Comparative Analysis of Neural Networking and Regression Models for Time Series Forecasting

S. Sholtanyuk*

Belarusian State University, FAMCS, the Department of Computer Applications and Systems, Minsk, 220030 Belarus
**e-mail: SSholtanyuk@bsu.by*

Abstract—Applicability of neural nets in time series forecasting has been considered and researched. For this, training of fully connected and recurrent neural networks on various time series with preliminary selection of optimal hyperparameters (optimization algorithm, amount of neurons on hidden layers, amount of epochs during training) has been performed. Comparative analysis of received neural networking forecasting models with each other and regression models has been performed. Conditions, affecting on accuracy and stability of results of the neural networks, have been revealed.

Keywords: neural network, perceptron, LSTM network, training of neural network, hyperparameters, forecasting accuracy and stability, MAE, linear regression, autoregression, ordinary least squares

DOI: 10.1134/S1054661820010137

INTRODUCTION

Nowadays there are various classes of forecasting models and methods. Forecasting models can be divided into two categories. In statistical models, dependence of value of time series on previous values of the same time series is expressed by a formula. In structural models, this dependence is set by some structure [1]. These models are used, when finding the dependence analytically seems to be difficult.

Neural networks are attributed to structural category of forecasting models. In these models, unlike in statistical ones, not only initial data affect accuracy and stability of forecasting results, but several structural characteristics of neural networks (such as amount of neurons and connections among them, activation functions, training duration, etc.) also do. This is a disadvantage of neural networks, which complicates their using in forecasting. That's why comparing of statistical and structural approaches to time series forecasting is of interest.

FEATURES OF NEURAL NETWORK MODELS FOR TIME SERIES ANALYSIS AND FORECASTING

The aim of this research is building and researching two neural networks, analyzing various time series. The following objectives have been set:

– setting of optimal values of neuronet hyperparameters;

– comparing of accuracy results of neural networks and regression models for time series with linear and non-linear trends.

Two neural networks are the object of the researching: perceptron and recurrent neural network. Perceptron consists of three sequential layers. The choice of this architecture is due to the fact that, on the one hand, neural networks with one hidden layer are universal approximators [2], and on the other hand, using of deep neural networks doesn't improve results in comparison with neuronets with 1–2 hidden layers [3]. Input layer consists of p neurons, to which p consecutive elements of the considered time series are fed. After the input layer, the network has a hidden one, consisting of arbitrary amount of neurons. On this layer, activation function ReLU (rectified linear unit) is acting according to the following formula:

$$f(x) = \max(0, x). \quad (1)$$

Output layer consists of the only neuron, on which prediction of value of time series from p previous values (lags) is calculated. This neural network is fully connected, i.e. all neurons of each layer (except the output one) are connected with all neurons of the next layer by directed connections (synapses) with own weights. Besides, input data are preliminarily normalized according to the following formula:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (2)$$

where x – value before normalizing, $x' \in [0, 1]$ – value after normalizing, $x_{\min}$ and $x_{\max}$ – minimal and maximal values of the considered time series respectively. The inverse of (2) is applied to the result obtained on the output layer. As a result, forecasted value of the time series is received.

Received September 29, 2019; revised October 14, 2019; accepted October 21, 2019

As the perceptron, the recurrent neural network consists of three layers. Input layer contains p neurons, output one – single neuron, and all neurons on input and hidden layers are connected with all neurons on hidden and output layers respectively. Unlike the perceptron, the recurrent neuronet hidden layer contains LSTM neurons, presenting a kind of recurrent neurons, which can use the value on their outputs on the next epoch, as well as remember input information [4]. On hidden layer, two activation functions are used: one function for data from input layer, and the other one for data, obtained on hidden layer and fed recurrently on input of LSTM neurons. In considered neural network, activation functions ReLU (1) and hard sigmoid, acting according to the following formula:

$$f(x) = \begin{cases} 0, & x < -2.5, \\ 0.2x + 0.5, & -2.5 < x < 2.5, \\ 1, & x > 2.5, \end{cases}$$

are used respectively.

As in the perceptron, input data are normalized according to formula (2), and then are fed to input of the recurrent neural network, producing a forecasting result, to which inverse transformation must be applied.

HYPERPARAMETERS EVALUATION SCHEME AND ANALYSIS OF THE NEURAL NETWORKS PERFORMANCE

Algorithm that implements the neural networks has been implemented on R language with using Keras library [5, 6]. Building of neural networking models for each series is performed with training of the neural networks.

