Available online at http://scik.org
J. Math. Comput. Sci. 11 (2021), No. 6, 6638-6656
https://doi.org/10.28919/jmcs/6469
ISSN: 1927-5307

A COMPARISON OF LEARNING ALGORITHMS FOR SEASONAL TIME SERIES FORECASTING USING NARX MODEL

HERMANSAH[1,2,*], DEDI ROSADI[1], ABDURAKHMAN[1], HERNI UTAMI[1]

[1]Department of Mathematics, Universitas Gadjah Mada, Yogyakarta, Indonesia

[2]Department of Mathematics Education, Universitas Riau Kepulauan, Batam, Indonesia

Abstract. In this study, we propose a nonlinear autoregressive network with exogenous inputs (NARX) model with two deterministic seasonal dummy approaches, that is binary dummy variables and sine-cosine pairs. For significant lag is selected using a stepwise AIC method, including a deterministic seasonal dummy. While the number of neurons in the hidden layer is conducted by trial and error method on one to five neurons. The NARX model is trained using five types of algorithms and a tangent hyperbolic activation function. Each algorithm is compared on all approaches to see the speed of convergence and forecasting accuracy. In addition, time series data is performed using data without and with the first differencing process. The results of the case study show that the best approach to the NARX model is to use binary dummy variables and data with the first differencing process. On the other hand, the GRPROP algorithm shows the least computation time, the fastest training process steps, and forecasting accuracy with the best MAPE value. Overall, the GRPROP algorithm is the best training algorithm in this case. However, the GRPROP algorithm on the variation of its parameters shows it is not stable. While the RPROP algorithm for parameter variations shows a better speed and stability of convergence than backpropagation and GRPROP. The backpropagation algorithm occasionally outperforms GRPROP on its parameter variations.

Keywords: time series; with and without first differencing; seasonal dummy; stepwise; backpropagation; RPROP; GRPROP; NARX model.

2010 AMS Subject Classification: 37M10.

*Corresponding author

E-mail address: hermansah@mail.ugm.ac.id

Received July 06, 2021

1. Introduction

Forecasting time series data is mostly done using statistical methods. However, these statistical methods are only limited to linear models, whereas in many cases forecasting time series data has a nonlinear tendency. As an alternative, neural networks (NN) models can be used on data where the assumptions of the linear model are not met [1, 2]. The NN model is a flexible method for modelling linear and nonlinear correlation. The popular NN model for modeling and forecasting time series data is the nonlinear autoregressive neural network (NARNN) model or better known as the feed-forward neural network (FFNN) [3].

In this study, we propose a NARNN model using external inputs or a nonlinear autoregressive network with exogenous inputs (NARX) model. The external input of the NARX model is carried out using two deterministic seasonal dummy approaches, that is binary dummy variables and sine-cosine pairs. These two approaches are compared their effects on the accuracy of forecasting time series data. In addition, time series data were given two treatments, namely raw data (without the first differencing process) and data with the first differencing process. In this case, the NARX model focuses on monthly time series data that has trend and seasonal patterns. The time series data is then linearly scaled between -0.8 and 0.8 to facilitate learning of the NARX model. The main data lag was selected using the same method as [4, 5]. Next, the significant lag was selected using the stepwise AIC method. A similar approach was applied by [6] on the NARNN model.

The architecture of the NARX model is carried out using 1 input layer, 1 hidden layer with tangent hyperbolic activation function, and 1 output layer with linear activation function. The initial weights and biases were assigned randomly. The constant parameter values used refer to [7, 8, 9], see more detail in section 3. The maximum number of training processes (epoch, which in [7] is called stepmax) used is 1000000 steps. The target value of the work function used is a threshold of 0.01. The iteration will be stopped if the value of the work function is less than the threshold value. While the number of neurons in the hidden layer is done by trial and error method. In this study, we followed [10] who suggested the use of one to five neurons for time series data. Furthermore, the NARX model is trained using five types of algorithms, namely backpropagation, resilient backpropagation with weight backtracking

(RPROP with WB), resilient backpropagation without weight backtracking (RPROP without WB), the modified globally convergent algorithm with the smallest absolute gradient (GRPROP with SAG), and the modified globally convergent algorithm with the smallest learning rate (GRPROP with SLR). Each algorithm is compared on the computational aspect to see the speed of convergence and forecasting accuracy. The measure of forecasting accuracy is done with the mean absolute percent error (MAPE). A method has excellent performance if the MAPE value is below 10% and has good performance if the MAPE value is between 10% and 20% [2].

This paper is structured as follows: In this section, we have explained the background as well as the contribution of our paper. In section 2, we provide a brief description of some of the theories required for a complete description of the paper. In section 3, we provide an empirical study with two real data, namely monthly data on the number of international airline passengers and monthly data on the number of deaths in the United States of America (USA). While the conclusions are given in Section 4.

2. Preliminaries

2.1. NARX Model. The nonlinear autoregressive network with exogenous input (NARX) has been reported to be very essential for discrete time nonlinear systems and has also been defined by the following mathematical relationship [11]:

$$\begin{aligned} y(t+1) &= f(y(t), y(t-1), ..., y(t-n_y+1); u(t), u(t-1), ..., u(t-n_u+1); \mathbf{w}) \\ &= f(\mathbf{y}(t); \mathbf{u}(t); \mathbf{w}) \end{aligned} \tag{1}$$

where $u(t)$ and $y(t)$ indicate the input and output of the model at the moment t, respectively, $n_u \geq 1$ and $n_y \geq 1$ $(n_y \geq n_u)$ represent input and output memory orders, $\mathbf{w}$ is the weight matrix while f is the non-linear function expected to be estimated using the multilayer perceptron (MLP) [12].

