ISSN 2224-087X. Електроніка та інформаційні технології. 2021. Випуск 16. С. 11–19
Electronics and information technologies. 2021. Issue 16. P. 11–19

УДК 519.85.004.42 DOI: https://doi.org/10.30970/eli.16.2

# LOW COMPLEXITY RECURRENT NEURAL NETWORKS FOR EDGE COMPUTING

O. Sinkevych

*Ivan Franko National University of Lviv,*
*50 Drahomanova str., Lviv, UA-79005, Ukraine*
*oleh.sinkevych@lnu.edu.ua*

This paper is dedicated to the development of recurrent neural networks in order to supplement the AI-based devices like microcontrollers and other mist computing systems. Due to the insignificant computational power of the edge devices the aim of the study is to design and analyze low complexity sequence models for a basic sensory time series forecasting on an example of univariate indoor temperature data. The description of data preparation and transformation followed by the models configuration via different architectures like simple LSTM and GRU is provided. To calculate an optimal set of hyper-parameters for the multiple neural network architectures a genetic algorithm has been implemented. The results of numerical experiments conducted for each model configuration consisting of both unidirectional and bidirectional cell connections are discussed. In addition to these studies the scheme of deploying the developed low-complexity models on STM32 microcontroller joined with the high-performance hub is proposed.

*Key words*: edge computing, recurrent neural networks, time series, genetic algorithm.

## Introduction

During the last decade edge computing research which is significantly based on the machine learning principles has been gaining more and more weight among other AI directions. The need to design full-fledged and cloud-independent systems is due to a number of reasons, such as privacy concerns, latency and autonomy [1]. By now the most popular used cases of edge computing systems are remote monitoring of assets in the oil and gas industry, autonomous vehicles, smart grids, predictive maintenance and smart homes [2]. The latter is mostly centered on the IoT energy management technologies to prevent energy losses and optimize their use for heating and cooling. Major challenges for developing the corresponding systems are: 1) providing a reliable mechanism for obtaining and accumulating data; 2) designing and selection of sufficient rule-based or machine learning algorithms to process gathered data in order to produce required response; 3) low cost hardware architecture design. Due to the software aspect of the mentioned problems, the machine learning engineering is of large-scale interest providing accurate and efficient solution to many of smart home issues [3, 4]. Despite the existence of the multiple smart home solutions, it is still necessary to keep development of the software applications within a scope of edge computing and AI-based methods such as promising deep learning models. For instance, in [5] authors propose the neural network model to manage the heating behavior in smart home. In [6] researchers present the lightweight and secure solution for the edge home automation problem; in [7] authors propose a joining cloud and edge collaborative processing algorithm strategy using the

Kubernetes deployment. One of the possible deep learning solutions for the edge smart home system is the prediction of some sensor time series data like indoor/outdoor temperature, humidity, energy consumption etc. For this reason a bunch of deep learning models on the basis of recurrent neural networks has been suggested [8, 9].

In order to complement the existing approaches to the design of predictive models for the needs of a smart home, here the goal is to develop and analyze several models of recurrent neural networks for the edge system software. Recurrent neural networks (LSTM, GRU) are the recommended choices for time series modeling and are widely applied to different sequence problems. The main challenge for this research is to ensure simplicity of the models in balance with their accuracy and computational complexity. The reason is the possibility of further model deployment on low-powered edge computing device STM32 microcontroller using the specific X-CUBE-AI expansion package [10].

**Data description and processing**

To build and analyze different recurrent neural network architectures for the predictive models the temperature time series data have been chosen. Because of the relatively simple behavior and accessibility the temperature data nevertheless are the valuable measurement for every smart home energy system. Also, this choice is considered as a starting point for further system scaling with the extra sensory measurements. For the numerical experiments, the indoor and outdoor temperature data have been obtained using the DS18B20 digital thermometers joined with Raspberry Pi3 database hub to store measurements. All the data were saved in InfluxDB database installed on a RPi3 microcomputer. This experiment has been conducted at Smart Autonomous System laboratory of Electronics and Computer Sciences faculty of Ivan Franko National University of Lviv.

