we have divided our data into 70:30 for training and testing in our experiments. K-fold cross-validation has been used for training data. The model was trained in configuration with reshuffled training data at each iteration of applied training algorithm.

4.2 Searching for the best time step

To improve Accuracy on time-changing data even more, we had to find the best time step for our proposed predictor. Results are shown in Table 2, and charts are presented in Figs. 6, 7. As we can see, proposed RNN-LSTM is very sensitive to changing the time step value. For example, changing it from 20 to 19 resulted in almost 50% reduction in Accuracy; however, further reduction to 18 increased it again. Because of that, we have tested some different configurations, and finally, we have chosen the time step value of 20 as it gave us the best results. In Table 2, we can see how the number of calculations improves decision processes in the proposed system. We can see that the number of iterations above 10 does not influence much this process. After experiments, we have concluded that 10 iterations are a minimum to make the system both flexible to input data and maintain prediction ability well.

4.3 Computing overall Accuracy based on error margin experiments

Our architecture predicts future time series values with a very high Accuracy. As shown in Figs. 8, 9 and Table 3, if we allow the network to have an error margin of 5% of the real value predictor Accuracy reaches about 75.66% which leads to the conclusion that proposed network architecture is very accurate with most of its predictions. However, if we step up the margin to 10%, the Accuracy stays around 99.12%. That implies that proposed RNN-LSTM network correctly predicts the overall trend, but it is still not intelligent enough to be ideal in terms of the exact value for time series. On the other hand, stepping even more gives us the Accuracy of 100% so we can conclude that this network is a good fit for the given problem.

4.4 Training times

During our numerical experiments, despite logging model's performance in terms of Loss and Accuracy, we have also tested training times. Results are shown in Table 4. As we can see, different optimization algorithms have also much different training times, which can lead us to confirm our choice of NAdam training algorithm for machine learning applications of time series data.

4.5 Non-machine learning prediction methods

In this section, we want to compare our proposed approach to statistical method. We have, therefore, implemented various measures. Sample Moving Average (SMA) for vibration prediction from readings is calculated as:

$$SMA = \frac{\sum_{i=1}^{n} c_i}{n} \tag{16}$$

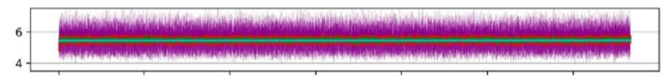
where n is number of assumed factors, c_i is i-th value.

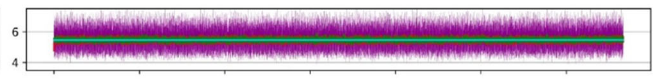
Exponential Moving Average (EMA) for vibration prediction from readings is calculated as:

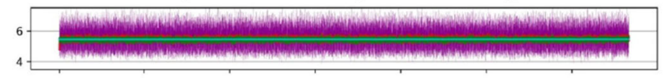
$$EMA_i = Y_i \cdot a + EMA_{i-1} \tag{17}$$

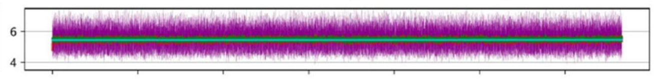
where $EMA_0 = Y_0$ and Y is a value from the i-th value of the set taken into account, a - is $\frac{2}{n-1}$ where n is the number of values of the set.

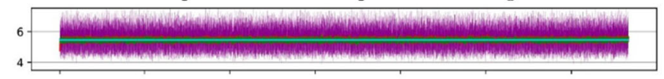
Linear Trend Line for vibration prediction from readings is built for the number of detected cases as:

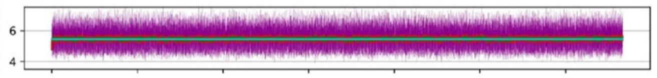

Graph for a time step of 10 readings.

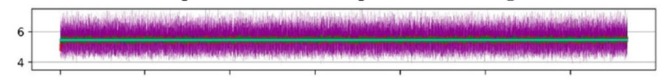

Graph for a time step of 11 readings.

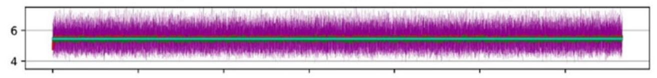

Graph for a time step of 12 readings.