Basically, the NARX network is trained under one out of two models [13, 14]. The first is the series-parallel architecture (or parallel architecture without feedback), where there is formation of the output's regressors only through the use of the output actual values:

$$\begin{aligned}\hat{y}(t+1) &= \hat{f}(y(t), y(t-1), ..., y(t-n_y+1); u(t), u(t-1), ..., u(t-n_u+1); \mathbf{w}) \\ &= \hat{f}(\mathbf{y}_{sp}(t); \mathbf{u}(t); \mathbf{w}) \qquad (2)\end{aligned}$$

The second model is the parallel architecture (or parallel architecture with feedback), where the output is the feedback for the feed-forward neural networks input, being part of the standard architecture:

$$\begin{aligned}\hat{y}(t+1) &= \hat{f}(\hat{y}(t), \hat{y}(t-1), ..., \hat{y}(t-n_y+1); u(t), u(t-1), ..., u(t-n_u+1); \mathbf{w}) \\ &= \hat{f}(\mathbf{y}_{p}(t); \mathbf{u}(t); \mathbf{w}) \qquad (3)\end{aligned}$$

The NARX networks trained in line with equations (2) and (3) are, therefore, represented as NARX-SP and NARX-P networks, respectively. This research was focused on NARX models with parallel architecture and without feedback (NARX-SP).

This research also applied the two conventional approaches as exogenous input variables in the NARX model. The first approach to modeling deterministic seasonal patterns is to use binary dummy variables $S-1$ for each time period t, where S is the seasonal duration. The second approach is to use a set of sine and cosine dummy variables, which have been shown to capture deterministic seasonal elements of the time series well. Two inputs $x_{s,1}$ and $x_{s,2}$ encode seasonality using an explanatory variable that is created using Sin(t) and Cos(t) for an explicit representation of the point in time within an identified seasonality of length s, see discussion in [15, 16] with

$$x_{s,1}(t) = \text{Sin}\left(\frac{2\pi t}{S}\right) \text{ and } x_{s,2}(t) = \text{Cos}\left(\frac{2\pi t}{S}\right) \qquad (4)$$

2.2. Backpropagation Learning. This is the algorithm mostly applied to supervised learning with multi-layered feed-forward networks. This is fundamentally built on the repeated implementation of the chain rule to calculate each weight's influence in the network regarding an arbitrary error function E [17]

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial s_i}\frac{\partial s_i}{\partial net_i}\frac{\partial net_i}{\partial w_{i,j}} \qquad (5)$$

where $w_{i,j}$ represents the neuron weight from i to j, s_i indicates the output, and net_i is the weighted sum of neuron i inputs. Moreover, after the determination of each weight's partial derivative, a simple gradient descent is used to minimize the error function.

$$w_{i,j}(t+1) = w_{i,j}(t) - \varepsilon \frac{\partial E}{\partial w_{i,j}}(t) \tag{6}$$

The learning rate of ε selected to scale the derivative significantly affects the time required up to the achievement of convergence. In a situation the value set is too small, there is going to be a need for several steps to achieve a satisfactory solution but a big value would cause oscillation and this has the ability to keep the error from falling beneath a defined value.

An early method proposed to ensure this problem is solved is through the introduction of a momentum-term

$$\Delta w_{i,j}(t) = -\varepsilon \frac{\partial E}{\partial w_{i,j}}(t) + \mu \Delta w_{i,j}(t-1) \tag{7}$$

The momentum parameter is observed to have scaled the initial step's effect on the present step and the momentum-term has been discovered to have the ability to ensure better stability of the learning procedure and acceleration of convergence in the error function's shallow regions.

This has, however, been discovered not to be truth every time based on practical experience since the momentum parameter's optimal value was later found to be equally problem-dependent as the learning rate without the possibility of achieving any general improvement [9].

2.3. RPROP Learning.

RPROP is an acronym for resilient propagation and it is defined as an effective new learning scheme applied to directly adapt the weight step with respect to the local gradient information. There was an introduction of an individual update-value $\Delta_{i,j}$ by [9] to ensure each weight strictly decides the weight-update size. The evolution of this adaptive update-value was observed in the learning process because of the local sight on the error function, E, through the use of the learning-rule presented as follows:

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ * \Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} > 0 \\ \eta^- * \Delta_{ij}^{(t-1)}, \text{ if } \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)}, \text{ else} \end{cases} \tag{8}$$

where $0 < \eta^- < 1 < \eta^+$.

The working principle of the adaptation-rule is such that each time there is a change in the sign of the corresponding weight $w_{i,j}$ partial derivative to show the last update was too enormous and that there is a jump of the algorithm on a local minimum, there is usually a reduction in the update-value Δ_{ij} by the factor η^-. Meanwhile, in a situation the sign is retained, there is usually a slight increase in the update-value to ensure the acceleration of convergence in shallow regions.