Training of the neural networks occurs as follows. Initially, $n-p$ training instances, where n – length of the considered time series, are formed. Each of them is a vector of p consecutive elements of time series, to which the next, $(p+1)$th element of the same series, is mapped. After that, the neural networks are trained on these instances for several epochs. Thus, the neural networks have the following hyperparameters, which should be evaluated:

– amount of neurons on the input layer p (which is also indicates maximal lag order, with which forecasting of the series will be performed);

– amount of neurons on the hidden layer;

– amount of training epochs.

Neural networks optimization algorithm, i.e. algorithm of synapses weights recalculation on each epoch, has been selected by example of modeled time series with deterministic quadratic trend:

$$y_t = a + bt + ct^2 + \xi_t, \quad t = \overline{1,n}, \tag{3}$$

where y_t – value of the time series in moment t, a, b and c – trend parameters, ξ_t, $t = \overline{1,n}$ – random values, which are normally distributed with zero mean and preset variance σ^2.

After the selection of optimization algorithm, hyperparameters evaluation was done on an example of the series (3). Training of the neural networks with various sets of values of hyperparameters was performed. For each of these sets training was repeated Q times. After each training, mean absolute error (MAE) was calculated according to the following formula:

$$e_q = \frac{1}{n-p}\sum_{t=p+1}^{n} |y_t - \hat{y}_{q,t}|, \quad q = \overline{1,Q}, \tag{4}$$

where $\hat{y}_{q,t}$, $t = \overline{p+1,n}$, $q = \overline{1,Q}$, – forecasted values of the time series y_t. For received values e_q, $q = \overline{1,Q}$, their mean value and standard deviation were calculated. After comparison of these characteristics, the set of optimal values of hyperparameters has been selected.

In Table 1, results of the perceptron and LSTM network performance, when using different optimization algorithms for the time series (3), are presented (the experiment has been repeated Q times for each optimizator).

As Table 1 shows, for the majority of optimization algorithms implemented in Keras library, standard deviation of MAEs exceeds their mean value, which indicates instability of both neural networks in these cases. That's why adaptive gradient method, or AdaGrad, has been chosen as optimization algorithm for training both neuronets, because it has demonstrated the best stability in comparison to other optimizators.

Given method is a modification of the method of stochastic gradient descent (SGD), which is widely used for training of neural networks. Unlike SGD, where the length of a gradient vector, along which weights of synapses change, doesn't depend on input data, in AdaGrad, frequency of input instances is taken into account to take notice of infrequently occurring values of time series, which can impact on the desired models [7]. If in stochastic gradient descent, parameters of optimizator θ_i, $i = \overline{1,\nu}$, are recalculated according to formula

$$\theta_{\tau+1,i} = \theta_{\tau,i} - \eta \cdot \nabla_\theta J(\theta_{\tau,i}),$$

where $\theta_{\tau,i}$ – value of parameter in the beginning of τth epoch, η – hyperparameter, called "general learning rate", than when using adaptive gradient, the following transform is taking place:

$$\theta_{\tau+1,i} = \theta_{\tau,i} - \frac{\eta}{\sqrt{G_{\tau,ii} + \varepsilon}} \cdot \nabla_\theta J(\theta_{\tau,i}),$$

where G_τ – diagonal matrix of the order of ν, in which ith diagonal element equals to the sum of the squares of the partial derivatives of the function J with respect to parameter θ_i, which are received on the first τ

Table 1. Comparison of the neural networks performance on an example of series $y_t = a + bt + ct^2 + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$. Results for $a = 21.4$, $b = 3.2$, $c = 0.07$, $n = 1109$, $\sigma = 15$, $Q = 25$ are presented

Name of optimizator	Perceptron		LSTM network	
	Mean of MAEs	Standard deviation of MAEs	Mean of MAEs	Standard deviation of MAEs
adadelta	191.79	265.55	129.53	139.62
adagrad	19.98	5.63	23.93	12.35
adam	107.89	154.34	298.10	516.45
adamax	184.50	349.35	92.85	89.19
nadam	83.21	85.46	137.06	212.40
rmsprop	462.29	498.85	800.32	780.07
sgd	26.20	9.97	101.78	31.17