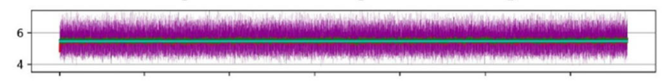

Graph for a time step of 13 readings.

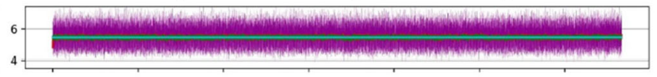

Graph for a time step of 14 readings.

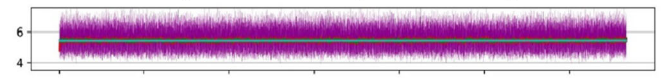

Graph for a time step of 15 readings.

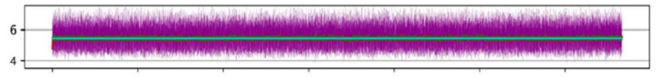

Graph for a time step of 16 readings.

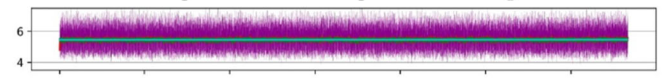

Graph for a time step of 17 readings.

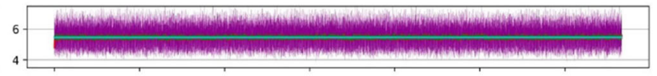

Graph for a time step of 18 readings.

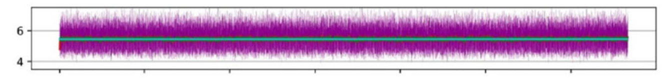

Graph for a time step of 19 readings.

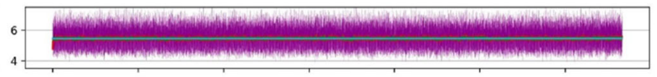

Graph for a time step of 20 readings.

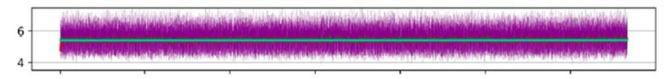

Graph for a time step of 21 readings.

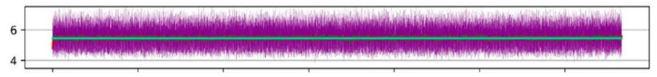

Graph for a time step of 22 readings.

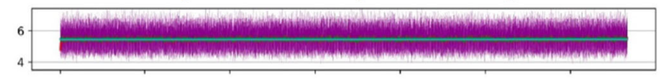

Graph for a time step of 23 readings.


Graph for a time step of 24 readings.


Graph for a time step of 25 readings.


Graph for a time step of 26 readings.


Graph for a time step of 27 readings.


Graph for a time step of 28 readings.

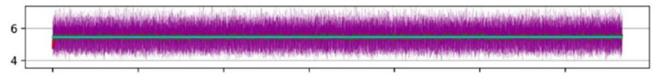

Graph for a time step of 29 readings.

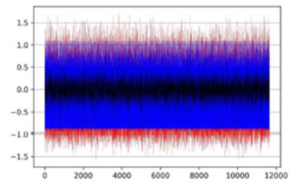
Fig. 10 Sample results of prediction from statistical methods. On vertical, we can see vibration readings from sensors, while on horizontal we can see interval. The violet line represents sensor readings, while light blue represents trend line. Red represents result from Exponential Moving Average and green line represents result from Sample Moving Average

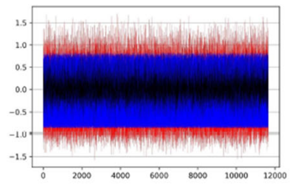
$$LTL = a \cdot x + b \tag{18}$$

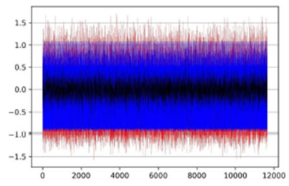
where X_i is i-th value of the set, $\overline{X}$ is arithmetic mean of the set, $Y_i$is i-th value of the set, $\overline{Y}$ is arithmetic mean of the set. Directional coefficient a of the trend line and free expression b are calculated as