The weight-update is also usually a straight-forward rule after the adaption of the update-value for each weight and, in a situation the derivative is positive (increasing error), there is a reduction in the weight using the update-value but a negative condition usually leads to the update-value addition

(9)
$$\Delta w_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)}, \text{ if } \frac{\partial E}{\partial w_{i,j}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)}, \text{ if } \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0 \\ 0 \text{ , else} \end{cases}$$

(10)
$$w_{i,j}^{(t+1)} = w_{i,j}^{(t)} + \Delta w_{ij}^{(t)}$$

There is, however, one exception and this is a situation there is a change in the sign on the partial derivative such that the initial step was observed to be too large while the minimum was missed, there is going to be the reversion of the previous weight-update

(11)
$$\Delta w_{ij}^{(t)} = -\Delta w_{ij}^{(t-1)}, \text{ if } \frac{\partial E}{\partial w_{i,j}}^{(t-1)} * \frac{\partial E}{\partial w_{i,j}}^{(t)} < 0$$

The backtracking weight-step is expected to change the sign on the derivative once again in the following step without allowing the update-value to be adapted in order to avert double punishment. It is possible to practically achieved this through the setting of $\frac{\partial E}{\partial w_{i,j}}^{(t-1)} := 0$ in the Δ_{ij} previous adaptation-rule. Meanwhile, there are changes in the update-values and weights each time there is a presentation of the whole pattern set once to the network (learning by epoch). It is, however, possible to use RPROP with and without weight backtracking.

2.4. GRPROP Learning. The unconstrained minimization principles are applied to RPROP's convergence to a local minimizer by assuming that (i) $f : \mathscr{D} \subset \mathbb{R}^n \to \mathbb{R}$ is the function requiring minimization while f is bounded under $\mathbb{R}^n$, (ii) there is continuous differentiation of f in a neighborhood $\mathscr{N}$ of the level set $\mathscr{L} = \{x : f(x) \leq f(x^0)\}$, (iii) f gradient represented by g is Lipschitz continuous on $\mathbb{R}^n$ for any two points x and $y \in \mathbb{R}^n$, condition $\| g(x) - g(y) \| \leq L \| x - y \|, \forall x, y \in \mathscr{N}$, where $L > 0$ denotes the g satisfies Lipschitz constant, and x^0 is the starting point of the following iterative scheme

$$x^{k+1} = x^k + \tau^k d^k, \; k = 0, 1, ... \tag{12}$$

The general iterative scheme (12) convergence, in which d^k is the search direction and $\tau^k > 0$ is a step length requiring the adopted search direction d^k to satisfy condition $g(x^k)^\top d^k < 0$ and this ensures d^k becomes a descent direction of $f(x)$ at x^k. The length of steps in (12) is defined using some rules such as Wolfe's

$$f(x^k + \tau^k d^k) - f(x^k) \leq \sigma_1 \tau^k g(x^k)^\top d^k \tag{13}$$

$$g(x^k + \tau^k d^k)^\top d^k \geq \sigma_2 g(x^k)^\top d^k \tag{14}$$

where $g(x)$ is the f gradient at x, and $0 < \sigma_1 < \sigma_2 < 1$, and Wolfe's theorem was applied to produce the convergence results [18, 19].

The fulfillment of the assumptions (i)-(iii), for any $w^0 \in \mathbb{R}^n$ and any sequence $\{w^k\}_{k=0}^{\infty}$ produced by the RPROP scheme

$$w^{k+1} = w^k - \tau^k \operatorname{diag}\{\eta_1^k, ..., \eta_i^k, ..., \eta_n^k\} \operatorname{sign}(g(w^k)), \; k = 0, 1, ... \tag{15}$$

where $\operatorname{sign}(g(w^k))$ denotes the column vector of the signs of the components of $g(w^k) = (g_1(w^k), g_2(w^k), ..., g_n(w^k))$, $\tau^k > 0$ satisfying Wolfe's conditions, $\eta_m^k (m = 1, 2, ..., i-1, i+1, ..., n)$ are small positive real numbers generated by RPROP's learning rates schedule:

$$\text{if } (g_m(w^{k-1}) \cdot g_m(w^k) > 0) \text{ then } \eta_m^k = \min(\eta_m^{k-1} \cdot \eta^+, \Delta_{\max}) \tag{16}$$

$$\text{if } (g_m(w^{k-1}) \cdot g_m(w^k) < 0) \text{ then } \eta_m^k = \max(\eta_m^{k-1} \cdot \eta^-, \Delta_{\min}) \tag{17}$$

$$\text{if } (g_m(w^{k-1}) \cdot g_m(w^k) = 0) \text{ then } \eta_m^k = \eta_m^{k-1} \tag{18}$$

where $0 < \eta^- < 1 < \eta^+, \Delta_{\max}$ is the learning rate upper bound, $\Delta_{\min}$ is the learning rate lower bound and

$$\eta_i^k = -\frac{\sum_{j\neq i, j=1}^{n} \eta_j^k g_j(w^k) + \delta}{g_i(w^k)}, \ 0 < \delta < \infty, \ g_i(w^k) \neq 0 \tag{19}$$

it holds that $\lim_{k\to\infty} g(w^k) = 0$.

