Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 22, No. 1, November 2022, pp. 51∼82
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 200/M/KPT/2020
DOI: 10.30812/matrik.v22i1.1826

Optimization of Performance Traditional Back-propagation with Cyclical Rule for Forecasting Model

Anjar Wanto[1], Ni Luh Wiwik Sri Rahayu Ginantra[2], Surya Hendraputra[3], Ika Okta Kirana[4], Abdi Rahim Damanik[5]
[1,4,5]STIKOM Tunas Bangsa, Pematangsiantar, Indonesia
[2]Institut Bisnis dan Teknologi Indonesia, Denpasar, Indonesia
[3]Politeknik Ganesha, Medan, Indonesia

Article Info

Article history:

Received March 04, 2022
Revised August 22, 2022
Accepted October 03, 2022

Keywords:

Cyclical rule
Forecasting Model
Machine learning
Optimization
Traditional Back-propagation

ABSTRACT

The traditional Back-propagation algorithm has several weaknesses, including long training times and significant iterations to achieve convergence. This study aims to optimize traditional Back-propagation using the cyclical rule method to cover these weaknesses. Optimization is done by changing the training function and standard Back-propagation parameters using the training function and cyclical rule parameters. After that, a comparison of the two results will be carried out. This study uses quantitative method of time-series data on coronavirus cases sourced from the Worldometer website, then analyzed using three forecasting models with five input layers, one hidden layer (5, 10, and 15 neurons) and one output layer. The results showed that the 5-10-1 model with the training function and cyclical rule parameters and the tansig and purelin activation functions could perform well in optimization, including faster training time and smaller iterations (epochs), MSE training performance, and better tests. Low and high accuracy (92%) with an error rate of 0.01. So it was concluded that the training function and cyclical rule parameters with the tansig and purelin activation functions were able to optimize the traditional Back-propagation method, and the 5-10-1 model could be used for forecasting active cases of the coronavirus in Asia.

Corresponding Author:

Anjar Wanto, 082294365929,
Informatics Engineering Study Program,
STIKOM Tunas Bangsa, Pematangsiantar, Indonesia.
Email: anjarwanto@ieee.org

How to Cite: A. Wanto, N. L. W. S. R. Ginantra, S. Hendraputra, I. O. Kirana, and A. R. Damanik, Optimization of Performance Traditional Back-propagation with Cyclical Rule for Forecasting Model, MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer, vol. 22, no. 1, pp. 51-82, Nov. 2022.

1. INTRODUCTION

The whole world has been shocked by the phenomenon of the emergence of a deadly virus that is quite dangerous and quickly spreads to humans called Coronavirus disease (COVID-2019). Coronaviruses (CoV) are viruses that cause illnesses such as the common cold to more severe illnesses such as Middle East Respiratory Syndrome (MERS-CoV) and Severe Acute Respiratory Syndrome (SARS-CoV) [1–4]. Even today, the virus still exists and infects the world community, especially Indonesia, although it is not too significant compared to previous years. However, this should always be watched out for, considering that this virus is a very dangerous virus [5]. Therefore, research on forecasting when the development of this virus will end in Indonesia is very important. As proposed in this study, it's just that it's still limited to building the best forecasting model using the Back-propagation algorithm which is optimized with Cyclical order weight / bias, which can later be used as a reference or reference for forecasting about the development of this corona virus.

Back-propagation is widely used for training feedforward neural networks which are gradient-based algorithms [6–8]. Based on its development, Back-propagation has different functions, including the activation function (transfer) and the training function, each of which has many methods and techniques that can be used to solve complex computational problems. Activation functions that are often used in machine learning (machine learning) based on artificial neural networks, especially Back-propagation, include: tansig activation function (hyperbolic sigmoid tangent) [9], logsig activation function (log-Sigmoid) [10], and linear activation function (purelin) [11]. Commonly used training functions in standard Back-propagation include the gradient descent technique which includes traingd, traingdm, traingdx, and traingda [12]. However, there are many other training functions that can be used to optimize and influence the results and computational processes, such as Levenberg Marquardt or commonly called trainlm [13, 14]. Batch training following the terms of bias and weight learning (trainb) [15], quasi-Newton BFGS (trainbfg) [16], quasi-Newton BFGS with reference adaptive control (trainbfgc) [17], bayesian regulation (trainbr) [18], unsupervised training bias/weights batch (trainbu) [19], cyclical rule (trainc) [20], conjugate gradient training (traincgf, traincgb, traingcgp) [21], One-Step Secant (trainoss) [22], learning function training with additional random order (trainr) [23], resilient or commonly called trainrp [24], unsupervised random command training of weights/bias (trainru) [25], learning function training with sequential incremental (trains) [26], and gradient conjugate scale (trainscg) [27]. In fact, the use of the transfer function or the training function produces different forecasting accuracy, depending on the parameters of the given method and the data to be forecasted [28–32]. Based on this, the discussion in this paper focuses on the use of the training function that will be used to optimize the capabilities of the traditional Back-propagation method. The widespread use of the training function is able to provide optimal performance in solving many complex problems [33].

S. Mohan, et al (2022), In his research, he predicted the impact of Covid-19 using the Supervised Machine Learning EAMA model using a combination of ensemble, autoregressive, and mobile regressive learning techniques. The dataset was obtained from the Indian ministry and the Worldometer dataset from February to July 2020. The study forecasts future data using the trends of the past data, and produces averaged aggregate results. The proposed model has 80%, 10%, and 10% of the total data for training, testing, and validation, respectively. Therefore, the prediction performance looks high, as well as the validation accuracy [34]. The difference between this research and the research to be conducted lies in the algorithm used and the resulting model. This study uses EAMA with a combination of ensemble, autoregressive, and mobile regressive learning techniques, while the proposed research uses Back-propagation which is optimized by using a combination of Cyclical rule techniques.

R. Katoch and A. Sidhu (2021) conducted a study for forecasting the dynamics of the Covid Epidemic in India by applying the Autoregressive Integrated Moving Average (ARIMA) model approach, to analyze the temporal dynamics of the COVID-19 outbreak in India and to predict the final size and trend of the epidemic during the period after 16 September 2020 with Indian epidemiological data in India. national and state levels. The time for data sampling was from January 30, 2020 to September 16, 2020. ARIMA modeling was used to monitor the spread of the Covid-19 disease in India. This has significant forecasting implications for all types of industries including healthcare. Empirical results say, the number of confirmed cases of COVID-19 will reach 25,669,294 in the next 230 days at the national level. The model predicts the final epidemic size at the national level at around 5,020,35925,669,294 cases. By looking at the exponential growth in the series, it is hoped that the hypothetical inflection point of the cumulative number of confirmed COVID-19 cases can be reached at least after 23 April 2021 at the national level. But in some areas such as Andhra Pradesh, Maharashtra, Karnataka and Tamil Nadu it took 72, 182, 183 and 82 days respectively to reach the inflection point, which does not appear to be a conservative approach. Empirical results of all actual values for October are within 95% of the estimated model output [35]. The difference between this research and the research to be conducted lies in the algorithm used and the resulting model. This study uses a Data Mining algorithm with the ARIMA method, while the proposed research uses a Back-propagation Neural network algorithm which is optimized by using a combination of Cyclical rule techniques.

M. Shawaqfah and F. Almomani (2021) forecasting the Covid-19 outbreak using the Levenberg-Marquardt Back-propagation Neural Network by taking case studies in several countries such as Qatar, Spain and Italy. The ANN architecture was developed to predict the impact of serious pandemic outbreaks in Qatar, Spain and Italy. Official statistical data collected from each country up to July 6th was used to validate and test the prediction model. The sensitivity of the model was analyzed using the root mean square error (RMSE), the mean absolute percentage error and the regression coefficient index R2, which yielded a highly accurate value of the predicted correlation for infected cases and deaths of 0.99 for the dates considered. The training in collecting data on infection and death cases was carried out using various combinations of transfer functions (sigmoid (S), sigmoid (S), hyperbolic tangent (HT), and hyperbolic secant (HS). infected cases/deaths due to covid-19 were achieved with a combination of 5-4-4-2 and 5-5-5-2 [36]. The difference between this research and the proposed research lies in the algorithm used and the resulting model. This study uses Levenberg-Marquardt Back-propagation, while the proposed research uses Back-propagation which is optimized by using a combination of Cyclical rule techniques with the benchmark for selecting the best model seen from the lowest Mean Square Error (MSE) value and the highest accuracy.

Based on the related studies that have been described, the research proposed in this paper is in the form of optimizing the optimal use of training functions on traditional Back-propagation Machine learning neural networks in active cases of Coronavirus in Asia. This paper discusses more about the techniques and methods of the training function used to solve the problem, the dataset of active cases of Coronavirus in Asia is only used to assist in the verification and testing process. In its application the traditional back-propagation machine learning method often gives poor convergence speed in the training process. Therefore, it is necessary to carry out various combinations of training functions to accelerate the convergence of network training, namely by using the cyclical order weight bias method. The performance, capability and accuracy of the traditional Back-propagation algorithm will be compared, evaluated and analyzed using the cyclical order weight bias (cyclical rule) method. This study aims to analyze the performance, ability and accuracy of the traditional Back-propagation algorithm and to optimize the training function with the cyclical order weight bias method. The results of the study will prove whether the cyclical order weight bias method is able to provide optimal performance based on the traditional Back-propagation method so that it is feasible to use for forecasting data on active cases of Coronavirus in Asia. The results of this study are expected to be used or become a reference to complete the forecasting of Active Coronavirus Cases in Asia, besides that it is expected to help academics for the development of the next study.