Table 2. Accuracy of the neural networks performance on an example of the series $y_t = a + bt + ct^2 + \xi_t$, $t = \overline{1,n}$

Amount of neurons on the input layer p	Amount of neurons on the hidden layer	Perceptron		LSTM network	
		Mean of MAEs	Standard deviations of MAEs	Mean of MAEs	Standard deviations of MAEs
5	10	70.33	90.39	3421.44	4274.61
5	50	26.79	17.75	145.20	132.09
5	100	19.26	4.51	21.23	11.24
5	250	22.32	17.37	19.17	1.84
5	500	19.11	8.24	20.90	2.81
10	10	155.01	493.26	2244.20	1511.83
10	50	30.98	23.24	257.16	234.40
10	100	25.18	13.10	128.21	153.57
10	250	22.94	11.12	48.02	45.90
10	500	21.92	10.56	47.21	62.31

epochs, ε – smoothing variable that avoids division by zero (usually on the order of 1×10^{-8}) [8].

After hyperparameters evaluation, the neural networks have been applied for building of forecasts for the following series with deterministic trends:

$$y_t = a + bt + \xi_t,$$
$$y_t = a + bt + c\sin t + \xi_t,$$
$$y_t = a + bt + ct^2 + \xi_t,$$
$$y_t = a\sin t + bt^2 + \xi_t,$$
$$y_t = a + b\sin t + \xi_t,$$

where a, b, c – trend parameters, $\xi_t \in N(0, \sigma^2)$. Synapses weights initialization for the given series was done randomly. After training of the neural network, MAE was calculated according to formula (4). This procedure was repeated Q times for each time series.

For each time series a linear regression model

$$y_t = \alpha + \beta t + \xi_t, \quad t = \overline{1,n} \qquad (5)$$

was built by ordinary least squares, as well as autoregression model:

$$y_t = \beta_1 y_{t-1} + \ldots + \beta_p y_{t-p} + \xi_t, \quad t = \overline{p+1,n}. \qquad (6)$$

For both models MAEs were calculated with formula (4) (for model (5) $p = 0$ was considered). These MAEs were compared with statistical characteristics of MAEs received for the perceptron and LSTM network.

RESEARCH RESULTS

Examples of the neural networks performance with various values of hyperparameters are presented in Table 2.

The perceptron with ten neurons on its hidden layer is less accurate and stable, because mean and standard deviation of MAEs in this occurrence exceed values of the same characteristics with other amount of neurons on the hidden layer. In other occurrences, connection between amount of neurons on the input

Table 3. Accuracy of the perceptron performance on an example of the series $y_t = a + bt + ct^2 + \xi_t$, $t = \overline{1,n}$

Length of series n	Amount of neurons on the hidden layer	Mean of MAEs	Standard deviation of MAEs
1000	100	21.72	6.95
1000	1000	24.37	16.29
1000	5000	19.82	9.00
1000	20000	24.70	15.14
1000	40000	23.25	15.69
5000	100	92.73	40.47
5000	1000	36.11	12.79
5000	5000	30.26	15.85
5000	20000	47.51	48.85
5000	40000	50.73	39.68

and on the hidden layers is not clearly visible. And for the recurrent neural network, optimal amount of neurons on hidden layer is about 100–500. If there are fewer neurons, less stable results are obtained, and if there are more neurons, training of LSTM network takes longer without significant improvement of results. Among all considered results, the best ones received, when amount of neurons on the hidden layer is 100, $p \in \{3, 5, 7\}$ (values of $p \in \{3, 5, 7, 10, 15\}$, amount of neurons on the hidden layer from set {10, 50, 100, 250, 500, 1000} for the perceptron and {10, 50, 100, 250, 500} for the LSTM network have been considered).

Dependence of accuracy of the perceptron performance on amount of neurons on the hidden layers for time series of various lengths has been also considered. Examples of results are presented in Table 3.

The best accuracy and stability results have been received for the following pairs of value of n and amount of neurons: (1000, 5000), (1000, 10000), (2500, 5000), (2500, 10000), (5000, 5000), (5000, 10000), (10000, 2500), (10000, 5000) (series of length of $n \in \{1000, 2500, 5000, 10000\}$ and amount of neurons from set {100, 500, 1000, 5000, 10000, 20000, 40000} were considered). When $n = 1000$, approximately equal results have been received for various amounts of neurons on the hidden layer. For others values of n, the neural network has demonstrated the best accuracy and stability with 2500–10000 neurons.