The modified RPROP, known as the GRPROP, is applied through the relations (15)-(19). It is, however, important to note that it is possible to apply relation (19) randomly or in a cyclic pattern on the local learning rates. In other situations, the selection of the *i*th coordinate with the absolute smallest no zero $g_i(w^k)$ value may be better to produce a larger value for η_i^k. Moreover, the replacement of each time there is a production of smallest learning rate value by the schedule of the RPROP with the η_i^k value derived from equation (19) is another method applicable and it was observed to have been used in all the experiments in [8].

The GRPROP algorithm used was induced by the globally convergent with the smallest absolute gradient and the smallest learning rate. This was associated with the resilient backpropagation without weight backtracking as well as modifying one learning rate, including those related to the smallest absolute gradient or the smallest learning rate. Moreover, there is the limitation of the learning rates in this algorithm to the vector-defined boundaries or the list with the minimum and maximum limits.

3. Main Results

This research was conducted in two real cases: the first is the number of international airline passengers and the second case is the number of accidental deaths in the United States of America (USA).

3.1. International Airline Passengers.

The data used is monthly data on the number of international airline passengers. Monthly data from January 1949 to December 1959 were used as training data and January to December 1960 were used as testing data. Monthly data on the number of international airline passengers is popular data and is often used as an example of the application of the classical time series method, among others, in [20, 21, 22, 23]. Graphically,

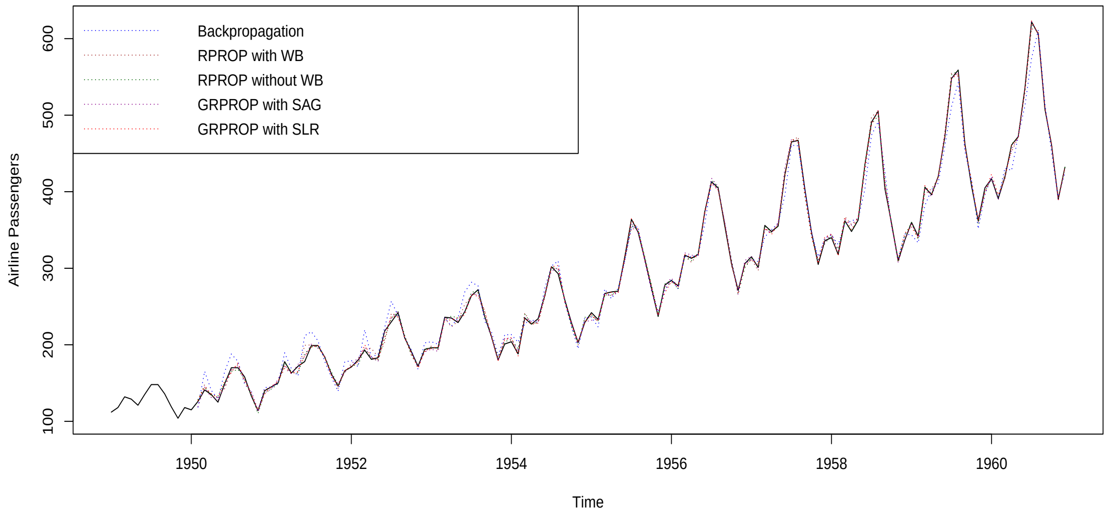

FIGURE 1. Result of algorithms on monthly data of international airline passengers from 1949 to 1960.

it can be seen in Figure 1 that the data has an uptrend and seasonal pattern with multiplicative nature.

In this study, the determination of the autoregressive lag was chosen using the following approach: if the length of the seasonal period is m, then the sequential m lag starting from lag 1 is used. For example, lag for quarterly data of 1:4 and for monthly data of 1:12 is used. According to [4, 5], seasonal patterns can be captured more easily through this approach. Next, the significant lag was selected using the stepwise AIC method. A similar approach was applied by [6] on a previous study for the NARNN model. In this study, the NARNN model is given external input or better known as the NARX model. The external input of the NARX model is made using a deterministic seasonal dummy, namely a binary dummy variable and a sine-cosine pair. These two approaches are compared their effects on the forecasting accuracy of the NARX model on data with and without the first differencing process.

The architecture of the NARX model used is 1 input layer, 1 hidden layer with tangent hyperbolic activation function, and 1 output layer with linear activation function. For the selection of the initial weights and biases the network is assigned randomly. The constant parameter values used refer to [7, 8, 9], namely the increase factor $\eta^+ = 1.2$, the decrease factor $\eta^- = 0.5$,

TABLE 1. Comparison of the computational results of each algorithm with data without the first differencing process and the external input of the NARX model with binary dummy variables

Algorithm type	Number of hidden neurons	Computational time	Reached stepmax	Reached threshold	MAPE testing
Backpropagation	1	5.2918 secs	52568	0.0099	0.0365
	2	3.3981 secs	35810	0.0099	0.0299
	3	8.3117 secs	74534	0.0099	0.0289
	4	6.0173 secs	47713	0.0099	0.0342
	5	6.6728 secs	46313	0.0099	0.0339
RPROP with WB	1	0.6120 secs	5077	0.0099	0.0356
	2	0.4850 secs	2855	0.0095	0.0257
	3	0.3117 secs	1137	0.0099	0.0258
	4	0.2356 secs	452	0.0096	0.0253
	5	0.2318 secs	344	0.0081	0.0231
RPROP without WB	1	0.4576 secs	4436	0.0099	0.0356
	2	0.3903 secs	2786	0.0098	0.0270
	3	0.3452 secs	1665	0.0098	0.0252
	4	0.2500 secs	416	0.0099	0.0286
	5	0.3810 secs	1201	0.0096	0.0263
GRPROP with SAG	1	58.998 secs	395408	0.0489	0.0390
	2	10.516 secs	97071	0.0093	0.0256
	3	2.6372 secs	18971	0.0073	0.0253
	4	0.3363 secs	1364	0.0096	0.0211
	5	0.2742 secs	806	0.0091	0.0223
GRPROP with SLR	1	1.5257 secs	15263	0.0085	0.0360
	2	0.3807 secs	2025	0.0095	0.0258
	3	1.5020 secs	10817	0.0059	0.0234
	4	3.2849 secs	22836	0.0082	0.0211
	5	9.0624 secs	25271	0.0092	0.0210