2. RESEARCH METHOD

2.1. Datasets Used

Data collection uses quantitative methods, namely in the form of daily times-series data on corona virus cases reported by countries in Asia in the last 1-2 months (June-July 2021), which can be seen on the Worldometer website https://www.worldometers.info/-coronavirus/ consisting of 49 countries (Afghanistan, Armenia, Azerbaijan, Bahrain, Bangladesh, Bhutan, Brunei, Cambodia, China, Cyprus, Georgia, Hong Kong, India, Indonesia, Iran, Iraq, Israel, Japan, Jordan, Kazakhstan, Kuwait, Kyrgyzstan , Laos, Lebanon, Macao, Malaysia, Maldives, Mongolia, Myanmar, Nepal, Oman, Pakistan, Palestine, Philippines, Qatar, South Korea, Saudi Arabia, Singapore, Sri Lanka, Syria, Taiwan, Tajikistan, Thailand, Timor-Leste, Turkey, UAE, Uzbekistan, Vietnam and Yemen) [37]. Worldometer is run by an international team of developers, researchers and volunteers with the aim of making world statistics available in mind-blowing format that is relevant to the current time and visible to the wider community around the world. Worldometer is published by a small, independent, digital media company based in the United States with no political, governmental, or corporate affiliations. In addition, it has no investors, donors, grants or backers of any kind, which is completely self-sufficient and self-financed through automated programmatic advertising that is sold in real time on multiple ad exchanges. Worldometer was selected as one of the best free reference sites by the American Library Association (ALA), the oldest and largest library association in the world. Worldometer provides a global provider of COVID-19 statistics for many concerned people around the world. The data is trusted and used by the UK Government, Johns Hopkins CSSE, Thailand Government, Pakistan Government, Sri Lanka Government, Vietnam Government, Financial Times, The New York Times, Business Insider, BBC, and many others [38]. For COVID-19 data, Worldometer collects data from official reports, directly or indirectly from Government communication channels, through local media sources it deems to be reliable. Worldometer provides the source of each data update in the ”Recent Updates” (News) section. The data is updated thanks to the participation of users worldwide and a dedicated team of analysts and researchers who validate the data based on a growing list of more than 5,000 sources. The research dataset used can be seen in Table 1.

Table 1. Active Cases of the Corona Virus Pandemic in Asia

No	Country (X)	23-6-2021 (A1)	24-6-2021 (A2)	3-7-2021 (A3)	4-7-2021 (A4)	5-7-2021 (A5)	11-7-2021 (B1)	12-7-2021 (A6)	24-7-2021 (A7)	25-7-2021 (A8)	28-7-2021 (A9)	29-7-2021 (A10)	30-7-2021 (B2)
1	Afghanistan	40.270	41.347	45.819	46.174	46.790	47.785	47.454	46.103	45.611	44.161	43.655	42.932
2	Armenia	3.716	3.707	3.869	3.896	3.881	4.186	4.181	5.007	5.090	5.213	5.394	5.558
3	Azerbaijan	841	815	961	1.006	1.035	1.508	1.534	2.861	3.148	3.913	4.526	5.442
4	Bahrain	5.314	4.813	2.376	2.252	1.821	1.148	1.115	834	818	851	892	991
5	Bangladesh	64.284	67.269	92.145	95.955	100.570	130.269	136.797	149.097	145.959	152.559	155.082	155.453
6	Bhutan	268	272	312	288	288	301	303	253	258	223	168	135
7	Brunei	8	8	9	9	9	17	18	41	49	60	60	55
8	Kamboja	4.863	4.787	6.479	6.831	7.053	7.582	7.468	6.548	6.302	6.337	6.121	6.012
9	China	492	486	431	428	429	466	478	663	681	795	862	932
10	Cyprus	1.548	1.717	4.288	4.827	5.561	10.629	11.461	19.340	19.545	19.718	19.539	20.822
11	Georgia	8.603	8.679	10.089	10.089	9.386	12.474	12.404	21.860	23.599	25.870	27.482	29.299
12	Hong Kong	75	78	95	95	94	81	78	69	68	63	62	61
13	India	619.739	602.386	492.301	489.128	470.798	457.915	439.266	415.721	415.411	405.967	409.805	415.397
14	Indonesia	171.542	181.435	281.677	295.228	309.999	376.015	380.797	569.901	574.135	556.281	558.392	549.343
15	Iran	259.551	257.766	241.580	243.760	245.020	261.680	266.697	336.936	336.582	376.739	392.069	412.455
16	Iraq	76.112	76.891	87.633	88.704	90.916	105.360	107.367	123.145	122.732	129.053	134.285	142.130
17	Israel	789	988	2.426	2.490	2.841	4.035	4.097	10.166	11.171	13.850	14.947	16.400
18	Jepangg	18.560	18.311	16.890	17.054	17.031	18.743	19.239	34.412	36.084	39.918	43.663	56.458
19	Jordan	6.507	6.525	6.417	6.712	6.776	7.139	7.301	7.489	7.568	9.036	9.357	9.989
20	Kazakhstan	20.828	21.025	27.231	28.614	30.372	38.137	40.510	69.436	73.092	85.163	86.911	91.133
21	Kuwait	18.555	18.615	18.536	18.390	18.514	17.915	17.972	15.014	14.516	13.430	13.001	11.958
22	Kyrgyzstan	9.290	9.626	15.271	15.952	16.812	19.538	20.094	20.640	20.470	18.498	17.758	16.678
23	Laos	115	123	191	210	222	542	648	2.163	2.441	2.707	2.925	3.139
24	Lebanon	5.011	4.894	4.354	4.559	4.248	5.640	5.438	10.062	10.456	12.691	13.849	15.530
25	Macao	2	2	3	4	4	2	2	3	5	6	6	5
26	Malaysia	61.162	60.117	66.958	67.669	69.447	87.841	91.272	147.386	153.633	170.224	175.113	183.706
27	Maladewa	4.274	4.306	3.288	3.249	3.238	2.508	2.509	2.618	2.613	2.623	2.539	2.607
28	Mongolia	35.545	36.828	39.198	39.824	39.037	33.438	31.071	6.044	4.924	1.907	1.043	728
29	Myanmar	12.850	13.512	21.677	23.271	25.440	42.017	45.240	75.390	76.522	76.864	77.895	79.841
30	Nepal	49.555	45.794	27.716	26.179	25.690	26.573	26.712	27.661	27.757	28.836	29.444	31.014
31	Oman	29.617	29.617	33.668	29.009	28.638	23.793	23.087	21.792	21.792	14.219	14.101	13.848
32	Pakistan	32.936	32.921	32.319	32.621	33.299	37.499	38.622	53.623	54.122	59.899	56.952	62.723
33	Palestina	2.980	2.728	2.372	2.222	2.153	1.668	1.615	992	981	1.029	1.105	1.230
34	Philippina	51.464	55.293	53.815	52.708	51.689	49.835	49.247	55.204	54.449	56.307	54.783	61.920
35	Qatar	1.863	1.836	1.601	1.533	1.477	1.464	1.505	1.621	1.633	1.712	1.770	1.885
36	Korea Selatan	6.359	6.391	8.185	8.444	8.723	12.243	12.915	19.461	20.048	20.823	20.850	21.960
37	Saudi Arabia	11.322	11.331	12.199	11.970	11.773	10.805	10.510	10.742	10.829	11.136	11.380	11.355
38	Singapura	318	317	305	295	295	242	250	1.301	1.422	1.779	1.889	2.091
39	Sri Lanka	32.411	32.396	29.472	29.077	28.289	26.599	26.332	23.403	24.131	25.341	25.719	27.070
40	Syria	1.654	1.683	1.889	1.916	1.941	2.003	2.007	2.016	2.018	2.041	2.050	2.061
41	Taiwan	3.404	3.113	2.452	2.172	2.060	1.671	1.694	996	977	896	894	804
42	Tajikistan	52	59	131	139	146	196	211	347	367	397	412	413
43	Thailand	39.517	41.366	57.470	59.938	63.520	85.689	90.578	143.744	150.248	171.921	178.270	192.526
44	Timor-Leste	903	868	897	915	925	902	909	726	725	654	736	870
45	Turki	89.123	87.513	79.725	79.932	79.840	81.982	81.831	113.622	120.562	153.289	171.307	204.207
46	UAE	19.403	19.442	19.849	19.875	19.916	20.064	20.083	20.461	20.510	20.615	20.642	20.698
47	Uzbekistan	3.513	3.452	2.981	3.086	3.253	4.003	4.133	4.664	4.771	4.619	4.629	5.169
48	Vietnam	8.401	8.514	11.316	12.028	12.923	20.422	22.743	65.772	72.981	90.790	92.732	100.417
49	Yaman	1.554	1.543	1.478	1.464	1.459	1.441	1.444	1.468	1.469	1.480	1.485	1.512

2.2. Cyclical Rule

The Cyclical Order weight/bias method is one of the methods of the Artificial Neural Network. The Cyclical Order weight/bias method is an artificial neural network method that trains the network with heavy and biased learning rules with additional updates after the data presented is input. The input data is presented in a circular order [39]. The general syntax of the Cyclical Order weight/bias (trainc) method is:

$$[net, TR] = trainc(net, TR, trainV, valV, testV) \tag{1}$$

$$info = trainc('info') \tag{2}$$

Explanation :
trainc is not called directly. Instead it is called with training (train) to build a network whose net.trainFcn property is set to $'trainc'$.
$[net, TR]$ = $trainc(net, TR, trainV, valV, testV)$: Command to build network

Net	:	Neural Network
TR	:	Initial training notes created with train
$trainV$	:	Training data created with train
$valV$	:	Validation data created with train
$testV$	:	Test data created with train and back
net	:	Trained network
TR	:	Records of various grades training during the Epoch

2.3. Research Stages

The stages carried out in this study are presented in Figure 1. The first step taken from the research stage is to collect research datasets (based on Table 1). The next stage is to separate the research data into 2 (two) parts, namely the training and testing sections. The training data is based on data on active cases of the A1-A5 corona virus pandemic with a B1 training target. As for the test data based on data A6-A10 with a test target of B2. The next stage is to normalize the training and testing data using the equation (3) [40–43].

$$x' = \frac{0.8(x - a)}{b - a} + 0.1 \tag{3}$$

Where: x' is the result of normalized data, 0.8 and 0.1 are the default values, x (data to be normalized), a and b are the lowest values and the highest values of the research data used.

Data training that has been normalized so that training using the Matlab 2011b application can be processed, the next step is to create a multi-layer neural network by applying Back-propagation / cyclical rules and selecting functions. The network architecture model for the training process uses 5 forecasting models with 5 input layers, 1 hidden layer (5, 10, 15, 20 and 25 neurons) and 1 output layer. The manufacture of multi-layer traditional Back-propagation neural networks in the hidden layer uses the tansig activation function (sigmoid tangent) and the output layer uses logsig (binary sigmoid), while the multi-layer cyclical rule neural network construction in the hidden layer uses the activation function tansig (sigmoid tangent) and output layer using purelin (linear function). This section also applies a training function, with each training data being activated in turn by utilizing the traingd (gradient descent) function in traditional Back-propagation which will then be optimized using the cyclical rule (trainc) method training function. The next step is to generate IW and LW (weight) and bias (b) values. Furthermore, based on the training function used, initialization of network parameters is carried out. Then enter the command to carry out the training process and see when it finds performance (performance). If the results reach convergence, then the training will continue to enter the normalized test data. But if the training results have not reached convergence, then return to the stage of making a neural network with the application of Back-propagation / cyclical rule (implementation of the training function). The next stage is followed by simulation of test data based on the results of the training. everything has been done, the final stage is an evaluation to see whether the performance of the cycle rules can optimize the training that was previously carried out using traditional Back-propagation.