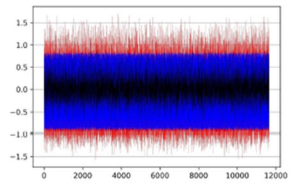
$$\begin{cases} a = \dfrac{\sum_{i=1}^{n}(X_i - \overline{X}) \cdot (Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \\ b = \overline{Y} - a \cdot \overline{X} \end{cases} \tag{19}$$

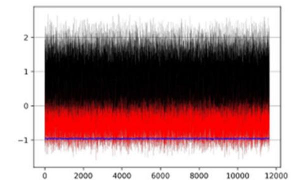
Results of vibration prediction are presented in Fig. 10. The violet line represents the data from sensor, while light blue exemplifies the LTL. Red and green chart constitutes Exponential Moving Average and Sample Moving Average, respectively. EMA and SMA have the same value of a "step"; therefore, we display them on the same charts.

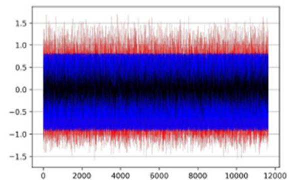

Graph for a time step of 10 readings.

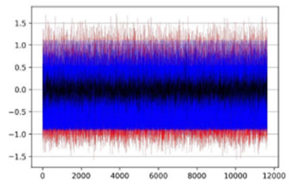

Graph for a time step of 11 readings.

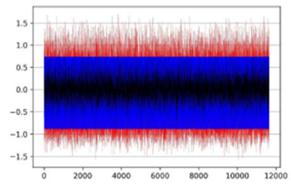

Graph for a time step of 12 readings.

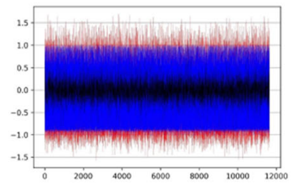

Graph for a time step of 13 readings.

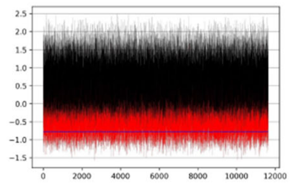

Graph for a time step of 14 readings.

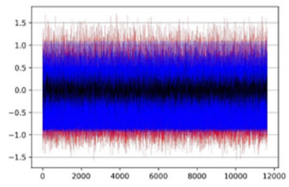

Graph for a time step of 15 readings.

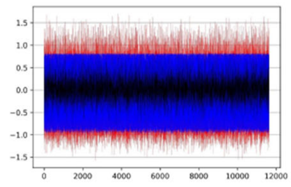

Graph for a time step of 16 readings.

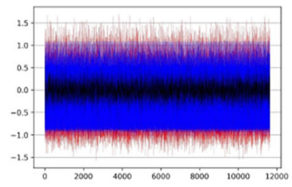

Graph for a time step of 17 readings.

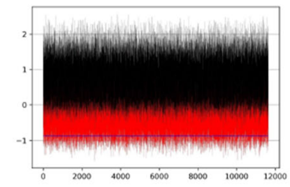

Graph for a time step of 18 readings.

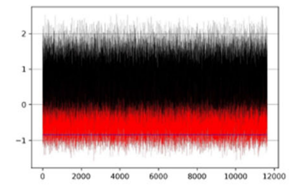

Graph for a time step of 19 readings.

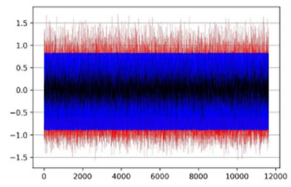

Graph for a time step of 20 readings.

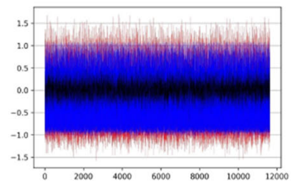

Graph for a time step of 21 readings.

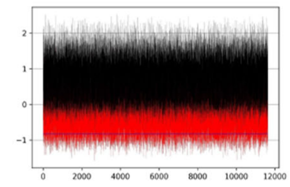

Graph for a time step of 22 readings.

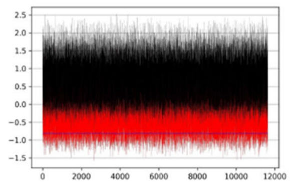

Graph for a time step of 23 readings.