On a computer with a processor Intel® Core™ i5-3230M CPU 2.60 GHz, average time of the perceptron training for series with length $n = 1000$ – 9 seconds per epoch, with $n = 10000$ – 82 seconds, and it doesn't depend on amount of neurons. Training was performed for 5–10 epochs, because with further training accuracy of the perceptron results is insignificantly changed (Figs. 1, 2).

Figures 1, 2 show that significant improvement of the results for various time series is reached after the second epoch. For example, for time series with quadratic trend (Fig. 1) MAE of normalized data has decreased with comparison to the first epoch. With

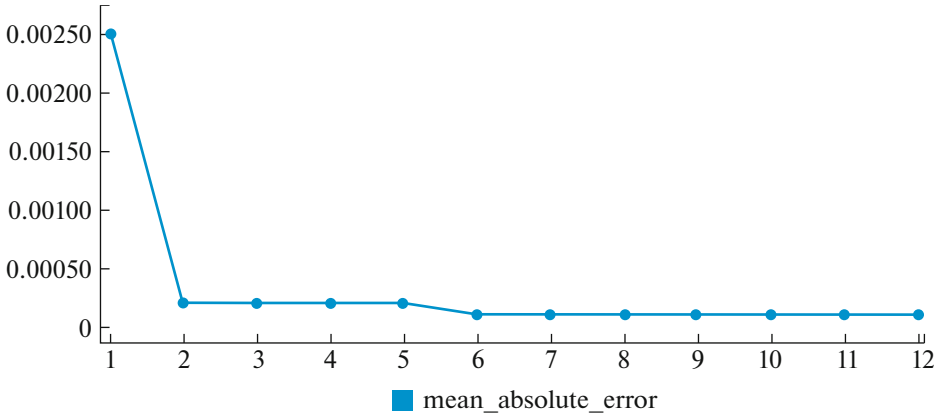

Fig. 1. Dynamics of mean absolute errors for normalized data after each epoch during the perceptron training for forecasting of series $y_t = a + bt + ct^2 + \xi_t$.

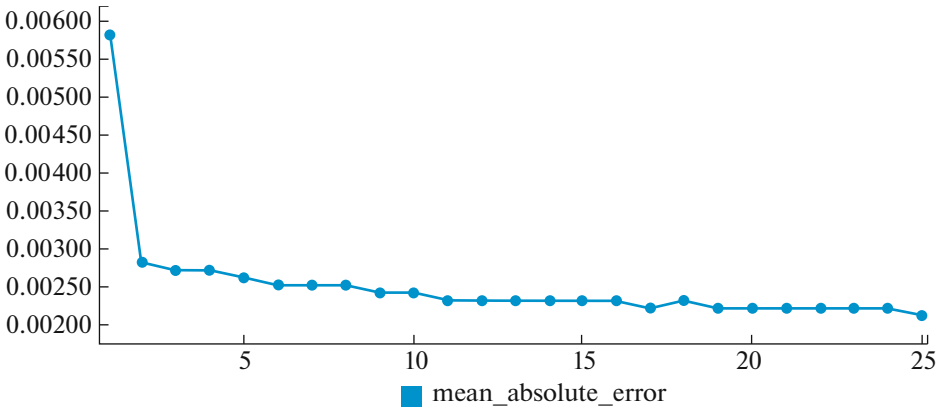

Fig. 2. Dynamics of mean absolute errors for normalized data, when forecasting series $y_t = a + bt + c\sin t + \xi_t$ with perceptron.