the upper limit $\Delta_{\max} = 0.1$, the lower limit $\Delta_{\min} = 10^{-10}$, the initial update value $\Delta_0 = 0.1$, and the learning rate $\varepsilon = 0.001$ (only for backpropagation). The maximum number of training

TABLE 2. Comparison of the computational results of each algorithm on the data with the first differencing process and the external input of the NARX model with binary dummy variables

Algorithm type	Number of hidden neurons	Computational time	Reached stepmax	Reached threshold	MAPE testing
	1	4.9675 secs	74772	0.0099	0.0298
	2	6.4983 secs	87279	0.0099	0.0237
Backpropagation	3	3.8116 secs	44779	0.0099	0.0249
	4	8.2948 secs	86271	0.0099	0.0220
	5	8.5292 secs	80817	0.0099	0.0212
	1	0.2031 secs	1261	0.0098	0.0299
	2	0.2655 secs	1772	0.0098	0.0197
RPROP with WB	3	0.3279 secs	910	0.0090	0.0189
	4	0.3749 secs	2008	0.0095	0.0147
	5	0.4061 secs	2126	0.0098	0.0116
	1	0.1405 secs	777	0.0095	0.0301
	2	0.3592 secs	3090	0.0098	0.0220
RPROP without WB	3	0.1718 secs	750	0.0099	0.0177
	4	0.2657 secs	1484	0.0098	0.0126
	5	0.1872 secs	636	0.0092	0.0138
	1	0.1093 secs	303	0.0094	0.0329
	2	0.4529 secs	3768	0.0095	0.0225
GRPROP with SAG	3	0.2499 secs	1303	0.0099	0.0179
	4	0.1562 secs	606	0.0090	0.0148
	5	0.2187 secs	931	0.0092	0.0136
	1	0.9216 secs	10632	0.0098	0.0298
	2	0.9372 secs	8900	0.0095	0.0212
GRPROP with SLR	3	2.5306 secs	22406	0.0089	0.0180
	4	0.7029 secs	3902	0.0091	0.0149
	5	7.1700 secs	52457	0.0093	0.0101

processes (stepmax) used is 1000000 steps. The target of the work function used is the goal performance value (threshold) of 0.01. The iteration will be stopped if the value of the work function is less than the threshold value. The number of neurons in the hidden layer is carried out using a trial and error method following [10] who suggests the use of one to five neurons in time series data.

Furthermore, the NARX model training algorithm uses five types of algorithms, namely the backpropagation, the resilient backpropagation with weight backtracking (RPROP with WB), the resilient backpropagation without weight backtracking (RPROP without WB), the modified globally convergent algorithm with the smallest absolute gradient (GRPROP with SAG), and the modified globally convergent algorithm with the smallest learning rate (GRPROP with SLR). In this study, each algorithm is compared on the computational aspect to see the speed of convergence and forecasting accuracy. The measure of forecasting accuracy is done with the mean absolute percent error (MAPE). A method has very good performance if the MAPE value is below 10% and has good performance if the MAPE value is between 10% and 20% [2].

In this study, the NARX model focuses on monthly data that has trend and seasonal patterns. The data were then scaled linearly between -0.8 and 0.8 to facilitate the training of the NARX model. In the case of the first data, four approaches are taken: first, the NARX model with external input uses a binary dummy variable on the data without the first differencing process, where the second approach uses the first differencing process. Third, the NARX model with external input uses dummy variables of sine-cosine pairs on the data without the first differencing process, where the fourth approach uses the first differencing process. In the first and second approaches, the selected lags using the stepwise AIC method are 1, 4, 5, 8:12 and 1:8, 10, 12, respectively, including a binary dummy variable. Meanwhile, in the third and fourth approaches, the lag is equal to 1:12 and the dummy variable is the sine-cosine pair.