Graph for a time step of 24 readings.


Graph for a time step of 25 readings.


Graph for a time step of 26 readings.


Graph for a time step of 27 readings.


Graph for a time step of 28 readings.

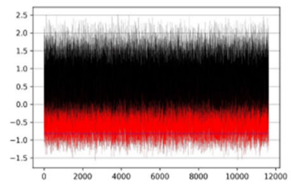

Graph for a time step of 29 readings.

Fig. 11 Sample results of prediction. On vertical, we can see vibration readings from sensors, while on horizontal we can see interval. In red, we can see original recording from sensors on the track. In blue, we can see how proposed RNN-LSTM model predicts vibration on the track. In black, we can see difference between them, what represents possible response from suspension system to keep the high-speed train stable

5 Conclusions

In Fig. 11, we see comparison between original sensor readings and results of prediction from our developed deep learning system. When we analyze results, we can see that original readings without prediction and dumping system show higher fluctuations in each interval than our proposed deep learning model. As a result, we see that application of neural network is very efficient in analysis of data samples. For our experiments, we have selected RNN with LSTM as

Table 5 Comparison of different solutions for prediction of time series data

	Solution	MAE	MSE
Our	Recurrent Neural Network with Long Short-Term Memory neurons	0.019	0.00056
Camero et al. [1]	Recurrent Neural Network based on the Mean Absolute Error random sampling	0.036	N/A
Qin et al. [14]	Dual-stage attention-based Recurrent Neural Network	0.19	0.014
Sahoo et al. [15]	Recurrent Neural Network with Long Short-Term Memory neurons	0.361	N/A
Liu et al. [9]	Dual-Stage Two-Phase attention-based Recurrent Neural Network	0.0661	N/A

initial test has shown better results than for standard architecture. We have selected NAdam training model as the most efficient training algorithm in our research case. This selection was based on comparative analysis, which results were discussed in section of experiments. We have compared various error margins and prediction intervals. Results have shown that our model is efficient in case of error at level above 0.1 which is very good results. As a summary of our research experiments, we can conclude that proposed deep learning architecture is interesting design for data collections with sensor readings in time intervals.

In Table 5, we can see comparisons to other models. Our proposed model is working very well. Proposed LSTM-RNN has reached lowest value MSE among compared solutions. For compared MAE, proposed model is second among all compared. Therefore, we can see that proposed model is working very well with time series data. Our model is based on RNN with LSTM neurons, while other is using attention mechanisms or random sampling ideas. Results show that proposed by us simplified processing gives MSE and MAE lowest than other presented models. These results confirm efficiency of our idea. We have also tested other configurations in our experiments. After some simple testing with changing the layers type from ReLU to ELU, we have experienced no gain in Accuracy (in some cases even up to 1% decrease), what's more the training time performance was highly decreased coming about 1/4 times longer than the ReLU. The reason is because of the higher computational complexity of ELU function and adding more negative data which are not only unusable for this model as all values should be above zero, but also creates additional noise. Therefore, we decided to present the proposed architecture as our final model in this task. Our proposed model has three sensitive points. One of them is time as we have to assume how many individual sensor readings we can take, so that both proposed neural network can properly predict and the relevant systems can react. The self-learning method may turn out to be another problem, i.e., when excessively monotonous sections of the route may influence neural network. The last vulnerable point is reality itself, as long as we can very accurately predict standard train runs, we cannot be sure whether they will always be like that, as heavy construction works near the traction can affect the entire system.

6 Final remarks

In our article, we proposed vibration prediction model by the use of LSTM-RNN deep learning architecture. We collaterally presented optional applications of those predictions to create a system allowing to decrease sensible vibrations to satisfying extent. As the entire architecture was developed based on the data from telegraphic sensors during a drive, the model was devised in regard to that sort of transport. However, being obtained with accurate measures and equipment, we will be able to recreate the systems on the rules of a different transport, including air transport, road transport and water transport. In the future, we are willing to focus on adaptive ability, which will yield in further development in deep learning models for smart transport.

Acknowledgements Authors would like to acknowledge contribution to this research from the Rector of the Silesian University of Technology, Gliwice, Poland, under proquality grant no. 09/010/RGJ21/0007.