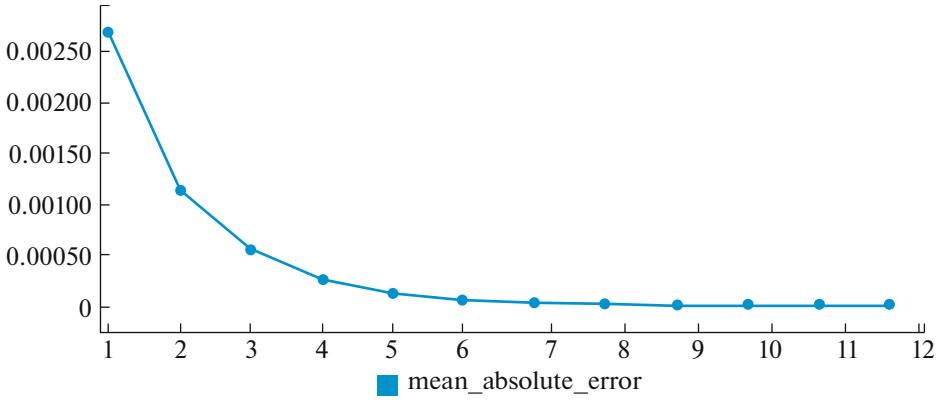

Fig. 3. Dynamics of mean absolute errors for normalized data, when forecasting series $y_t = a + bt + ct^2 + \xi_t$ with LSTM-net with 100 neurons on hidden layer.

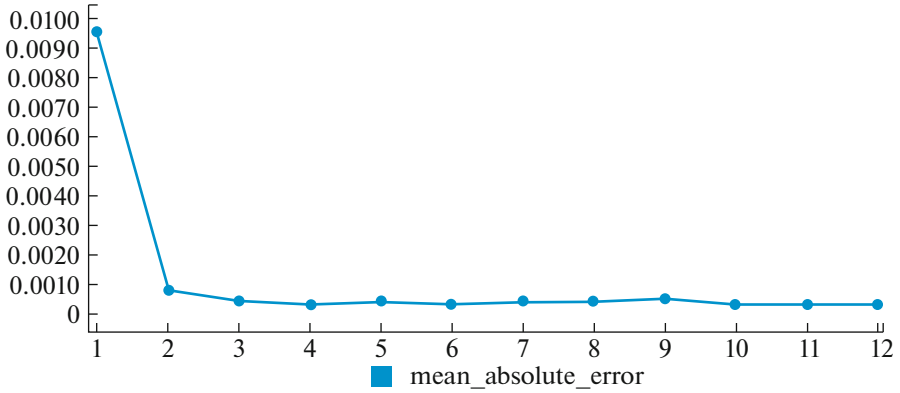

Fig. 4. Dynamics of mean absolute errors for normalized data, when forecasting series $y_t = a + bt + ct^2 + \xi_t$ with LSTM-net with 500 neurons on hidden layer.

further training MAE either remains almost unchanged (Fig. 1) or slowly decreases (for example, for series $y_t = a + bt + c\sin t + \xi_t$ (Fig. 2) MAE after the 12th epoch was 2.3×10^{-3}, and after the 24th epoch – 2.2×10^{-3}, i.e. after increasing the amount of epochs by 2 times the results increased by $23/22 \approx 1.045$ times).

Analysis of the recurrent neural network in forecasting time series with length $n = 1000$ has also been performed. Average training time of LSTM network is

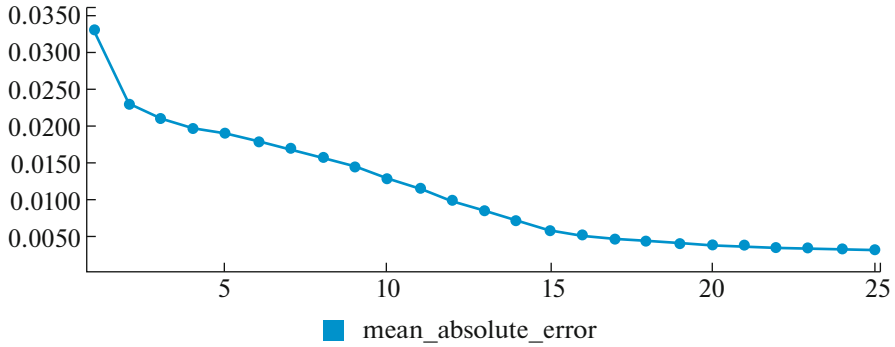

Fig. 5. Dynamics of mean absolute errors for normalized data, when forecasting series $y_t = a + bt + c\sin t + \xi_t$ with LSTM-net with 100 neurons on hidden layer.