In fig. 1 the 30 minute resampled indoor temperature data for March, 2021 as well as the corresponding histogram are shown.

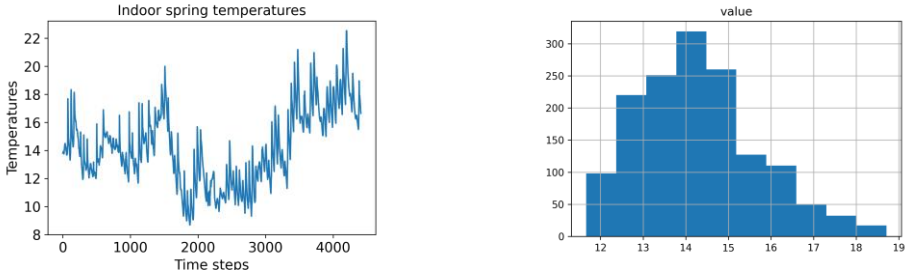


Fig. 1. March temperature data and histogram

The histogram indicates the quasi-stationarity (close to the shape of the normal distribution curve with a pronounced right tail). Since the recurrent neural networks process stationary time series in a way better than non-stationary ones, one should ensure the stationarity of the given data. To do that an Augmented Dickey-Fuller unit root test has been applied. As a result, the value of statistics -5.283274 is less than -3.435, which suggests that we can reject the null hypothesis (non-stationarity of the series) with a significance level less than 1%.

Before normalization and reduction of the temperature range to the matrix form, it is necessary to divide the training $\boldsymbol{Y}_T = (\boldsymbol{X}_{train}, \boldsymbol{Y}_{train})$ and validation $\boldsymbol{Y}_V = (\boldsymbol{X}_{valid}, \boldsymbol{Y}_{valid})$ sets in

some proportion, for example 80% and 20%, respectively, where $\boldsymbol{X}_{train}$ is the feature training set and $\boldsymbol{Y}_{train}$ is the label training set (the same for $\boldsymbol{X}_{valid}$ and $\boldsymbol{Y}_{valid}$). The choice of proportion depends on the length of the time series $\boldsymbol{y} = [y^1, y^2, \ldots, y^n]$, $n = 1483$ and can vary depending on the specific task.

For neural network modeling in order to ensure the correct learning of the recurrent neural network, the series must be normalized or standardized. Here, we apply the normalization of the temperature series within the $[0,1]$ range according to the formula:

$$\forall y^i \in \{\boldsymbol{Y}_T, \boldsymbol{Y}_V\}: \; y^i = \frac{y^i - \min(\boldsymbol{Y}_T)}{\max(\boldsymbol{Y}_T) - \min(\boldsymbol{Y}_T)}. \tag{1}$$

The normalization of the training set $\boldsymbol{Y}_T$ takes place before the normalization of the validation set $\boldsymbol{Y}_V$, during which the determination of the values $\min(\boldsymbol{Y}_T)$ and $\max(\boldsymbol{Y}_T)$, which should be used in (1) for normalization to avoid the problem of data leakage.

After normalization of the temperature series by (1), the normalized datasets of temperature data are converted to the matrix forms (regression-to-supervised learning formulation) as follows:

$$\boldsymbol{X}_{train} = \begin{bmatrix} y^1 & \ldots & y^v \\ y^2 & \ldots & y^{v+1} \\ \ldots & \ldots & \ldots \end{bmatrix}, \; \boldsymbol{Y}_{train} = \begin{bmatrix} y^{v+1} & \ldots & y^{v+p} \\ y^{v+2} & \ldots & y^{v+p+1} \\ \ldots & \ldots & \ldots \end{bmatrix}, \tag{2}$$

where $v$ is the length of the predictor vector, $p$ is the length of the target vector. Equation (2) also has to be applied to the validation of the set $\boldsymbol{Y}_V$. The values of parameters $v$ and $p$ can be estimated based on statistical analysis of series or dynamically in the context of selection of hyperparameters and configurations of the neural network.