Author contributions All authors contributed equally to the research by conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft, review & editing.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relations that could have appeared to influence the work reported in this paper.

References

1. Camero A, Toutouh J, Alba E (2018) Comparing deep recurrent networks based on the mae random sampling, a first approach. In: Conference of the Spanish Association for Artificial Intelligence, pp. 24–33. Springer
2. Cui E (2020) Metro vehicle vibration energy harvesting dataset. IEEE Dataport.http://dx.doi.org/10.21227/0jcz-t470
3. Hajihassani M, Armaghani DJ, Marto A, Mohamad ET (2015) Ground vibration prediction in quarry blasting through an artificial neural network optimized by imperialist competitive algorithm. Bull Eng Geol Environ 74(3):873–886
4. Hasanipanah M, Naderi R, Kashir J, Noorani SA, Qaleh AZA (2017) Prediction of blast-produced ground vibration using particle swarm optimization. Eng Comput 33(2):173–179
5. Jiang Y, Yin S, Kaynak O (2020) Optimized design of parity relation-based residual generator for fault detection: data-driven approaches. IEEE Trans Ind Inform 17(2):1449–1458
6. Kim JZ, Lu Z, Nozari E, Pappas GJ, Bassett DS (2021) Teaching recurrent neural networks to infer global temporal structure from local examples. Nat Mach Intell 3(4):316–323
7. Kouroussis G, Vogiatzis KE, Connolly DP (2017) A combined numerical/experimental prediction method for urban railway vibration. Soil Dyn Earthq Eng 97:377–386
8. Krack M, Salles L, Thouverez F (2017) Vibration prediction of bladed disks coupled by friction joints. Archiv Comput Methods Eng 24(3):589–636
9. Liu Y, Gong C, Yang L, Chen Y (2020) Dstp-rnn: a dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction. Exp Syst Appl 143:113082
10. Ma S, Gao L, Liu X, Lin J (2019) Deep learning for track quality evaluation of high-speed railway based on vehicle-body vibration prediction. IEEE Access 7:185099–185107
11. Manogaran G, Nguyen TN (2021) Displacement-aware service endowment scheme for improving intelligent transportation systems data exchange. IEEE Transactions on Intelligent Transportation Systems
12. Nguyen H, Drebenstedt C, Bui XN, Bui DT (2020) Prediction of blast-induced ground vibration in an open-pit mine by a novel hybrid model based on clustering and artificial neural network. Nat Resour Res 29(2):691–709
13. Paneiro G, Durão FO, e Silva MC, Neves PF (2018) Artificial neural network model for ground vibration amplitudes prediction due to light railway traffic in urban areas. Neural Comput Appl 29(11):1045–1057
14. Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G (2017) A dual-stage attention-based recurrent neural network for time series prediction. arXiv preprint arXiv:1704.02971
15. Sahoo BB, Jha R, Singh A, Kumar D (2019) Long short-term memory (lstm) recurrent neural network for low-flow hydrological time series forecasting. Acta Geophysica 67(5):1471–1481
16. Tang Y, Yu F, Pedrycz W, Yang X, Wang J, Liu S (2021) Building trend fuzzy granulation based lstm recurrent neural network for long-term time series forecasting. IEEE Transactions on Fuzzy Systems
17. Weber C, Karantonis P (2018) Rail ground-borne noise and vibration prediction uncertainties. In: Noise and Vibration Mitigation for Rail Transportation Systems, pp. 307–318. Springer
18. Woźniak M, Połap D (2017) Hybrid neuro-heuristic methodology for simulation and control of dynamic systems over time interval. Neural Netw 93:45–56
19. Yin Z, Barucca P (2021) Stochastic recurrent neural network for multistep time series forecasting. arXiv preprint arXiv:2104.12311
20. Zhang X, Kuehnelt H, De Roeck W (2021) Traffic noise prediction applying multivariate bi-directional recurrent neural network. Appl Sci 11(6):2714
21. Zheng X, Dai W, Qiu Y, Hao Z (2019) Prediction and energy contribution analysis of interior noise in a high-speed train based on modified energy finite element analysis. Mech Syst Sig Process 126:439–457