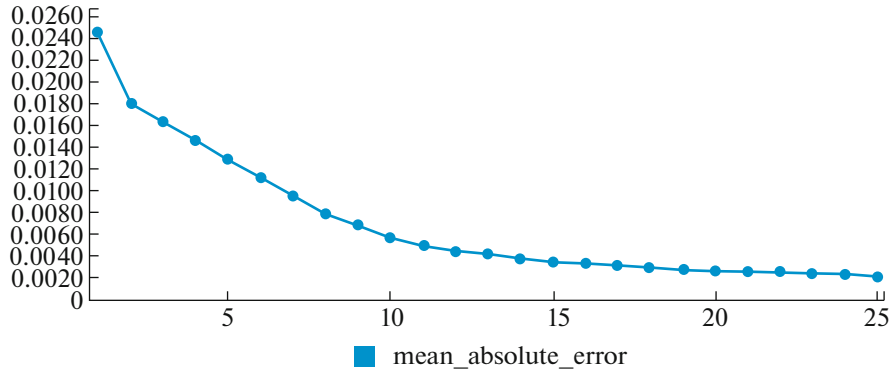

Fig. 6. Dynamics of mean absolute errors for normalized data, when forecasting series $y_t = a + bt + c\sin t + \xi_t$ with LSTM-net with 500 neurons on hidden layer.

9 s, when there are 100 neurons on hidden layer, 14 s, when there are 250 neurons, and 50 s, when there are 500 neurons (Figs. 3–6).

With 100 neurons on hidden layer of the LSTM network training is as long as training of the perceptron, but such net is less accurate compared to the fully-connected neuronet. On the other hand, with using 500 neurons on hidden layer, one can obtain more accurate results, but such LSTM network will be trained significantly longer. Thus, recommended amount of epochs for the recurrent neural network will depend not only on length of considered time series, but also on architecture of the neural network itself.

Further, dependence of accuracy of the neural networks performance on parameters a, b, c, σ with $p = 7$ and 100 neurons on the hidden layer was considered. There were 7 epochs during training of the perceptron, and 10 epochs during training of the LSTM network. Received characteristics of MAEs and their comparison to MAEs of regression models for series with linear trend are presented in Table 4 (in Tables 4–8, "S.d." stands for "standard deviation", "LR MAE" and "AR MAE" denote MAEs obtained with linear regression and autoregression respectively).

For all series with linear trends, both neural networks are more accurate than autoregression model (6), but less accurate than linear regression (5). Besides, mean absolute error of the model (5) doesn't depend on free term and coefficient, which set linear trend of series, while when modeling with both the perceptron and the LSTM network, with the increasing of b the accuracy and stability of the results get worse. Variance σ^2 also impacts on the accuracy and stability of the results.

In Tables 5–8, accuracy results and comparison of the models for series with non-linear trends are presented.

When forecasting time series with linear trend and sinusoidal oscillations, variance σ^2 of random values ξ_t plays an important role: its increasing involves increasing of MAEs with all the models. The recurrent neural network gives more stable results than the perceptron, as evidenced by the values of standard deviations of MAEs.

When forecasting series with quadratic trend, trend parameter c also impacts on performance of the models. By increasing of this parameter accuracy and stability of the neural networking models get worse, as

Table 4. Comparison of accuracy of neural networking models with regression models for series $y_t = a + bt + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$

a	b	σ	Perceptron		LSTM network		LR MAE	AR MAE
			Mean of MAEs	S.d. of MAEs	Mean of MAEs	S.d. of MAEs		
5	1	10	10.06	6.00	8.99	0.43	8.07	10.59
5	5	10	10.84	11.32	9.49	1.26	8.13	12.41
5	10	10	10.93	13.64	10.84	5.86	7.94	15.78
5	50	10	13.42	19.43	22.68	13.25	8.01	55.51
5	100	10	16.39	35.33	52.14	45.73	7.56	108.95
10	1	10	9.51	5.03	8.79	0.45	7.89	10.68
10	5	10	11.09	12.38	10.80	3.27	8.26	12.26
10	10	10	12.31	20.22	11.61	4.04	8.38	15.98
10	50	10	16.05	31.38	26.50	18.66	8.16	55.41
10	100	10	19.35	61.89	68.24	60.55	7.84	109.02
1	10	1	1.88	2.09	5.76	4.70	0.76	10.90
1	10	5	6.25	1.78	7.78	3.58	3.98	12.19
1	10	10	12.59	5.60	11.68	2.81	8.12	15.87
1	10	50	53.59	9.64	46.43	4.08	42.12	61.86
1	10	100	101.13	12.89	89.43	6.02	82.10	101.14