In this study, for demonstration purposes the values of parameters $v$ and $p$ equal to 12 and 3, respectively (12 previous values are used to predict the next 3). Such a configuration corresponds to the multi-step prediction problem.

The last step of data preparation for the usage in recurrent neural networks (RNN) is the 3D transformation of $\boldsymbol{X}_{train}$ and $\boldsymbol{X}_{valid}$ feature matrices which can be performed using the next scheme:

$$\boldsymbol{X}_{train} = \left(y_{ij}\right)_{r \times v} \Rightarrow \boldsymbol{X}^{3D}_{train} = \left(y_{ijk}\right)_{r \times v \times k}, \tag{3}$$

where $r$ is the number of rows, $k$ is the third dimension equal to 1 (univariate time series case). Hence, the resulting dimensions of $\boldsymbol{X}^{3D}_{train}$ and $\boldsymbol{X}^{3D}_{valid}$ are $(1173,12,1)$ and $(282,12,1)$.

**Models development**

After the preparation of training and validation sets the hyperparameters and architecture of RNN should be established. Since it is assumed that the designed RNN is going to be

deployed on a microcontroller, and the one-dimensional temperature time series does not contain seasonal components and trends and has not been differentiated, we limit ourselves to single-layer RNN (vanilla type if RNN) and investigate modeling accuracy within the multi-step temperature prediction problem.

The hyperparameters $\boldsymbol{H}$ of RNN for this problem can be determined using the grid search, random search, Bayesian optimization or genetic algorithm [11, 12]. For the sake of simplicity the next hyperparameters have been chosen to be optimized using the genetic algorithm: a) the length of the hidden state vector $h_l$, b) the batch size of data for training $batch$, c) the learning rate $\alpha$. The rest of hyperparameters and configuration have been defined as follows: the number of epochs equals to 40, the activation function have been selected as $\tanh()$, optimization algorithm – Adam.

In this section, the configuration parameters were determined on the basis of a genetic algorithm, where the objective function has been set as

$$\Phi(\boldsymbol{H}) = \frac{1}{N_v}\sum_{i=1}^{N_v}\left(\text{model}\left(\boldsymbol{x}^i, \boldsymbol{H}\right) - y^i\right)^2, \tag{4}$$

where $N_v$ is the number of samples in validation set, $\boldsymbol{x}^i \in \boldsymbol{X}_{valid}^{3D}$, $\text{model}\left(\boldsymbol{x}^i, \boldsymbol{H}\right)$ defines the result of model's prediction, $y^i \in \boldsymbol{Y}_{valid}$.

Fig. 2 shows the structure of the general RNN for modeling the temperature time series. In fig. 2 $RNN_1$ is the recurrent layer (which is represented by LSTM or GRU cell types), the input of which is fed to three-dimensional input tensors. This layer can consist of unidirectional or bidirectional connections between cells. Bidirectional RNN consists of two sub-layers, one of which implements data propagation in one direction, since the second layer implements data propagation in the opposite direction. In [13] it is shown that in some time series modeling problems, bidirectional architectures demonstrate higher prediction accuracy, so both variants of the RNN architecture are considered here.

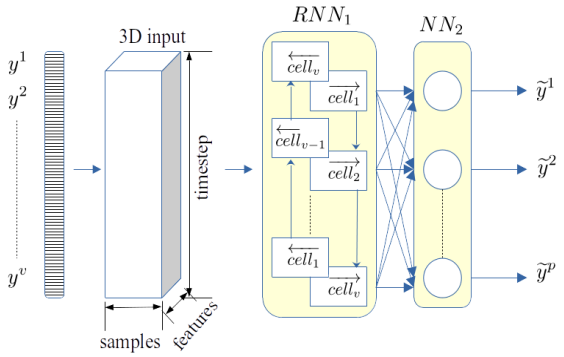


Fig 2. Architecture of proposed RNN

The output values of this layer are fed into the input of the fully connected layer $NN_2$ in order to return a vector of fixed size. The resulting vector $\tilde{\boldsymbol{y}} = [\tilde{y}^1, \tilde{y}^2, \ldots, \tilde{y}^p]$ is the prediction made by the developed model.