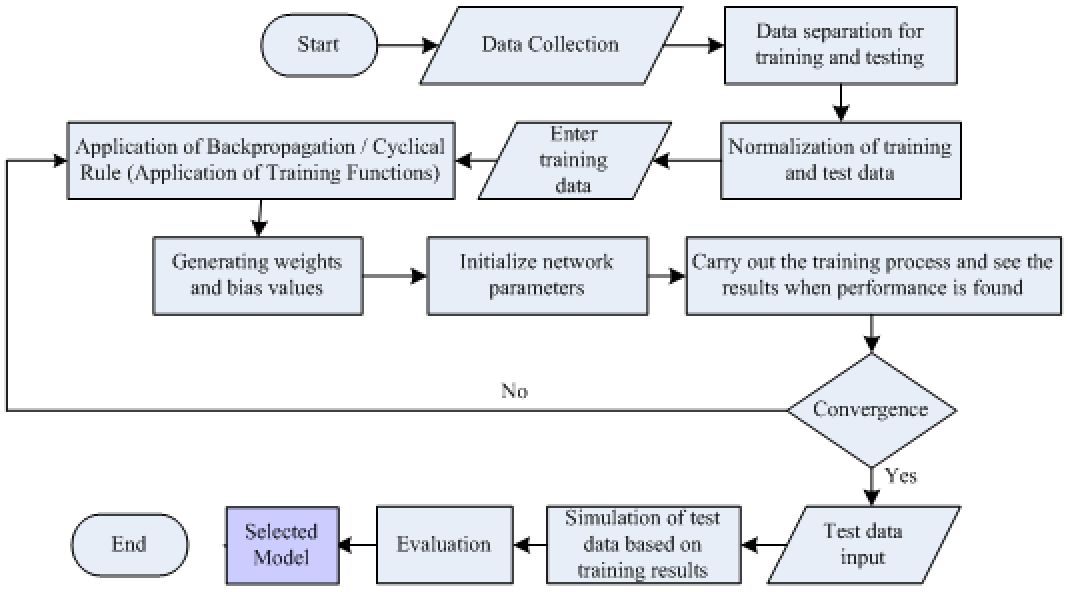

Start
Data Collection
Data separation for training and testing
Normalization of training and test data
Enter training data
Application of Backpropagation / Cyclical Rule (Application of Training Functions)
Generating weights and bias values
Initialize network parameters
Carry out the training process and see the results when performance is found
Convergence
No
Yes
Test data input
Simulation of test data based on training results
Evaluation
Selected Model
End

Figure 1. Research Stages

3. RESULT AND ANALYSIS

3.1. Activation Function and Training Parameters

Each use of the activation function and training parameters is generally processed and processed and processed using the 2011b version of the Matlab application. The program code syntax for each algorithm/method includes:

Traditional Back-propagation code syntax
Traditional Back-propagation Parameters % Enter training data % Enter testing data >> *p=[Normalized training data]* % Input Target Data Output >> *t=[Target training data]* >> *net = newff(minmax(p),[hidden layer,output layer],'tansig','logsig','traingd');* % Generating weight and bias >> *net.IW1,1;* >> *net.b1;* >> *net.LW2,1;* >> *net.b2;* % Back-propagation parameter value >> *net.trainparam.epochs = 100000;* >> *net.trainParam.goal = 0.001;* >> *net.trainParam.lr =0.01;* >> *net.trainParam.show = 1000;* % Conducted Training >> *net = train(net,p,t)* % View results when performance is found >> *[a,Pf,Af,e,perf] = sim(net,p,[],[],t)* % Enter data input (test) >> *p1=[Normalized test data]* % Entering target data (test) >> *t1=[Test data target]* % Simulation of the use of Test data is carried out based on the results of the Training >> *[a,Pf,Af,e,perf] = sim(net,p1,[],[],t1)*

Cyclical order code syntax
Cyclical order Parameters
% Enter training data
% Enter testing data
>> *p=[Normalized training data]*
% Input Target Data Output
>> *t=[Target training data]*
>> *net = newff(minmax(p),[hidden layer,output layer],'tansig','purelin','trainc');*
>> *net.IW1,1;*
>> *net.b1;*
>> *net.LW2,1;*
>> *net.b2;*
% Cyclical order parameter value
>> *net.trainParam.epochs = 2000;*
>> *net.trainParam.goal = 0.0001;*
>> *net.trainParam.max_fail = 6;*
>> *net.trainParam.show = 25;*
>> *net.trainParam.showCommandLine = false;*
>> *net.trainParam.showWindow = true;*
>> *net.trainParam.time = inf;*
% Conducted Training
>> *net = train(net,p,t)*
% View results when performance is found
>> *[a,Pf,Af,e,perf] = sim(net,p,[],[],t)*
% Enter data input (test)
>> *p1=[Normalized test data]*
% Entering target data (test)
>> *t1=[Test data target]*
% Simulation of the use of Test data is carried out based on the results of the Training
>> *[a,Pf,Af,e,perf] = sim(net,p1,[],[],t1)*

In the Traditional Back-propagation and Cyclical order program code syntax, it can be seen some similarities and some differences in the use of parameter codes. Some similarities that can be seen include: both must enter training data (p and t), both must build a network $(net = newff)$, both must generate weights and biases $(LWandb)$, both must using parameters $(net.trainparam)$, both must do training $(net = train)$, both must use code to see performance results when found, $([a, Pf, Af, e, perf] = sim(net, p, [], [], t))$, both must enter test data (p and t1), and both perform a simulation of the use of test data based on the results of the training. While the differences between Traditional Back-propagation and Cyclical order include: back-propagation uses the tansig and logsig activation functions while the cyclical order uses tansig and purelin. Back-propagation uses the traingd training function while the cyclical order uses trainc. Another difference from the cyclical order to optimize the performance of back-propagation is by changing the epochs value to 2000, the goal to 0.0001, adding the max_fail command with a value of 25, no need to use the *learning rate* (lr), changing the show value to 25, adding the parameter showCommandLine = false, add the parameters *showWindow = true and time = inf*. So that later it will produce faster and better convergence than traditional back-propagation.

3.2. Normalized Data for Training and Performance Testing

The research dataset based on Table 1 is normalized first using equation (1). Furthermore, it is divided into 2 parts, namely for training data and test data. Training data is taken from columns A1-A5 with the target of training B1. As for the test data based on data A6-A10 with a test target of B2. Normalization of training data can be seen in Table 2.

Table 2. Normalization Data for Training

X	A1	A2	A3	A4	A5	B1
1	0,15198	0,15337	0,15914	0,15960	0,16040	0,16168
2	0,10479	0,10478	0,10499	0,10503	0,10501	0,10540
3	0,10108	0,10105	0,10124	0,10130	0,10133	0,10194
4	0,10686	0,10621	0,10306	0,10290	0,10235	0,10148
5	0,18298	0,18683	0,21894	0,22386	0,22982	0,26816
6	0,10034	0,10035	0,10040	0,10037	0,10037	0,10039
7	0,10001	0,10001	0,10001	0,10001	0,10001	0,10002
8	0,10627	0,10618	0,10836	0,10882	0,10910	0,10978
9	0,10063	0,10062	0,10055	0,10055	0,10055	0,10060
10	0,10200	0,10221	0,10553	0,10623	0,10718	0,11372
11	0,11110	0,11120	0,11302	0,11302	0,11211	0,11610
12	0,10009	0,10010	0,10012	0,10012	0,10012	0,10010
13	0,90000	0,87760	0,73549	0,73140	0,70774	0,69111
14	0,32144	0,33421	0,46361	0,48110	0,50017	0,58538
15	0,43504	0,43274	0,41185	0,41466	0,41629	0,43779
16	0,19825	0,19925	0,21312	0,21450	0,21736	0,23600
17	0,10102	0,10127	0,10313	0,10321	0,10366	0,10521
18	0,12396	0,12363	0,12180	0,12201	0,12198	0,12419
19	0,10840	0,10842	0,10828	0,10866	0,10874	0,10921
20	0,12688	0,12714	0,13515	0,13693	0,13920	0,14923
21	0,12395	0,12403	0,12392	0,12374	0,12390	0,12312
22	0,11199	0,11242	0,11971	0,12059	0,12170	0,12522
23	0,10015	0,10016	0,10024	0,10027	0,10028	0,10070
24	0,10647	0,10631	0,10562	0,10588	0,10548	0,10728
25	0,10000	0,10000	0,10000	0,10000	0,10000	0,10000
26	0,17895	0,17760	0,18643	0,18735	0,18964	0,21339
27	0,10551	0,10556	0,10424	0,10419	0,10418	0,10323
28	0,14588	0,14754	0,15060	0,15141	0,15039	0,14316
29	0,11659	0,11744	0,12798	0,13004	0,13284	0,15424
30	0,16397	0,15911	0,13578	0,13379	0,13316	0,13430
31	0,13823	0,13823	0,14346	0,13744	0,13697	0,13071
32	0,14251	0,14249	0,14172	0,14211	0,14298	0,14840
33	0,10384	0,10352	0,10306	0,10287	0,10278	0,10215
34	0,16643	0,17137	0,16947	0,16804	0,16672	0,16433
35	0,10240	0,10237	0,10206	0,10198	0,10190	0,10189
36	0,10821	0,10825	0,11056	0,11090	0,11126	0,11580
37	0,11461	0,11462	0,11574	0,11545	0,11519	0,11395
38	0,10041	0,10041	0,10039	0,10038	0,10038	0,10031
39	0,14184	0,14182	0,13804	0,13753	0,13651	0,13433
40	0,10213	0,10217	0,10244	0,10247	0,10250	0,10258
41	0,10439	0,10402	0,10316	0,10280	0,10266	0,10215
42	0,10006	0,10007	0,10017	0,10018	0,10019	0,10025
43	0,15101	0,15340	0,17418	0,17737	0,18199	0,21061
44	0,10116	0,10112	0,10116	0,10118	0,10119	0,10116
45	0,21504	0,21297	0,20291	0,20318	0,20306	0,20583
46	0,12504	0,12509	0,12562	0,12565	0,12571	0,12590
47	0,10453	0,10445	0,10385	0,10398	0,10420	0,10516
48	0,11084	0,11099	0,11460	0,11552	0,11668	0,12636
49	0,10200	0,10199	0,10191	0,10189	0,10188	0,10186