Table 5. Comparison of neural networking models with regression models for series $y_t = 1 + t + c\sin t + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$

c	σ	Perceptron		LSTM network		LR MAE	AR MAE
		Mean of MAEs	S.d. of MAEs	Mean of MAEs	S.d. of MAEs		
1	1	1.35	2.26	1.28	0.29	0.97	1.65
1	10	9.46	4.63	9.21	0.70	7.94	10.48
1	50	42.48	2.92	41.27	0.53	38.68	43.28
10	1	3.84	5.38	6.22	0.20	6.41	5.11
10	10	10.88	7.27	10.71	0.54	9.74	11.75
10	50	42.23	7.95	44.95	1.15	38.59	43.21
50	1	1.32	1.19	10.68	3.24	31.81	4.64
50	10	11.47	8.37	16.36	2.85	32.99	12.96
50	50	47.17	8.28	48.68	1.39	50.72	48.03

well as accuracy of regression models. The LSTM network has shown better results in stability in comparison with the perceptron.

For series $y_t = a\sin t + bt^2 + \xi_t$ with increasing of coefficient b MAEs with neural networks and autoregression increase, and with autoregression changing of the results is more substantial. The recurrent neural network has shown better results than the perceptron in all the examples, except $a = 10$, $b = 0.1$. Influence of parameters a and σ is less significant.

For series with no trend and with sinusoidal oscillations, forecasting accuracy exclusively depends on variance σ^2.

On the whole, from Tables 5–8 we can conclude that for series with non-linear trends neural networking models better fit. Series $y_t = 1 + t + c\sin t + \xi_t$ with $c = 1$ and $c = 10$ are exceptions (Table 5). In this occurrence, non-linear component $c\sin t$, which varies from $-c$ to c, has less impact on behavior of the series, than random values ξ_t, which are in interval $(-3\sigma, 3\sigma)$, i.e. from -3 to 3, from -30 to 30 and from -150 to 150 for the suggested examples. That's why the results of the perceptron and the LSTM network are similar to the results with series with linear trend, and neural network works worse than linear regression model (5).

Table 6. Comparison of neural networking models with regression models for series $y_t = 1 + t + ct^2 + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$

c	σ	Perceptron		LSTM network		LR MAE	AR MAE
		Mean of MAEs	S.d. of MAEs	Mean of MAEs	S.d. of MAEs		
0.01	1	1.80	4.43	2.25	1.97	641.55	14.98
0.01	10	12.31	14.69	9.90	3.49	641.38	18.89
0.01	50	47.86	26.46	44.33	1.81	643.20	57.56
0.1	1	13.00	46.00	27.97	29.79	6415.04	140.45
0.1	10	16.08	30.37	22.38	6.13	6414.37	140.97
0.1	50	59.73	76.45	51.62	6.42	6416.26	152.50
1	1	135.08	490.01	225.02	242.08	64149.95	1395.65
1	10	127.08	495.31	319.93	235.33	64150.18	1395.72
1	50	109.20	154.98	219.13	93.98	64150.21	1396.38

Table 7. Comparison of neural networking models with regression models for series $y_t = a\sin t + bt^2 + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$

a	b	σ	Perceptron		LSTM network		LR MAE	AR MAE
			Mean of MAEs	S.d. of MAEs	Mean of MAEs	S.d. of MAEs		
10	0.02	1	27.18	16.07	7.84	1.86	14229.98	28.52
10	0.02	5	24.73	13.86	9.58	5.59	14229.93	29.13
10	0.02	20	35.21	10.54	18.99	1.26	14229.50	36.79
10	0.1	1	31.44	20.96	30.01	39.16	14229.96	139.59
10	0.1	5	28.25	16.36	35.10	40.74	14229.90	139.71
10	0.1	20	35.24	8.68	52.27	98.51	14230.65	141.59
50	0.02	1	44.54	11.53	32.02	3.65	14230.10	40.37
50	0.02	5	49.78	12.01	32.82	5.67	14229.98	40.57
50	0.02	20	51.37	17.44	34.38	1.65	14229.41	45.16
50	0.1	1	55.62	51.46	40.12	15.70	14230.07	142.36
50	0.1	5	49.84	22.22	39.04	9.42	14230.00	142.49
50	0.1	20	54.02	14.69	45.94	26.27	14230.02	144.33