For the numerical experiments it has been chosen two distinctive RNN cell types: LSTM and GRU, which are efficiently being used in many of sequence learning problems.

**Results and discussion**

Firstly, let's consider the application of LSTM cell types in the scope of architecture (fig. 1). As a result of conducted numerical experiments (multiple starts of model training) the following values of hyperparameters for unidirectional and bidirectional LSTM neural network have been obtained: $h_l = (97,136)$, $batch = (64,64)$, $\alpha = (0.01, 0.01)$.

In fig. 3 it is presented the results of training both configurations of the LSTM neural networks: unidirectional (fig. 3(1)) and bidirectional (fig. 3(2)), where the bidirectional architecture allows one to obtain a relatively smaller value of the mean square error (MSE) on the validation data set.

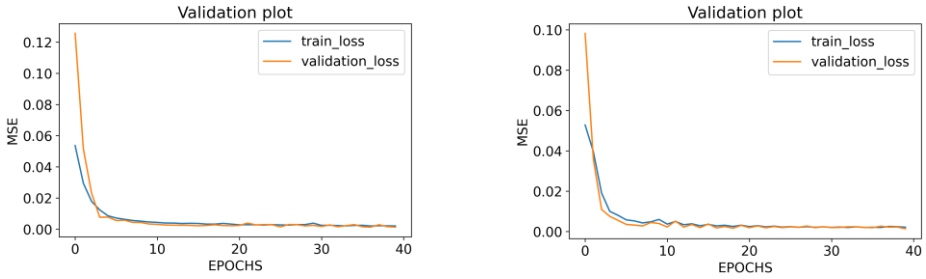


Fig. 3. MSE plots during training of LSTMs

In fig 4 it is shown the distributions of the mean squared error for unidirectional (fig. 4(1)) and bidirectional (fig. 4(2)) LSTM neural networks, where bidirectional LSTM produces smaller error values for samples from the validation set. These distributions have been calculated for each sample in $\boldsymbol{X}_{valid}$.

Numerical experiments conducted for both unidirectional and bidirectional GRU types of RNN allowed to obtain such values of hyperparameters: $h_l = (59,79)$, $batch = (128,64)$, $\alpha = (0.1, 0.01)$.

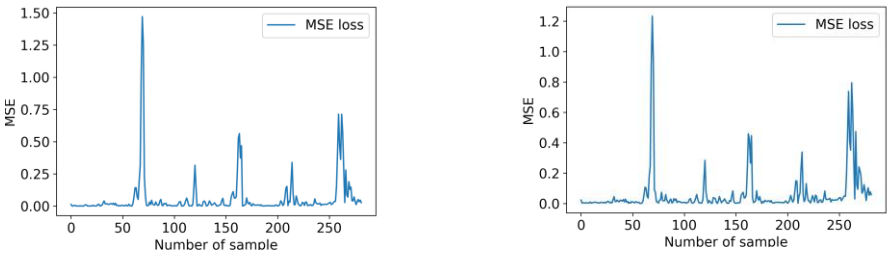


Fig. 4. MSE samples values for LSTMs

For these neural networks, the mean squared errors on validation set during the training process have been shown the following distribution (fig. 5). Here one can see that the behavior of bidirectional GRU during training phase is different compared to bi-directional LSTM, i.e., unidirectional GRU performs better from the starting epochs.