The results of the comparison of the NARX model with binary dummy variables for each algorithm can be seen in Tables 1 and 2, while the NARX model with dummy sine-cosine variables can be seen in Tables 3 and 4. The comparison process was carried out using data without the first differencing process (Tables 1 and 3) and with the first differencing process (Tables 2 and 4). The parameter used for comparison of each algorithm is the model that gives the best results. Overall, the results of the MAPE value on the testing data show that the NARX model with a binary dummy variable is more accurate. On the other hand, the

TABLE 3. Comparison of the computational results of each algorithm with data without the first differencing process and the external input of the NARX model with dummy variables of sine-cosine pairs

Algorithm type	Number of hidden neurons	Computational time	Reached stepmax	Reached threshold	MAPE testing
	1	2.8043 secs	39987	0.0099	0.0410
	2	5.6381 secs	62482	0.0099	0.0400
Backpropagation	3	7.7963 secs	81549	0.0099	0.0322
	4	4.9519 secs	45807	0.0099	0.0371
	5	4.7123 secs	41487	0.0099	0.0318
	1	0.9101 secs	9890	0.0090	0.0403
	2	0.4120 secs	1897	0.0091	0.0328
RPROP with WB	3	0.5389 secs	2639	0.0095	0.0271
	4	0.3279 secs	1457	0.0090	0.0302
	5	0.3254 secs	1449	0.0099	0.0310
	1	0.4034 secs	3369	0.0097	0.0400
	2	0.5233 secs	4572	0.0091	0.0368
RPROP without WB	3	0.7927 secs	2037	0.0097	0.0324
	4	0.6415 secs	2455	0.0088	0.0290
	5	0.3507 secs	1969	0.0097	0.0294
	1	53.499 secs	592622	0.0634	0.0479
	2	1.3656 secs	7657	0.0250	0.0359
GRPROP with SAG	3	50.719 secs	443708	0.0063	0.0270
	4	9.0323 secs	63096	0.0095	0.0233
	5	7.5300 secs	51807	0.0097	0.0242
	1	1.0672 secs	8602	0.0099	0.0402
	2	0.9008 secs	7644	0.0092	0.0355
GRPROP with SLR	3	7.0123 secs	60033	0.0097	0.0262
	4	1.1670 secs	7602	0.0086	0.0282
	5	15.572 secs	61565	0.0092	0.0232

NARX model got better on the data with the first differencing process. In the data without the first differencing process, the convergence of the RPROP with WB algorithm provides the

TABLE 4. Comparison of the computational results of each algorithm on the data with the first differencing process and the external input of the NARX model with dummy variables of sine-cosine pairs

Algorithm type	Number of hidden neurons	Computational time	Reached stepmax	Reached threshold	MAPE testing
Backpropagation	1	1.2190 secs	10878	0.0099	0.0308
	2	6.1995 secs	84655	0.0099	0.0293
	3	6.1074 secs	73626	0.0099	0.0216
	4	7.9596 secs	86052	0.0099	0.0200
	5	5.3138 secs	52447	0.0099	0.0251
RPROP with WB	1	0.1353 secs	490	0.0085	0.0309
	2	0.1500 secs	667	0.0089	0.0267
	3	0.1942 secs	951	0.0076	0.0236
	4	0.3236 secs	1839	0.0099	0.0189
	5	0.3797 secs	2224	0.0084	0.0177
RPROP without WB	1	0.1438 secs	678	0.0089	0.0311
	2	0.1515 secs	736	0.0099	0.0266
	3	0.3436 secs	2551	0.0098	0.0213
	4	0.1562 secs	672	0.0099	0.0180
	5	0.3849 secs	1409	0.0092	0.0179
GRPROP with SAG	1	6.6670 secs	86165	0.0455	0.0314
	2	22.761 secs	96849	0.0093	0.0269
	3	48.914 secs	356735	0.0092	0.0220
	4	2.5986 secs	20892	0.0089	0.0157
	5	6.5366 secs	19338	0.0089	0.0171
GRPROP with SLR	1	2.0126 secs	25188	0.0084	0.0307
	2	3.6242 secs	39906	0.0096	0.0272
	3	0.9276 secs	7836	0.0083	0.0231
	4	1.7895 secs	13807	0.0096	0.0206
	5	12.4531 secs	96478	0.0098	0.0123

least computation time (0.2318 secs) and the fastest training process steps (344 steps). The convergence speed of the RPROP without WB algorithm ranks second with a slight difference.

While the data with the first differencing process, the convergence of the GRPROP with SAG algorithm provides the least computation time (0.1093 secs) and the fastest training process steps (303 steps). The convergence speed of the RPROP with WB algorithm ranks second with 0.1353 secs and 490 steps. Overall, the best results were obtained using the GRPROP algorithm. However, the RPROP algorithm shows that it is much better with respect to the resistance of its parameter variations than backpropagation and GRPROP. The backpropagation algorithm occasionally outperforms GRPROP with SAG with respect to the resistance of its parameter variations (indicated by the reached threshold). In this case, the GRPROP with SAG algorithm often does not reach convergence for the stepmax or threshold used.

Furthermore, the comparison of the forecasting accuracy of each algorithm is carried out using the MAPE value on the testing data. In this case, the GRPROP with SLR algorithm consistently gives the best MAPE value on the data without and with the first differencing process. Sequentially, the best MAPE values were obtained at 0.0210 and 0.0101. While the GRPROP algorithm with SAG ranks second with a slight difference. On the other hand, the number of neurons in the hidden layer of each algorithm is obtained differently to achieve convergence. The results generally show that the greater the number of neurons in the hidden layer, the smaller the MAPE value in the testing data obtained. This indicates that the number of neurons in the hidden layer affects forecasting accuracy. Overall, the GRPROP algorithm is more accurate than the backpropagation and RPROP algorithms. However, the RPROP algorithm outperforms backpropagation in terms of convergence speed and forecasting accuracy.