Table 8. Comparison of neural networking models with regression models for series $y_t = a + b\sin t + \xi_t$, $t = \overline{1,n}$, $\xi_t \in N(0, \sigma^2)$

a	b	σ	Perceptron		LSTM network		LR MAE	AR MAE
			Mean of MAEs	S.d. of MAEs	Mean of MAEs	S.d. of MAEs		
−100	10	1	1.14	0.068	1.15	0.06	19.16	0.92
−100	10	5	5.18	0.097	4.78	0.15	19.65	4.40
−100	10	20	18.72	0.274	16.96	0.10	25.86	16.90
−100	50	1	1.11	0.109	1.38	0.23	19.16	1.04
−100	50	5	5.26	0.117	5.87	0.41	19.60	4.48
−100	50	20	20.13	0.166	20.27	0.50	26.19	17.25
100	10	1	1.06	0.046	1.14	0.08	19.08	0.90
100	10	5	4.90	0.083	4.94	0.11	19.59	4.45
100	10	20	19.08	0.194	17.19	0.12	25.54	16.33
100	50	1	1.07	0.076	1.41	0.18	19.16	1.10
100	50	5	5.17	0.085	5.80	0.40	19.77	4.79
100	50	20	19.19	0.310	18.42	0.60	26.13	17.90

Besides, for series $y_t = a\sin t + bt^2 + \xi_t$ with $a = 50$, $b = 0.02$ and $y_t = a + b\sin t + \xi_t$ (Tables 7 and 8) the results of the perceptron are bit worse than ones of autoregression model (6). This is due to the fact that for these series a strongly marked periodicity is inherent, by virtue of which an apparent dependence of value of time series on its past values is taking place, and this dependence can be evaluated with autoregression model. At the same time, results of the LSTM network for series $y_t = a\sin t + bt^2 + \xi_t$ are better than ones of the perceptron and autoregression.

With increasing of variance of random values ξ_t the accuracy of neural networks decreases, and an influence of variance on the stability is not visible. Parameters, setting trend of time series, principally impact on the accuracy of the models, but the stability not always depends on them. For example, in series $y_t = 1 + t + ct^2 + \xi_t$ with increasing of parameter c standard deviation of MAEs also significantly increases (Table 6), but in other cases there is no such an obvious dependence.

CONCLUSIONS

From the research results we can conclude than perceptrons and LSTM networks with single hidden layer can be used in time series forecasting. Selection of values of hyperparameters is important, because it impacts on accuracy and stability of the neural networks. The impact of selection of optimization algorithm during neural networks training was also shown.

Comparative analysis of neural networking models with each other, as well as with linear regression and autoregression, has been performed. Its result is that linear regression better approximates series with linear trends, autoregression and the LSTM network – series with periodicity and the perceptron better works with non-periodical series with non-linear trends. The perceptron training time only depends on length of forecasted time series, and training time for the recurrent neural network depends also on amount of neurons on hidden layer of the network. At the same time, with proper choosing of this amount the LSTM network gives more accurate and stable results than the perceptron, as a rule. Thus, with the considered neural networks it's possible to build a forecast for arbitrary time series with deterministic trend for a limited time.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. I. A. Chuchueva, "The time series forecasting model by maximum likelihood sampling," *Candidate of Science Dissertation* (Bauman Moscow State Technical University, Moscow, 2012) [in Russian].
2. K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," Neural Networks **2** (5), 359–366 (1989).
3. Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," in *Large-Scale Kernel Machines*, Ed. by L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (MIT Press, Cambridge, MA, 2007), pp. 323–362.
4. S. Hochreiter and J. Schmidhuber, "Long-Short Term Memory," Neural Comput. **9** (8), 1735–1780 (1997).
5. R Development Core Team, *R*: *A language and environment for statistical computing*. R Foundation for Statistical Computing (Vienna, Austria, 2018). Available at https://www.R-project.org/.
6. J. J. Allaire and F. Chollet, *Keras*: *R Interface to* '*Keras*'. R package version 2.2.0 (2018). https://CRAN.R-project.org/package=keras.
7. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," J. Mach. Learn. Res. **12**, 2121–2159 (2011).
8. S. Ruder, "An overview of gradient descent optimization algorithms," arXiv preprint arXiv:1609.04747v2 (2017). https://arxiv.org/abs/1609.04747. Accessed January 19, 2019.

Stanislav Sholtanyuk. Born in 1996. Assistant of the Department of Computer Applications and Systems, Faculty of Applied Mathematics and Computer Science, Belarusian State University.