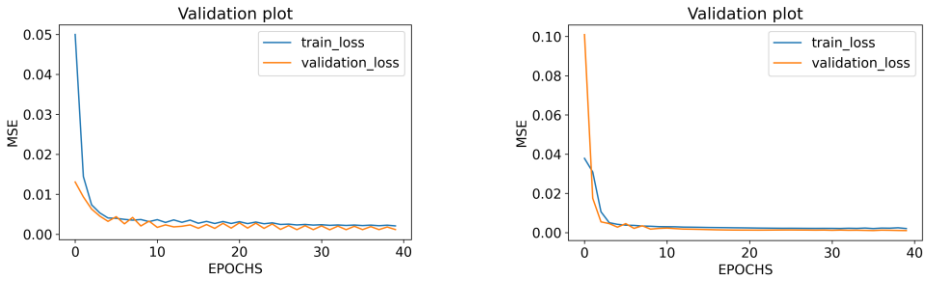


Fig. 5. MSE plots during training of GRUs

Fig 6 presents the distributions of the mean squared error for unidirectional (fig. 6(1)) and bidirectional (fig. 6(2)) GRU neural networks. Here, the bidirectional GRU produces smaller errors for the samples from validation set compared to the unidirectional GRU.

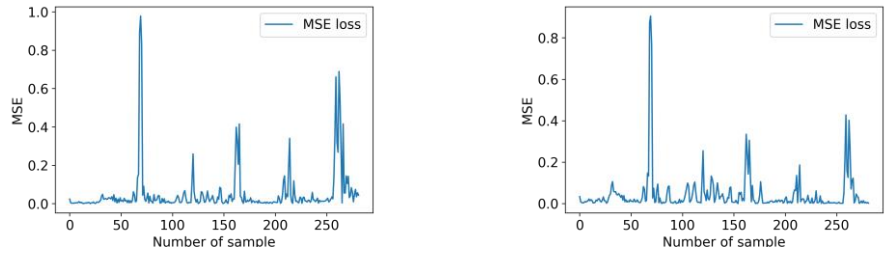


Fig. 6. MSE samples values for GRUs

On the basis of the provided results there are a few clear conclusions about the application of RNN family models for the prediction task: 1) it is possible to efficiently use pure LSTM and GRU predictive model even with the single recurrent layer architecture without additional feature engineering process; 2) such data as the monthly temperature measurements can be accurately modeled by the unidirectional and bidirectional LSTM/GRU networks; 3) despite the bidirectional architecture performs more accurate, the computational costs are bigger, especially for low-powered edge devices, hence unidirectional neural network is preferable here; 4) well configured GRU type of recurrent layer can produce even better results than LSTM for such data, nevertheless this inference should be investigated for other sensory data; 5) genetic optimization is also suitable for hyperparameters calculation due to the ability of parallel computations and good convergence; 6) for the microcontroller deployment it is desirable to select the simpler model with equal or bigger accuracy, so it is recommended to use unidirectional GRU model, if its performance equals to or better than LSTM.

**Scheme for neuro-controller system**

The designed and studied RNN models can be easily deployed on a STM32 microcontroller with the X-CUBE-AI support. The proposed scheme of smart home neuro-controller based on STM32 is depicted in fig. 7.

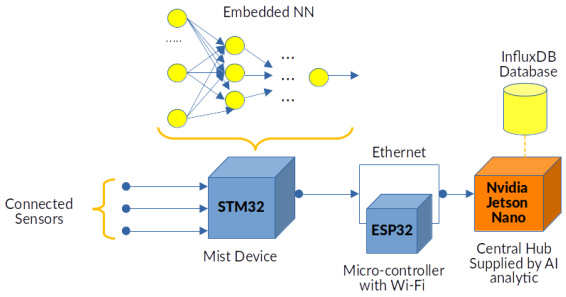


Fig. 7. Smart home neuro-controller scheme

STM32 microcontroller which is used as a main mist device with the deployed (embedded) trained neural network is connected to a bunch of sensors (temperature, humidity, energy). These sensors produce the multivariate time series data which can be: 1) input for the short-term forecasting or anomaly detection routine or b) processed by neuro-controller and stored into InfluxDB database installed on main smart home hub using the Nvidia Jetson Nano microcomputer. The processed data are transferred to hub via Ethernet or the additional ESP32 microcontroller equipped with Wi-Fi module. This scheme serves as a basis for scientific research and has the prospect of implementation as a final digital solution for a smart home.