3.2. Accidental Deaths in the USA.

Monthly data on the number of deaths in the United States of America (USA) due to accidents, can be graphically seen in Figure 2. Monthly data from January 1973 to December 1978 were used as training data and data from January 1979 to June 1979 were used as test data. This data is quite popular because of its complex pattern and has been discussed in several studies. Brockwell and Davis [24] have used this data as an example of applying the ARIMA model and exponential smoothing.

Based on the results in the first case, the NARX model approach in the second case is only shown using data with the first differencing process and external input using a binary dummy variable. In this case, the selected lag using the stepwise AIC method is 1, 2, 8, 11, 12, and a

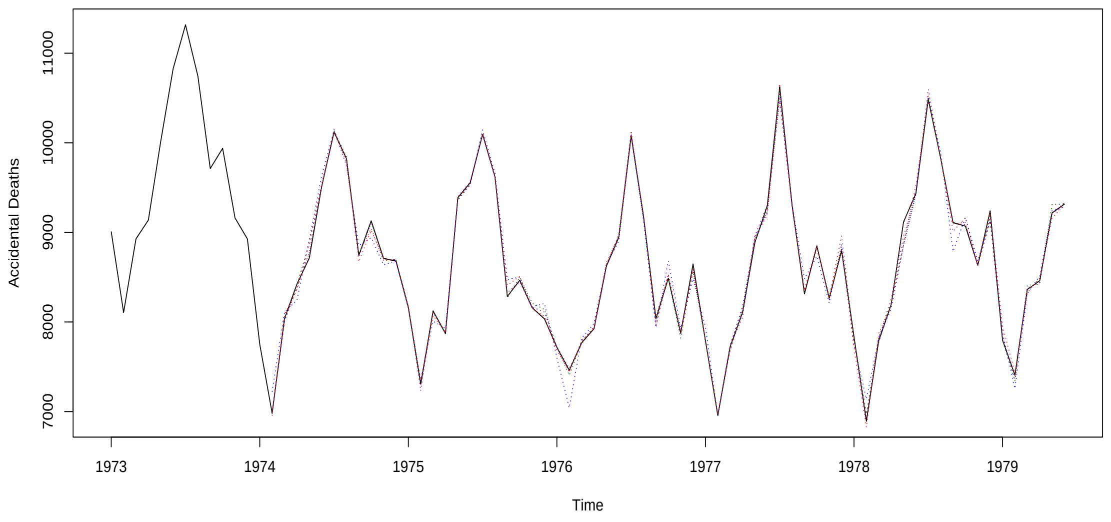

FIGURE 2. Result of algorithms on monthly data of accidental deaths in the USA from 1973 to 1979.

binary dummy variable. The results of the comparison of each algorithm can be seen in Table 5. In Table 5, the GRPROP with SAG algorithm provides the least computation time (0.1561 secs), the fastest training process steps (311 steps), and the best MAPE value (0.26%). Overall, the best results were obtained using the GRPROP with SAG algorithm with respect to convergence speed and forecasting accuracy.

4. CONCLUSION

In this study, the best approach to the NARX model was obtained using data with the first differencing process and external input using a binary dummy variable. The fastest computation time and training process steps are obtained using the GRPROP with SAG algorithm. While the forecasting accuracy was obtained using a different GRPROP algorithm, GRPROP with SLR on monthly data on the number of international airline passengers and GRPROP with SAG on monthly data on the number of deaths in the USA. Overall, the GRPROP algorithm (either with SAG or SLR) is the best training algorithm in this study. However, the GRPROP with SAG algorithm for parameter variations often shows that it does not reach convergence, while GRPROP with SLR requires more computational time and training process steps. On the

TABLE 5. Comparison of the computational results of each algorithm on the data of accidental deaths in the USA with the first differencing process and the external input of the NARX model with binary dummy variables

Algorithm type	Number of hidden neurons	Computational time	Reached stepmax	Reached threshold	MAPE testing
Backpropagation	1	4.8647 secs	86681	0.0099	0.0212
	2	5.7174 secs	96034	0.0099	0.0175
	3	4.2801 secs	65740	0.0099	0.0161
	4	6.7802 secs	97750	0.0099	0.0119
	5	4.7801 secs	63905	0.0099	0.0111
RPROP with WB	1	0.2499 secs	1854	0.0096	0.0219
	2	0.4530 secs	4073	0.0097	0.0148
	3	0.3437 secs	2583	0.0099	0.0080
	4	0.2030 secs	811	0.0098	0.0040
	5	0.2186 secs	1128	0.0099	0.0039
RPROP without WB	1	0.1562 secs	600	0.0096	0.0214
	2	0.1874 secs	953	0.0098	0.0131
	3	0.1874 secs	724	0.0090	0.0109
	4	0.2970 secs	1902	0.0098	0.0053
	5	0.2343 secs	1379	0.0092	0.0031
GRPROP with SAG	1	0.1561 secs	311	0.0099	0.0227
	2	0.1561 secs	572	0.0088	0.0154
	3	0.2031 secs	1224	0.0098	0.0098
	4	0.1874 secs	895	0.0091	0.0052
	5	0.2186 secs	1153	0.0097	0.0026
GRPROP with SLR	1	0.6873 secs	8294	0.0083	0.0217
	2	0.3905 secs	3691	0.0084	0.0152
	3	0.4686 secs	4226	0.0098	0.0134
	4	0.2186 secs	1163	0.0090	0.0084
	5	0.3124 secs	2080	0.0096	0.0047

other hand, the RPROP algorithm for its parameter variations shows a better speed and stability of convergence than GRPROP. The RPROP algorithm (both with and without WB) shows that it is easy to implement. In addition, the computational time of the training process is obtained very small, while the number of steps of the training process is significantly reduced. The RPROP algorithm on several data patterns was found to be very good with respect to convergence and robustness. Meanwhile, the backpropagation algorithm occasionally outperforms GRPROP with SAG for its parameter variations.