Table 3. Normalization Data for Testing

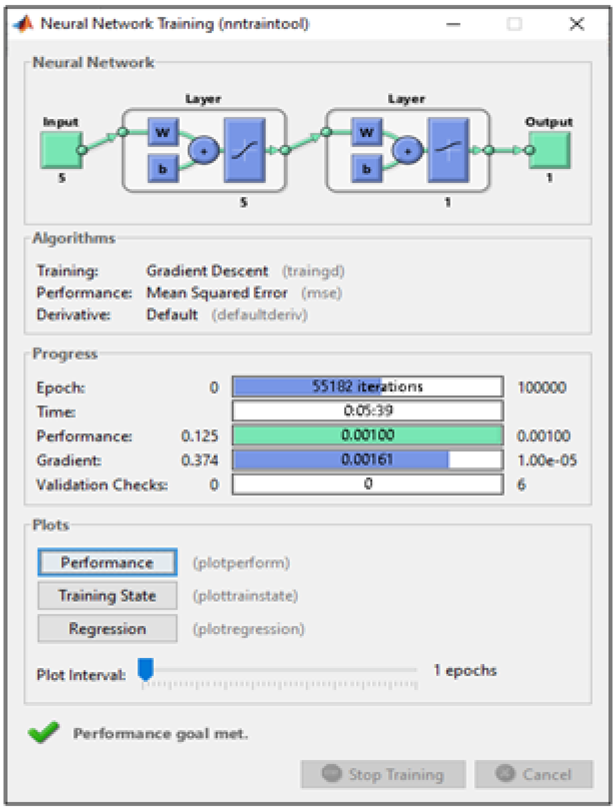
X	A6	A7	A8	A9	A10	B2
1	0,16612	0,16424	0,16355	0,16153	0,16083	0,15982
2	0,10582	0,10697	0,10709	0,10726	0,10751	0,10774
3	0,10213	0,10398	0,10438	0,10545	0,10630	0,10758
4	0,10155	0,10116	0,10114	0,10118	0,10124	0,10138
5	0,29061	0,30775	0,30338	0,31257	0,31609	0,31661
6	0,10042	0,10035	0,10036	0,10031	0,10023	0,10019
7	0,10002	0,10005	0,10007	0,10008	0,10008	0,10007
8	0,11040	0,10912	0,10878	0,10883	0,10853	0,10837
9	0,10066	0,10092	0,10095	0,10110	0,10120	0,10130
10	0,11597	0,12695	0,12723	0,12747	0,12722	0,12901
11	0,11728	0,13046	0,13288	0,13604	0,13829	0,14082
12	0,10011	0,10009	0,10009	0,10008	0,10008	0,10008
13	0,71207	0,67927	0,67883	0,66567	0,67102	0,67881
14	0,63060	0,89410	0,90000	0,87512	0,87806	0,86545
15	0,47161	0,56949	0,56899	0,62495	0,64631	0,67471
16	0,24960	0,27159	0,27101	0,27982	0,28711	0,29804
17	0,10571	0,11416	0,11556	0,11930	0,12082	0,12285
18	0,12680	0,14795	0,15028	0,15562	0,16084	0,17867
19	0,11017	0,11043	0,11054	0,11259	0,11304	0,11392
20	0,15644	0,19675	0,20184	0,21866	0,22110	0,22698
21	0,12504	0,12092	0,12022	0,11871	0,11811	0,11666
22	0,12800	0,12876	0,12852	0,12577	0,12474	0,12324
23	0,10090	0,10301	0,10340	0,10377	0,10407	0,10437
24	0,10757	0,11402	0,11457	0,11768	0,11929	0,12164
25	0,10000	0,10000	0,10000	0,10001	0,10001	0,10000
26	0,22718	0,30537	0,31407	0,33719	0,34400	0,35597
27	0,10349	0,10365	0,10364	0,10365	0,10354	0,10363
28	0,14329	0,10842	0,10686	0,10265	0,10145	0,10101
29	0,16303	0,20505	0,20662	0,20710	0,20854	0,21125
30	0,13722	0,13854	0,13867	0,14018	0,14102	0,14321
31	0,13217	0,13036	0,13036	0,11981	0,11965	0,11929
32	0,15381	0,17472	0,17541	0,18346	0,17935	0,18740
33	0,10225	0,10138	0,10136	0,10143	0,10154	0,10171
34	0,16862	0,17692	0,17587	0,17846	0,17633	0,18628
35	0,10209	0,10226	0,10227	0,10238	0,10246	0,10262
36	0,11799	0,12711	0,12793	0,12901	0,12905	0,13060
37	0,11464	0,11497	0,11509	0,11551	0,11585	0,11582
38	0,10035	0,10181	0,10198	0,10248	0,10263	0,10291
39	0,13669	0,13261	0,13362	0,13531	0,13583	0,13772
40	0,10279	0,10281	0,10281	0,10284	0,10285	0,10287
41	0,10236	0,10139	0,10136	0,10125	0,10124	0,10112
42	0,10029	0,10048	0,10051	0,10055	0,10057	0,10057
43	0,22621	0,30029	0,30935	0,33955	0,34840	0,36826
44	0,10126	0,10101	0,10101	0,10091	0,10102	0,10121
45	0,21402	0,25832	0,26799	0,31359	0,33870	0,38454
46	0,12798	0,12851	0,12858	0,12872	0,12876	0,12884
47	0,10576	0,10650	0,10665	0,10643	0,10645	0,10720
48	0,13169	0,19164	0,20169	0,22650	0,22921	0,23992
49	0,10201	0,10204	0,10204	0,10206	0,10207	0,10210

3.3. Training Results and Performance

The training process to see the ability / performance of the traditional Back-propagation algorithm and the cyclical rule algorithm, was carried out by utilizing the MATLAB 2011b application using 5 (five) network models: 5-5-1, 5-10-1, and 5-15-1.

1. Model Network 5-5-1

a. Traditional Back-propagation Algorithm
The display of the results of the 5-5-1 architectural model network training, can be seen in Figure 2. It is explained that the results of the training using the traditional Back-propagation algorithm with the tansig and logsig activation functions in the 5-5-1 model produce an epoch of 55182 iterations with training time for 5 (five) minutes 39 (thirty-nine) seconds. This means that the network convergence value of the 5-5-1 model occurs in the 55182 iteration. The results of the training and testing of the 5-5-1 model network on Traditional Back-propagation (training), are presented in Tables 4 and 5.
In Table 4 it can be seen that the training targets are obtained from the normalization table of training data. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE/Perf was obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0490073120 and an MSE/Perf of 0.0010001492.
In Table 5 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE was obtained from Street SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.1136168910 and an MSE/Perf of 0.0023187121. Note (information) is obtained from: If the value of Error $<= 0.01$ then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results Accuracy rate of 76% obtained from Total correct data / 49 $*$ 100.


Neural Network Training (nntraintool)
Neural Network
Input
Layer
W
b
5
Layer
W
b
1
Output
1
Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Stop Training
Cancel

```
Algorithms
Training:      Gradient Descent   (traingd)
Performance:   Mean Squared Error (mse)
Derivative:    Default  (defaultderiv)

Progress
Epoch:              0     55182 iterations   100000
Time:                     0:05:39
Performance:        0.125 0.00100            0.00100
Gradient:           0.374 0.00161            1.00e-05
Validation Checks:  0     0                  6

Plot Interval:  1 epochs

Performance goal met.
```

Figure 2. Model 5-5-1 Network Training on Traditional Back-propagation (trainingd)

Table 4. Model 5-5-1 Training Results (traingd)

X	Target	*Output*	Error	SSE
1	0,16168	0,13800	0,02368	0,0005608205
2	0,10540	0,12160	-0,01620	0,0002624076
3	0,10194	0,12100	-0,01906	0,0003631292
4	0,10148	0,12240	-0,02092	0,0004376741
5	0,26816	0,18210	0,08606	0,0074059420
6	0,10039	0,12100	-0,02061	0,0004249382
7	0,10002	0,12090	-0,02088	0,0004360010
8	0,10978	0,12160	-0,01182	0,0001395991
9	0,10060	0,12100	-0,02040	0,0004162023
10	0,11372	0,12060	-0,00688	0,0000473609
11	0,11610	0,12290	-0,00680	0,0000462436
12	0,10010	0,12090	-0,02080	0,0004325577
13	0,69111	0,73460	-0,04349	0,0018917083
14	0,58538	0,49430	0,09108	0,0082962880
15	0,43779	0,53950	-0,10171	0,0103444518
16	0,23600	0,19100	0,04500	0,0020253149
17	0,10521	0,12080	-0,01559	0,0002431704
18	0,12419	0,12570	-0,00151	0,0000022735
19	0,10921	0,12210	-0,01289	0,0001660763
20	0,14923	0,12600	0,02323	0,0005395090
21	0,12312	0,12580	-0,00268	0,0000071644
22	0,12522	0,12220	0,00302	0,0000091110
23	0,10070	0,12090	-0,02020	0,0004081584
24	0,10728	0,12200	-0,01472	0,0002167395
25	0,10000	0,12090	-0,02090	0,0004368100
26	0,21339	0,16000	0,05339	0,0028503580
27	0,10323	0,12180	-0,01857	0,0003446622
28	0,14316	0,13440	0,00876	0,0000767645
29	0,15424	0,12270	0,03154	0,0009945136
30	0,13430	0,14050	-0,00620	0,0000384436
31	0,13071	0,13280	-0,00209	0,0000043635
32	0,14840	0,13200	0,01640	0,0002690833
33	0,10215	0,12170	-0,01955	0,0003821794
34	0,16433	0,14740	0,01693	0,0002865549
35	0,10189	0,12130	-0,01941	0,0003768548
36	0,11580	0,12190	-0,00610	0,0000371912
37	0,11395	0,12360	-0,00965	0,0000932138
38	0,10031	0,12100	-0,02069	0,0004280840
39	0,13433	0,13210	0,00223	0,0000049875
40	0,10258	0,12120	-0,01862	0,0003465915
41	0,10215	0,12180	-0,01965	0,0003859472
42	0,10025	0,12090	-0,02065	0,0004264048
43	0,21061	0,13990	0,07071	0,0050000160
44	0,10116	0,12110	-0,01994	0,0003975325
45	0,20583	0,19920	0,00663	0,0000438977
46	0,12590	0,12610	-0,00020	0,0000000410
47	0,10516	0,12150	-0,01634	0,0002668397
48	0,12636	0,12200	0,00436	0,0000190058
49	0,10186	0,12120	-0,01934	0,0003741299
			Sum SSE	0,0490073120
			MSE/Perf	**0,0010001492**