**Conclusions and future work**

In this study, it has been considered a problem of low-complexity recurrent neural network design for the edge computations performed by a STM32 microcontroller. Real temperature data collected via RPi3 based server and DS18B20 sensors have been used and pre-processed. Indoor temperature time series has been queried as the model data to develop predictive unidirectional and bidirectional LSTM/GRU models and to compare their accuracy. Also, for optimization of the hyperparameters a genetic algorithm has been applied. This choice was made for reasons of its parallel computations ability (to use GPU CUDA library) and flexibility of settings. The neural network trainings have been conducted using a Nvidia RTX 2080 Super GPU. The obtained results demonstrate that unidirectional GRU architecture is the optimal choice for modeling the considered temperature data and can be accurately applied to the other univariate sensory smart home measurements.

In future work the multivariate models will be designed and analyzed as well as the process of model quantization for STM32 microcontroller. Also, it is necessary to investigate hybrid models like CNN-GRU/LSTM, ConvGRU/LSTM, encoder-decoder etc. Feature engineering procedure can enhance the performance of the models and should be studied as well.

## References

[1] Deep Learning for Edge Computing Applications: A State-of-the-Art Survey / [F. Wang, M. Zhang, X. Wang та ін.]. // IEEE Access. – 2020. – №8. – P. 58322 – 58336. DOI: 10.1109/ACCESS.2020.2982411.

[2] Edge computing use case examples [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://stlpartners.com/edge_computing/10-edge-computing-use-case-examples/.

[3] *Mao J*. Application of learning algorithms in smart home IoT system security / J. Mao, Q. Lin, J. Bian. // Mathematical Foundations of Computing. – 2018. – №1. – P. 63–76. DOI: 10.3934/mfc.2018004.

[4] Smart Home System Based on Deep Learning Algorithm / Y. Peng, J. Peng, J. Li, L. Yu. // Journal of Physics: Conference Series. – 2019. – №1187. – P. 032086. DOI: 10.1088/1742-6596/1187/3/032086.

[5] Design of a Prototype Neural Network for Smart Homes and Energy Efficiency / T.Teich, F. Roessler, D. Kretz, S. Franke. // Procedia Engineering. – 2014. – №69. – P. 603–608. DOI: 10.1016/j.proeng.2014.03.032.

[6] *Chakraborty T*. Home automation using edge computing and Internet of Things / T. Chakraborty, S. Datta. // IEEE International Symposium on Consumer Electronics (ISCE). – 2017. – P. 47–49. DOI: 10.1109/ISCE.2017.8355544.

[7] Design of Smart Home System Based on Collaborative Edge Computing and Cloud Computing / Q.Ma, H. Huang, W. Zhang, M. Hua. // Algorithms and Architectures for Parallel Processing. – 2020. – P. 355–366. DOI: 10.1007/978-3-030-60248-2_24.

[8] Self-Learning Algorithm to Predict Indoor Temperature and Cooling Demand from Smart WiFi Thermostat in a Residential Building / [K. Huang, K. Hallinan, R. Lou та ін.]. // Sustainability. – 2020. – №12. – P. 7110–7124. DOI: https://doi.org/10.3390/su12177110.

[9] Deep Learning architecture for temperature forecasting in an IoT LoRa based system / [I. Ouahab, B. Abdelhakim, A. Astito та ін.]. // Conference: the 2nd International Conference on Networking, Information Systems & Security (NISS19). – 2019. – P. 1–6. DOI: 10.1145/3320326.3320375.