CONFLICT OF INTERESTS

The author(s) declare that there is no conflict of interests.

REFERENCES

[1] Hermansah, D. Rosadi, Abdurakhman, H. Utami, Comparison of ARIMA, neural network, and wavelet models for forecasting indonesia sharia stock index, in: Proceedings of the 62nd ISI World Statistics Congress, International Statistical Institute, Kuala Lumpur, Malaysia, 2019, pp. 286-293.

[2] Hermansah, D. Rosadi, H. Utami, Abdurakhman, G. Darmawan, Multivariate time series data forecasting using multi-output NARNN model, in: Proceedings of the 7th International Conference on Research, Implementation, and Education of Mathematics and Sciences (ICRIEMS), Atlantis Press, Yogyakarta, Indonesia, 2020, pp. 288-294.

[3] Hermansah, D. Rosadi, H. Utami, Abdurakhman, G. Darmawan, Hybrid MODWT-FFNN model for time series data forecasting, in: Proceedings of the 8th SEAMS-UGM International Conference on Mathematics and its Applications, AIP Publishing, Yogyakarta, Indonesia, 2019, pp. 1-8.

[4] F. Martinez, M. P. Frias, M. D. Perez, A. J. Rivera, A methodology for applying k-nearest neighbor to time series forecasting, Artif. Intell. Rev. 52 (3) (2017), 2019-2037.

[5] F. Martinez, F. Charte, A. J. Rivera, M. P. Frias, Automatic time series forecasting with GRNN: A comparison with other models, in: Proceedings of the International Work Conference on Artificial Neural Networks, Springer, Gran Canaria, Spain, 2019, pp. 198-209.

[6] Hermansah, D. Rosadi, Abdurakhman, H. Utami, Selection of input variables of nonlinear autoregressive neural network model for time series data forecasting, Med. Stat. 13 (2) 2020, 116-124, .

[7] F. Gunther, S. Fritsch, Neuralnet: Training of neural networks, R J. 2 (1) (2010), 30-38.

[8] A. D. Anastasiadis, G. D. Magoulas, M. N. Vrahatis, New globally convergent training scheme based on the resilient propagation algorithm, Neurocomputing 64 (2005), 253-270.

[9] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: The RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE, San Francisco, California, 1993, pp. 586-591.

[10] A. Sihabuddin, Variable length moving average dan korelasi kedua pada model peramalan multivariabel valuta dengan jaringan saraf tiruan, Universitas Gadjah Mada, Yogyakarta, (2016).

[11] H. T. Siegelmann, B. G. Horne, C. L. Giles, Computational capabilities of recurrent NARX neural networks, IEEE Trans. Syst., Man, Cybern. B. Cybern. 27 (2) (1997), 208-215.

[12] S. N. Kumpati, P. Kannan, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Netw. 1 (1) (1990), 4-27.

[13] C. Jiang, F. Song, Sunspot forecasting by using chaotic time-series analysis and NARX network, J. Comput. 6 (7) (2011), 1424-1429.

[14] E. Diaconescu, The use of NARX neural networks to predict chaotic time series, WSEAS Trans. Comput. Res. 3 (3) (2008), 182-191.

[15] S. F. Crone, N. Kourentzes, Feature selection for time series prediction: A combined filter and wrapper approach for neural networks, Neurocomputing 73 (10) (2010), 1923-1936.

[16] Hermansah, D. Rosadi, Abdurakhman, H. Utami, Time series forecasting with trend and seasonal patterns using NARX network ensembles, Math. Stat. 9 (4) (2021), 511-520.

[17] D. E. Rumelhart, J. L. Mcclelland, Parallel distributed processing: Explorations in the microstructure of cognition, MIT Press, Cambridge, (1986).

[18] P. Wolfe, Convergence conditions for ascent methods, J. Soc. Ind. Appl. Math. 11 (2) (1969), 226-235.

[19] P. Wolfe, Convergence conditions for ascent methods II: Some corrections, J. Soc. Ind. Appl. Math. 13 (2) (1971), 185-188.

[20] G. E. P. Box, G. M. Jenkins, Time series analysis: Forecasting and control, Holden Day, San Francisco, (1976).

[21] J. Faraway, C. Chatfield, Time series forecasting with neural networks: A comparative study using the airline data, J. R. Stat. Soc. 47 (2) (1998), 231-250.

[22] Suhartono, Feedforward neural networks untuk pemodelan runtun waktu, Universitas Gadjah Mada, Yogyakarta, (2007).

[23] R. J. Hyndman, A. B. Koehler, J. K. Ord, R. D. Snyder, Forecasting with exponential smoothing: The state space approach, Springer Science and Business Media, Germany, (2008).

[24] P. J. Brockwell, R. A. Davis, Introduction to time series and forecasting, Springer Verlag, New York, (2002).