Table 5. Model 5-5-1 Test Results (traingd)

Y	Target	*Output*	Error	SSE	Note
1	0,15982	0,14620	0,01362	0,0001854742	0
2	0,10774	0,12140	-0,01366	0,0001865475	1
3	0,10758	0,12020	-0,01262	0,0001592612	1
4	0,10138	0,12120	-0,01982	0,0003929086	1
5	0,31661	0,43080	-0,11419	0,0130402110	1
6	0,10019	0,12100	-0,02081	0,0004332508	1
7	0,10007	0,12090	-0,02083	0,0004337285	1
8	0,10837	0,12290	-0,01453	0,0002109940	1
9	0,10130	0,12090	-0,01960	0,0003843220	1
10	0,12901	0,12220	0,00681	0,0000463856	1
11	0,14082	0,12120	0,01962	0,0003850463	0
12	0,10008	0,12100	-0,02092	0,0004375539	1
13	0,67881	0,70420	-0,02539	0,0006444683	1
14	0,86545	0,67380	0,19165	0,0367315218	0
15	0,67471	0,52400	0,15071	0,0227147765	0
16	0,29804	0,33890	-0,04086	0,0016693829	1
17	0,12285	0,11870	0,00415	0,0000172147	1
18	0,17867	0,12380	0,05487	0,0030102882	0
19	0,11392	0,12180	-0,00788	0,0000621584	1
20	0,22698	0,15470	0,07228	0,0052247488	0
21	0,11666	0,12730	-0,01064	0,0001132191	1
22	0,12324	0,12790	-0,00466	0,0000217489	1
23	0,10437	0,12030	-0,01593	0,0002537295	1
24	0,12164	0,11950	0,00214	0,0000045659	1
25	0,10000	0,12090	-0,02090	0,0004366353	1
26	0,35597	0,36300	-0,00703	0,0000493627	1
27	0,10363	0,12140	-0,01777	0,0003157793	1
28	0,10101	0,13930	-0,03829	0,0014660006	1
29	0,21125	0,16030	0,05095	0,0025957079	0
30	0,14321	0,12960	0,01361	0,0001852944	0
31	0,11929	0,13170	-0,01241	0,0001539314	1
32	0,18740	0,14140	0,04600	0,0021156116	0
33	0,10171	0,12140	-0,01969	0,0003876527	1
34	0,18628	0,15000	0,03628	0,0013160115	0
35	0,10262	0,12110	-0,01848	0,0003413706	1
36	0,13060	0,12270	0,00790	0,0000623531	1
37	0,11582	0,12320	-0,00738	0,0000544743	1
38	0,10291	0,12040	-0,01749	0,0003058713	1
39	0,13772	0,12980	0,00792	0,0000626741	1
40	0,10287	0,12130	-0,01843	0,0003397010	1
41	0,10112	0,12150	-0,02038	0,0004154459	1
42	0,10057	0,12090	-0,02033	0,0004131995	1
43	0,36826	0,35980	0,00846	0,0000716389	1
44	0,10121	0,12120	-0,01999	0,0003996211	1
45	0,38454	0,31410	0,07044	0,0049618417	0
46	0,12884	0,12690	0,00194	0,0000037555	1
47	0,10720	0,12170	-0,01450	0,0002102580	1
48	0,23992	0,14080	0,09912	0,0098245364	0
49	0,10210	0,12120	-0,01910	0,0003646556	1
			Sum SSE	0,1136168910	**76%**
			MSE/Perf	**0,0023187121**	

b. Cyclical Rule Algorithm

The display of the results of the training of the 5-5-1 architectural model network with the Cyclical rule algorithm can be seen in Figure 3. It is explained that the results of the training using the Cyclical rule algorithm with the tansig and purelin activation functions in the 5-5-1 model produce 1534 iterations of epoch with training time is 4 (four) minutes and 5 (five) seconds. It means that the network convergence value of the 5-5-1 model occurs in the 1534 iteration.

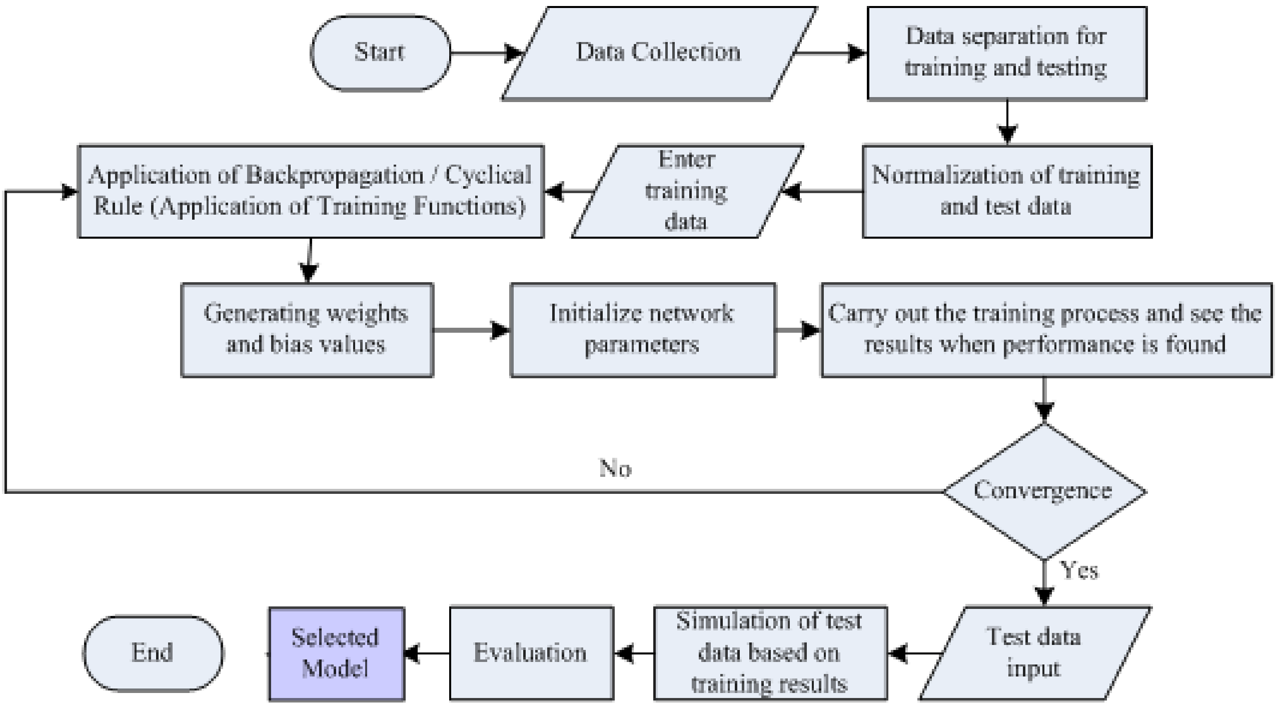

Start
Data Collection
Data separation for training and testing
Normalization of training and test data
Enter training data
Application of Backpropagation / Cyclical Rule (Application of Training Functions)
Generating weights and bias values
Initialize network parameters
Carry out the training process and see the results when performance is found
Convergence
No
Yes
Test data input
Simulation of test data based on training results
Evaluation
Selected Model
End

Figure 3. Model 5-5-1 Network Training on Cyclical rule (trainc)

The results of the training and testing of the 5-5-1 model network on the Cyclical rule (trainc), are presented in Table 6 and Table 5 below.

Table 6. Model 5-5-1 Training Results (trainc)

X	Target	*Output*	Error	SSE
1	0,16168	0,17120	-0,00952	0,0000905990
2	0,10540	0,10260	0,00280	0,0000078456
3	0,10194	0,09740	0,00454	0,0000206484
4	0,10148	0,10050	0,00098	0,0000009591
5	0,26816	0,23460	0,03356	0,0011261248
6	0,10039	0,09630	0,00409	0,0000166952
7	0,10002	0,09580	0,00422	0,0000178030
8	0,10978	0,10680	0,00298	0,0000089090
9	0,10060	0,09650	0,00410	0,0000168015
10	0,11372	0,10260	0,01112	0,0001236116
11	0,11610	0,11370	0,00240	0,0000057587
12	0,10010	0,09590	0,00420	0,0000176566
13	0,69111	0,69640	-0,00529	0,0000280240
14	0,58538	0,58070	0,00468	0,0000219395
15	0,43779	0,43940	-0,00161	0,0000025846
16	0,23600	0,22500	0,01100	0,0001210770
17	0,10521	0,09970	0,00551	0,0000303169
18	0,12419	0,12540	-0,00121	0,0000014588
19	0,10921	0,10720	0,00201	0,0000040519
20	0,14923	0,14100	0,00823	0,0000676890
21	0,12312	0,12790	-0,00478	0,0000228163
22	0,12522	0,12130	0,00392	0,0000153542
23	0,10070	0,09610	0,00460	0,0000211331
24	0,10728	0,10380	0,00348	0,0000120960
25	0,10000	0,09580	0,00420	0,0000176400
26	0,21339	0,19930	0,01409	0,0001984927
27	0,10323	0,10180	0,00143	0,0000020590
28	0,14316	0,16230	-0,01914	0,0003662809
29	0,15424	0,13150	0,02274	0,0005169216
30	0,13430	0,14410	-0,00980	0,0000960457
31	0,13071	0,15100	-0,02029	0,0004116398
32	0,14840	0,15040	-0,00200	0,0000039850
33	0,10215	0,09990	0,00225	0,0000050652
34	0,16433	0,18480	-0,02047	0,0004191055
35	0,10189	0,09870	0,00319	0,0000101586
36	0,11580	0,10980	0,00600	0,0000360185
37	0,11395	0,11700	-0,00305	0,0000093314
38	0,10031	0,09630	0,00401	0,0000160786
39	0,13433	0,14680	-0,01247	0,0001554193
40	0,10258	0,09910	0,00348	0,0000121315
41	0,10215	0,10010	0,00205	0,0000042208
42	0,10025	0,09600	0,00425	0,0000180661
43	0,21061	0,18760	0,02301	0,0005294965
44	0,10116	0,09730	0,00386	0,0000149134
45	0,20583	0,21380	-0,00797	0,0000635921
46	0,12590	0,13010	-0,00420	0,0000176615
47	0,10516	0,10100	0,00416	0,0000173453
48	0,12636	0,11480	0,01156	0,0001336236
49	0,10186	0,09840	0,00346	0,0000119547
			SSE SSE	0,0048892011
			MSE/Perf	0,0000997796