[10] AI expansion pack for STM32CubeMX [Електронний ресурс] – Режим доступу до ресурсу: https://www.st.com/en/embedded-software/x-cube-ai.html.

[11] *Jin H*. Auto-Keras: An Efficient Neural Architecture Search System / H. Jin, Q. Song, X. Hu. // ACM SIGKDD International Conference. – 2019. – P. 1946–1956. DOI: 10.1145/3292500.3330648.

[12] Embedding Sequence Model in STM32 Based Neuro-Controller / [O. Sinkevych, Y. Boyko, O. Rechynskyi та ін.]. // 2021 IEEE 12th International Conference on Electronics and Information Technologies (ELIT). – 2021. – P. 113–118. DOI: 10.1109/ELIT53502.2021.9501132.

[13] *Siami-Namini S*. The Performance of LSTM and BiLSTM in Forecasting Time Series / S. Siami-Namini, N. Tavakoli, A. Siami Namin. // 2019 IEEE International Conference on Big Data (Big Data). – 2019. – P. 3285–3292. DOI: https://doi.org/10.1109/BigData47090.2019.9005997.

# РЕКУРЕНТНІ НЕЙРОННІ МЕРЕЖІ МАЛОЇ СКЛАДНОСТІ ДЛЯ ГРАНИЧНИХ ОБЧИСЛЕНЬ

**О. Сінькевич**

*Львівський національний університет імені Івана Франка,*
*вул. Драгоманова, 50, 79005 Львів, Україна*
oleh.sinkevych@lnu.edu.ua

Дана стаття присвячена розробці рекурентних нейронних мереж для програмного забезпечення інтелектуальних пристроїв, які можуть функціонувати на основі штучного інтелекту, таких як мікроконтролери та інші системи граничних обчислень. Через незначну обчислювальну потужність таких пристроїв метою дослідження є розробка та аналіз прогностичних моделей низької складності для прогнозування сенсорних часових рядів на прикладі одновимірного часового ряду, а саме виміряних температур у приміщенні. Розроблені моделі можуть легко розгортатися на мікроконтролері сімейства STM32 за допомогою пакету розширення X-CUBE-AI. Температурні дані були зібрані з використанням датчиків температури DS18B20, що підключались до Raspberry Pi 3 зі встановленою базою даних InfluxDB. У статті наведений опис підготовки та перетворення даних у вигляді часового ряду для подальшого використання у процесі тренування рекурентних нейронних мереж. Розглянута низка конфігурацій моделей на основі простих одношарових архітектур LSTM та GRU з однонаправленими та двонаправленими зв'язками між відповідними комірками нейронних мереж. Для розрахунку оптимального набору гіперпараметрів для чотирьох архітектур нейронних мереж було реалізовано генетичний алгоритм. Оптимізація гіперпараметрів відбувалась на базі мінімізації функції середньоквадратичної помилки на зразках з валідаційної вибірки, що вимагало багаторазовий процес навчання нейронних мереж для кожного набору гіперпараметрів з простора пошуку. Наведені результати чисельних експериментів, проведених для конфігурацій моделей з однонаправленими та двонаправленими зв'язками між комірками свідчать про можливість використання такого підходу для побудови прогностичних моделей для одновимірних сенсорних часових рядів. Аналіз отриманих результатів дає змогу визначати оптимальну прогностичну модель низької складності для розгортання на мікроконтролері. Також запропоновано схему розгортання розроблених моделей на мікроконтролері STM32, об'єднаному з високопродуктивним хабом на базі мікрокомп'ютера Nvidia Jetson Nano з графічним прискорювачем, що може ефективно застосовуватися для тренування простих нейронних мереж.

*Ключові слова*: граничні обчислення, рекурентні нейронні мережі, часові ряди, генетичний алгоритм.

*Стаття: надійшла до редакції* *30.11.2021,*
*доопрацьована* *30.11.2021,*
*прийнята до друку* *01.12.2021*