Table 7. Model 5-5-1 Test Results (trainc)

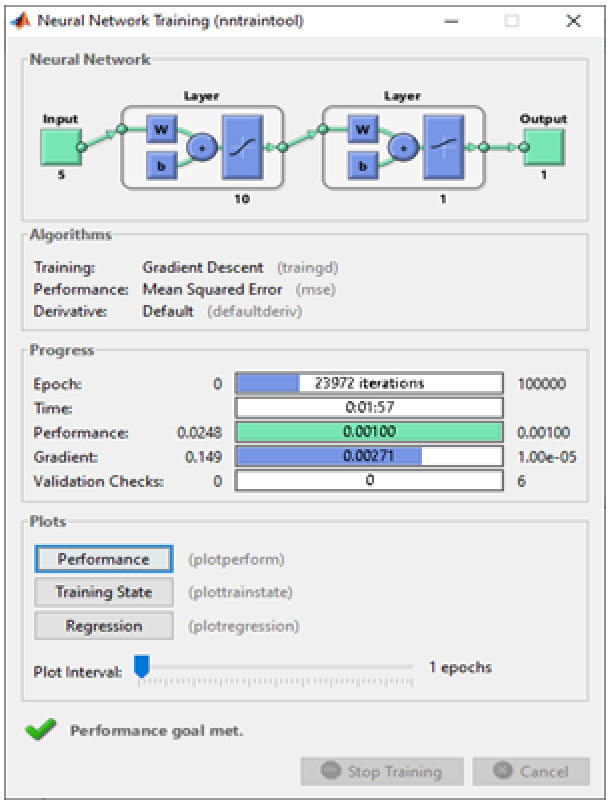
Y	Target	*Output*	Error	SSE	Note
1	0,15982	0,17550	-0,01568	0,0002458971	1
2	0,10774	0,10590	0,00184	0,0000033921	1
3	0,10758	0,10240	0,00518	0,0000268337	1
4	0,10138	0,09710	0,00428	0,0000183020	1
5	0,31661	0,28470	0,03191	0,0010180092	0
6	0,10019	0,09630	0,00389	0,0000150957	1
7	0,10007	0,09590	0,00417	0,0000174210	1
8	0,10837	0,10740	0,00097	0,0000009494	1
9	0,10130	0,09720	0,00410	0,0000167761	1
10	0,12901	0,13700	-0,00799	0,0000638289	1
11	0,14082	0,14450	-0,00368	0,0000135233	1
12	0,10008	0,09590	0,00418	0,0000174909	1
13	0,67881	0,67690	0,00191	0,0000036619	1
14	0,86545	0,65720	0,20825	0,0433700177	0
15	0,67471	0,64350	0,03121	0,0009743276	0
16	0,29804	0,26760	0,03044	0,0009267103	0
17	0,12285	0,12060	0,00225	0,0000050583	1
18	0,17867	0,16850	0,01017	0,0001033494	0
19	0,11392	0,11050	0,00342	0,0000116686	1
20	0,22698	0,23330	-0,00632	0,0000399118	1
21	0,11666	0,12140	-0,00474	0,0000224718	1
22	0,12324	0,13410	-0,01086	0,0001180172	1
23	0,10437	0,10120	0,00317	0,0000100560	1
24	0,12164	0,11830	0,00334	0,0000111342	1
25	0,10000	0,09580	0,00420	0,0000176751	1
26	0,35597	0,34010	0,01587	0,0002519883	0
27	0,10363	0,10090	0,00273	0,0000074519	1
28	0,10101	0,09040	0,01061	0,0001126063	0
29	0,21125	0,23210	-0,02085	0,0004348021	1
30	0,14321	0,14730	-0,00409	0,0000167094	1
31	0,11929	0,13330	-0,01401	0,0001961935	1
32	0,18740	0,19980	-0,01240	0,0001538647	1
33	0,10171	0,09720	0,00451	0,0000203500	1
34	0,18628	0,19380	-0,00752	0,0000565974	1
35	0,10262	0,09890	0,00372	0,0000138666	1
36	0,13060	0,13710	-0,00650	0,0000422969	1
37	0,11582	0,11630	-0,00048	0,0000002310	1
38	0,10291	0,09910	0,00381	0,0000145224	1
39	0,13772	0,13890	-0,00118	0,0000014002	1
40	0,10287	0,09960	0,00327	0,0000106865	1
41	0,10112	0,09720	0,00392	0,0000153469	1
42	0,10057	0,09650	0,00407	0,0000165868	1
43	0,36826	0,34290	0,02536	0,0006433312	0
44	0,10121	0,09700	0,00421	0,0000177197	1
45	0,38454	0,31430	0,07024	0,0049337056	0
46	0,12884	0,13430	-0,00546	0,0000298343	1
47	0,10720	0,10510	0,00210	0,0000044088	1
48	0,23992	0,24540	-0,00548	0,0000300436	1
49	0,10210	0,09860	0,00350	0,0000122783	1
			SSE SSE	0,0541084020	**82%**
			MSE/Perf	0,0011042531	

In Table 6 it can be seen that the training targets were obtained from the normalization table of training data. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE/Perf was obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0048892011 and an MSE/Perf of 0.0000997796.

In Table 7 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE was obtained from Street SSE divided by the number of data (Jl SSE / 49). This results in a total SSE of 0.0541084020 and an MSE/Perf of 0.0011042531. Note (information) is obtained from: If the value of Error $<= 0.01$ then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results Accuracy rate of 82% obtained from Total correct data / 49 $*$ 100.

2. Model Network 5-10-1

a. Traditional Back-propagation Algorithm
The display of the results of the 5-10-1 architectural model network training, can be seen in Figure 4. It is explained that the results of training using the traditional Back-propagation algorithm with the tansig and logsig activation functions in the 5-10-1 model produce epochs of 23972 iterations with training time for 1 (one) minute 57 (fifty seven) seconds. It means that the convergence value of the 5-10-1 model network occurs in the 23972 iteration. The results of training and testing of the 5-10-1 model network on Traditional Back-propagation (training), are presented in Table8 and Table 9. In Table 8 it can be seen that the training targets are obtained from the normalization table of training data. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE/Perf was obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0490077611 and an MSE/Perf of 0.0010001584. In Table 9 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The number of SSE is obtained from the total SSE as a whole. MSE was obtained from Street SSE divided by the number of data (Jl SSE / 49). This results in a total SSE of 0.0559678512 and an MSE/Perf of 0.0011422010. Note (information) is obtained from: If the value of Error $<= 0.01$ then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results Accuracy rate of 88% obtained from Total correct data / 49 $*$ 100.



Algorithms

Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

Progress

Epoch:	0	23972 iterations	100000
Time:		0:01:57	
Performance:	0.0248	0.00100	0.00100
Gradient:	0.149	0.00271	1.00e-05
Validation Checks:	0	0	6

Plots

Plot Interval: 1 epochs

Performance goal met.

Figure 4. Model Network Training 5-10-1 on Traditional Back-propagation (traingd)

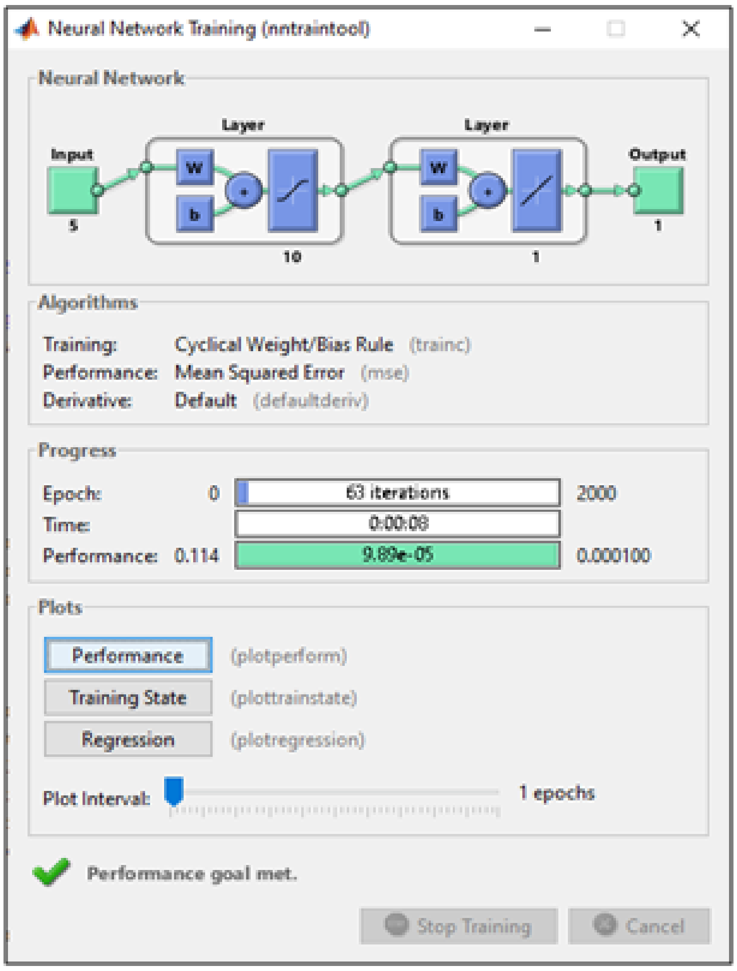
Table 8. Results of the 5-10-1 Model Training (traingd)

X	**Target**	***Output***	**Error**	**SSE**
1	0,16168	0,15020	0,01148	0,0001318283
2	0,10540	0,11290	-0,00750	0,0000562350
3	0,10194	0,11110	-0,00916	0,0000838314
4	0,10148	0,10960	-0,00812	0,0000659452
5	0,26816	0,24480	0,02336	0,0005455860
6	0,10039	0,11050	-0,01011	0,0001022936
7	0,10002	0,11030	-0,01028	0,0001056915
8	0,10978	0,11620	-0,00642	0,0000411548
9	0,10060	0,11060	-0,01000	0,0001000207
10	0,11372	0,11600	-0,00228	0,0000052072
11	0,11610	0,11750	-0,00140	0,0000019607
12	0,10010	0,11040	-0,01030	0,0001060492
13	0,69111	0,60800	0,08311	0,0069066450
14	0,58538	0,76590	-0,18052	0,0325860402
15	0,43779	0,46890	-0,03111	0,0009676877
16	0,23600	0,20660	0,02940	0,0008645657
17	0,10521	0,11320	-0,00799	0,0000639028
18	0,12419	0,12030	0,00389	0,0000151492
19	0,10921	0,11470	-0,00549	0,0000301078
20	0,14923	0,13760	0,01163	0,0001351949
21	0,12312	0,12230	0,00082	0,0000006779
22	0,12522	0,12640	-0,00118	0,0000013961
23	0,10070	0,11050	-0,00980	0,0000960974
24	0,10728	0,11270	-0,00542	0,0000293989
25	0,10000	0,11030	-0,01030	0,0001060900
26	0,21339	0,17420	0,03919	0,0015357577
27	0,10323	0,11160	-0,00837	0,0000699746
28	0,14316	0,14200	0,00116	0,0000013492
29	0,15424	0,13570	0,01854	0,0003435800
30	0,13430	0,11250	0,02180	0,0004752274
31	0,13071	0,13050	0,00021	0,0000000446
32	0,14840	0,13320	0,01520	0,0002311543
33	0,10215	0,11120	-0,00905	0,0000818918
34	0,16433	0,14990	0,01443	0,0002081652
35	0,10189	0,11100	-0,00911	0,0000830422
36	0,11580	0,11730	-0,00150	0,0000022454
37	0,11395	0,11840	-0,00445	0,0000198446
38	0,10031	0,11050	-0,01019	0,0001038400
39	0,13433	0,12720	0,00713	0,0000508836
40	0,10258	0,11170	-0,00912	0,0000831191
41	0,10215	0,11090	-0,00875	0,0000764844
42	0,10025	0,11050	-0,01025	0,0001050537
43	0,21061	0,18180	0,02881	0,0008300617
44	0,10116	0,11090	-0,00974	0,0000948329
45	0,20583	0,16980	0,03603	0,0012978392
46	0,12590	0,12390	0,00200	0,0000039898
47	0,10516	0,11200	-0,00684	0,0000467203
48	0,12636	0,12120	0,00516	0,0000266211
49	0,10186	0,11120	-0,00934	0,0000872811
			Sum SSE	0,0490077611
			MSE/Perf	0,0010001584

Table 9. Model Test Results 5-10-1 (traingd)

Y	Target	*Output*	Error	SSE	Note
1	0,15982	0,14330	0,01652	0,0002728738	0
2	0,10774	0,11440	-0,00666	0,0000443322	1
3	0,10758	0,11450	-0,00692	0,0000478847	1
4	0,10138	0,11080	-0,00942	0,0000887726	1
5	0,31661	0,33580	-0,01919	0,0003683998	1
6	0,10019	0,11040	-0,01021	0,0001043396	1
7	0,10007	0,11040	-0,01033	0,0001066294	1
8	0,10837	0,11390	-0,00553	0,0000305326	1
9	0,10130	0,11100	-0,00970	0,0000941702	1
10	0,12901	0,12860	0,00041	0,0000001687	1
11	0,14082	0,13880	0,00202	0,0000040909	1
12	0,10008	0,11040	-0,01032	0,0001064568	1
13	0,67881	0,66480	0,01401	0,0001963815	0
14	0,86545	0,72120	0,14425	0,0208094168	0
15	0,67471	0,75810	-0,08339	0,0069531881	1
16	0,29804	0,29240	0,00564	0,0000318312	1
17	0,12285	0,12620	-0,00335	0,0000112288	1
18	0,17867	0,15850	0,02017	0,0004066713	0
19	0,11392	0,11780	-0,00388	0,0000150859	1
20	0,22698	0,23810	-0,01112	0,0001236005	1
21	0,11666	0,11700	-0,00034	0,0000001159	1
22	0,12324	0,12190	0,00134	0,0000017860	1
23	0,10437	0,11320	-0,00883	0,0000779493	1
24	0,12164	0,12380	-0,00216	0,0000046794	1
25	0,10000	0,11030	-0,01030	0,0001060039	1
26	0,35597	0,46750	-0,11153	0,0124380178	1
27	0,10363	0,11210	-0,00847	0,0000717439	1
28	0,10101	0,10010	0,00091	0,0000008310	1
29	0,21125	0,21140	-0,00015	0,0000000231	1
30	0,14321	0,13370	0,00951	0,0000904836	1
31	0,11929	0,11460	0,00469	0,0000220251	1
32	0,18740	0,17670	0,01070	0,0001143997	0
33	0,10171	0,11080	-0,00909	0,0000826081	1
34	0,18628	0,16200	0,02428	0,0005893666	0
35	0,10262	0,11160	-0,00898	0,0000805725	1
36	0,13060	0,12970	0,00090	0,0000008035	1
37	0,11582	0,11850	-0,00268	0,0000071860	1
38	0,10291	0,11230	-0,00939	0,0000881566	1
39	0,13772	0,12930	0,00842	0,0000708408	1
40	0,10287	0,11170	-0,00883	0,0000779862	1
41	0,10112	0,11060	-0,00948	0,0000899176	1
42	0,10057	0,11070	-0,01013	0,0001025624	1
43	0,36826	0,46960	-0,10134	0,0102689902	1
44	0,10121	0,11070	-0,00949	0,0000900700	1
45	0,38454	0,39640	-0,01186	0,0001406515	1
46	0,12884	0,12550	0,00334	0,0000111417	1
47	0,10720	0,11360	-0,00640	0,0000409635	1
48	0,23992	0,27730	-0,03738	0,0013973542	1
49	0,10210	0,11130	-0,00920	0,0000845656	1
			Sum SSE	0,0559678512	**88%**
			MSE/Perf	0,0011422010	

b. Cyclical rule Algorithm
The display of the training results of the 5-10-1 architectural model network with the Cyclical rule algorithm, can be seen in Figure 5. It is explained that the results of the training using the Cyclical rule algorithm with the tansig and purelin activation functions in the 5-10-1 model produce an epoch of 1534 iterations with training time is 4 (four) minutes and 5 (five) seconds. This means that the network convergence value of the 5-10-1 model occurs in the 1534 iteration.



Training: Cyclical Weight/Bias Rule (trainc)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

Epoch: 0 63 iterations 2000
Time: 0:00:08
Performance: 0.114 9.89e-05 0.000100

(plotperform)
(plottrainstate)
(plotregression)

Plot Interval: 1 epochs

Performance goal met.

Figure 5. Model Network Training 5-10-1 on Cyclical rule (trainc)

The results of the training and testing of the 5-10-1 model network on the Cyclical rule (trainc), are presented in Tables 10 and 11. In Table 10 it can be seen that the training targets were obtained from the normalization table of the training data. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE/Perf is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0048486476 and an MSE/Perf of 0.0000989520. In Table 11 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.1229207220 and an MSE/Perf of 0.0025085862. Note (information) is obtained from: If the value of Error <= 0.01 then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results Accuracy rate of 92% obtained from Total correct data / 49 $*$ 100.

Table 10. Model 5-10-1 Training Results (trainc)

X	**Target**	***Output***	**Error**	**SSE**
1	0,16168	0,17220	-0,01052	0,0001106357
2	0,10540	0,10290	0,00250	0,0000062550
3	0,10194	0,09870	0,00324	0,0000105239
4	0,10148	0,09870	0,00278	0,0000077247
5	0,26816	0,28210	-0,01394	0,0001943854
6	0,10039	0,09740	0,00299	0,0000089160
7	0,10002	0,09700	0,00302	0,0000091166
8	0,10978	0,10890	0,00088	0,0000007829
9	0,10060	0,09760	0,00300	0,0000089938
10	0,11372	0,10750	0,00622	0,0000386645
11	0,11610	0,11170	0,00440	0,0000193576
12	0,10010	0,09710	0,00300	0,0000090119
13	0,69111	0,67090	0,02021	0,0004082916
14	0,58538	0,63050	-0,04512	0,0020354570
15	0,43779	0,40670	0,03109	0,0009667325
16	0,23600	0,24670	-0,01070	0,0001144151
17	0,10521	0,10190	0,00331	0,0000109302
18	0,12419	0,12230	0,00189	0,0000035804
19	0,10921	0,10750	0,00171	0,0000029342
20	0,14923	0,14920	0,00003	0,0000000007
21	0,12312	0,12460	-0,00148	0,0000021805
22	0,12522	0,12670	-0,00148	0,0000021950
23	0,10070	0,09740	0,00330	0,0000108707
24	0,10728	0,10340	0,00388	0,0000150383
25	0,10000	0,09700	0,00300	0,0000090000
26	0,21339	0,21000	0,00339	0,0000114836
27	0,10323	0,10130	0,00193	0,0000037439
28	0,14316	0,15930	-0,01614	0,0002604501
29	0,15424	0,14340	0,01084	0,0001174169
30	0,13430	0,13320	0,00110	0,0000012094
31	0,13071	0,13490	-0,00419	0,0000175470
32	0,14840	0,14790	0,00050	0,0000002538
33	0,10215	0,09980	0,00235	0,0000055253
34	0,16433	0,17400	-0,00967	0,0000935489
35	0,10189	0,09900	0,00289	0,0000083362
36	0,11580	0,11130	0,00450	0,0000202639
37	0,11395	0,11460	-0,00065	0,0000004287
38	0,10031	0,09740	0,00291	0,0000084670
39	0,13433	0,13810	-0,00377	0,0000141882
40	0,10258	0,10000	0,00258	0,0000066721
41	0,10215	0,09940	0,00275	0,0000075871
42	0,10025	0,09730	0,00295	0,0000087050
43	0,21061	0,21240	-0,00179	0,0000032013
44	0,10116	0,09840	0,00276	0,0000076274
45	0,20583	0,21890	-0,01307	0,0001709416
46	0,12590	0,12730	-0,00140	0,0000019672
47	0,10516	0,10180	0,00336	0,0000113217
48	0,12636	0,11900	0,00736	0,0000541633
49	0,10186	0,09910	0,00276	0,0000076041
			Sum SSE	0,0048486476
			MSE/Perf	0,0000989520

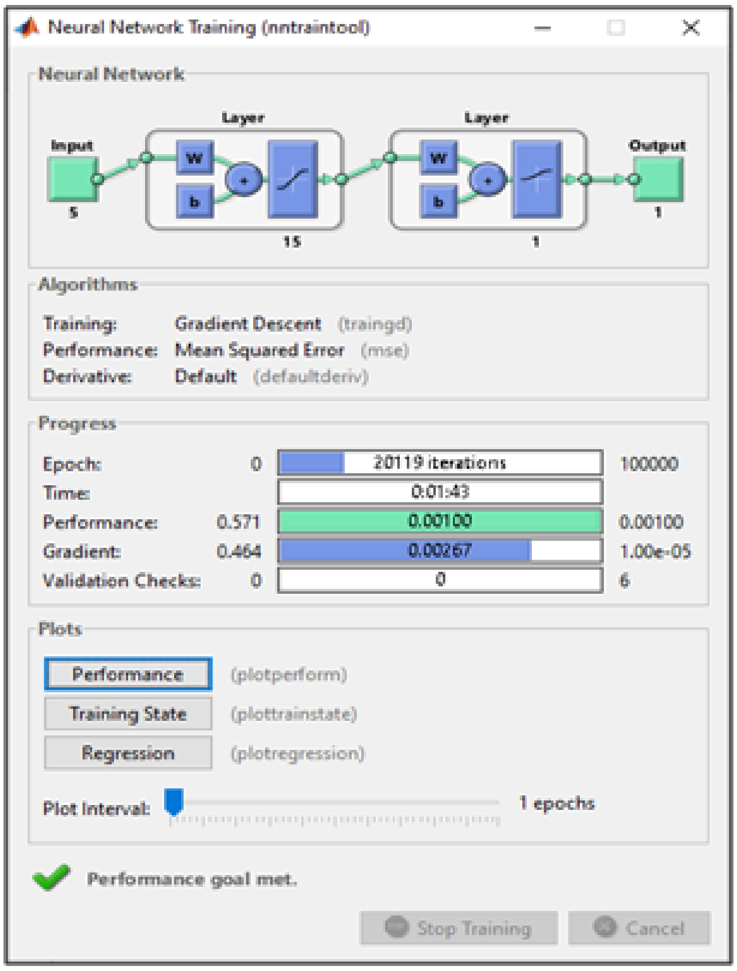
Table 11. Model Test Results 5-10-1 (trainc)

Y	Target	*Output*	Error	SSE	Note
1	0,15982	0,16730	-0,00748	0,0000559670	1
2	0,10774	0,10560	0,00214	0,0000045871	1
3	0,10758	0,10540	0,00218	0,0000047529	1
4	0,10138	0,09860	0,00278	0,0000077177	1
5	0,31661	0,33180	-0,01519	0,0002308498	1
6	0,10019	0,09720	0,00299	0,0000089122	1
7	0,10007	0,09710	0,00297	0,0000088438	1
8	0,10837	0,10710	0,00127	0,0000016240	1
9	0,10130	0,09850	0,00280	0,0000078169	1
10	0,12901	0,12760	0,00141	0,0000019901	1
11	0,14082	0,14590	-0,00508	0,0000257800	1
12	0,10008	0,09710	0,00298	0,0000088936	1
13	0,67881	0,60420	0,07461	0,0055671923	0
14	0,86545	0,54790	0,31755	0,1008409839	0
15	0,67471	0,61430	0,06041	0,0036498781	0
16	0,29804	0,31460	-0,01656	0,0002741701	1
17	0,12285	0,12550	-0,00265	0,0000070275	1
18	0,17867	0,17570	0,00297	0,0000087977	1
19	0,11392	0,11480	-0,00088	0,0000007816	1
20	0,22698	0,27000	-0,04302	0,0018505119	1
21	0,11666	0,11680	-0,00014	0,0000000197	1
22	0,12324	0,12200	0,00124	0,0000015288	1
23	0,10437	0,10180	0,00257	0,0000066106	1
24	0,12164	0,12260	-0,00096	0,0000009278	1
25	0,10000	0,09700	0,00300	0,0000090251	1
26	0,35597	0,40990	-0,05393	0,0029079985	1
27	0,10363	0,10100	0,00263	0,0000069160	1
28	0,10101	0,10710	-0,00609	0,0000370685	1
29	0,21125	0,23320	-0,02195	0,0004818863	1
30	0,14321	0,14710	-0,00389	0,0000151143	1
31	0,11929	0,10620	0,01309	0,0001714290	0
32	0,18740	0,20370	-0,01630	0,0002658277	1
33	0,10171	0,09910	0,00261	0,0000068179	1
34	0,18628	0,19110	-0,00482	0,0000232626	1
35	0,10262	0,09990	0,00272	0,0000074190	1
36	0,13060	0,13140	-0,00080	0,0000006458	1
37	0,11582	0,11590	-0,00008	0,0000000065	1
38	0,10291	0,10040	0,00251	0,0000063042	1
39	0,13772	0,14250	-0,00478	0,0000228800	1
40	0,10287	0,10030	0,00257	0,0000065999	1
41	0,10112	0,09850	0,00262	0,0000068514	1
42	0,10057	0,09770	0,00287	0,0000082523	1
43	0,36826	0,41250	-0,04424	0,0019568260	1
44	0,10121	0,09810	0,00311	0,0000096688	1
45	0,38454	0,38020	0,00434	0,0000188386	1
46	0,12884	0,13080	-0,00196	0,0000038498	1
47	0,10720	0,10400	0,00320	0,0000102382	1
48	0,23992	0,30590	-0,06598	0,0043535188	1
49	0,10210	0,09940	0,00270	0,0000073118	1
			Sum SSE	0,1229207220	**92%**
			MSE/Perf	0,0025085862	

3. Model Network 5-15-1

a. Traditional Back-propagation Algorithm

The display of the training results of the 5-15-1 architectural model network can be seen in Figure 6. It is explained that the results of the training using the traditional Back-propagation algorithm with the tansig and logsig activation functions in the 5-15-1 model produce an epoch of 20119 iterations with training time for 1 (one) minute 43 (forty three) seconds. It means that the network convergence value of the 5-15-1 model occurs in iteration 20119. In Table 12 it can be seen that the training targets are obtained from the normalization table of training data. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE/Perf is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0489988383 and an MSE/Perf of 0.0009999763. In Table 9 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.2029316320 and an MSE/Perf of 0.0041414619. Note (information) is obtained from: If the value of Error $<= 0.01$ then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results Accuracy rate of 76% obtained from Total correct data / 49 $*$ 100.


Neural Network Training (nntraintool)
Neural Network
Input
Layer
Layer
Output
5
15
1
1

Algorithms

Training: Gradient Descent (traingd)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

Progress

Epoch:	0	20119 iterations	100000
Time:		0:01:43	
Performance:	0.571	0.00100	0.00100
Gradient:	0.464	0.00267	1.00e-05
Validation Checks:	0	0	6

Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval:
1 epochs
Stop Training
Cancel

Performance goal met.

Figure 6. Model 5-15-1 Network Training on Traditional Back-propagation (traingd)

The results of the training and testing of the 5-15-1 model network on Traditional Back-propagation (training), are presented in Tables 12 and 13 below.

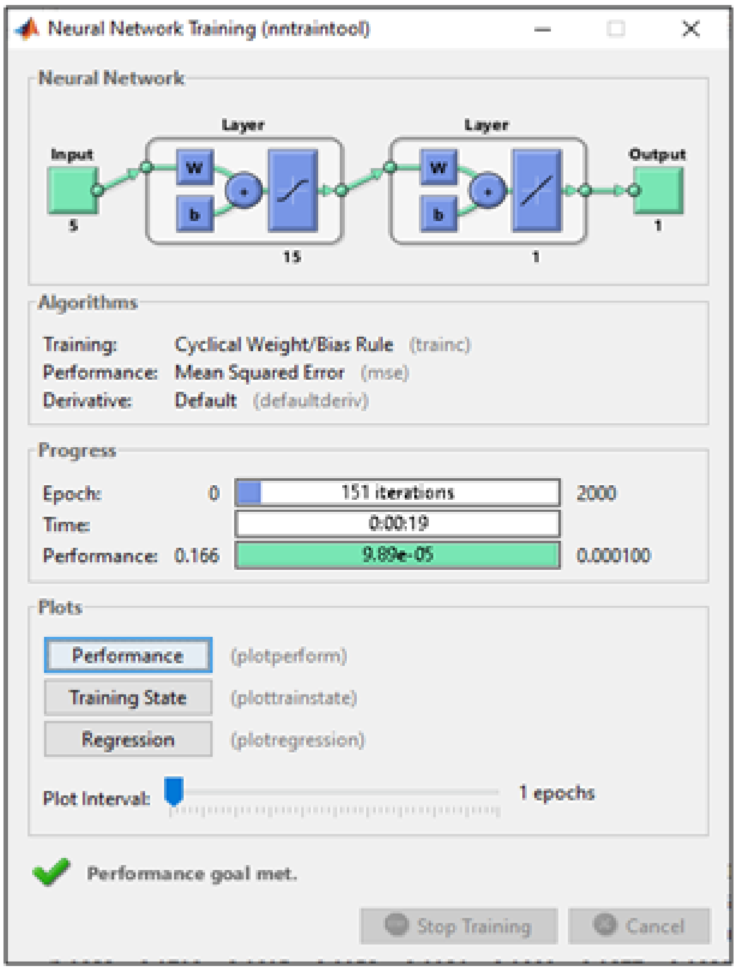
Table 12. Model 5-15-1 Training Results (traingd)

X	**Target**	***Output***	**Error**	**SSE**
1	0,16168	0,15660	0,00508	0,0000258232
2	0,10540	0,10820	-0,00280	0,0000078344
3	0,10194	0,10580	-0,00386	0,0000148683
4	0,10148	0,10670	-0,00522	0,0000272553
5	0,26816	0,22930	0,03886	0,0015099273
6	0,10039	0,10520	-0,00481	0,0000231749
7	0,10002	0,10500	-0,00498	0,0000248067
8	0,10978	0,11070	-0,00092	0,0000008376
9	0,10060	0,10530	-0,00470	0,0000220997
10	0,11372	0,10890	0,00482	0,0000232139
11	0,11610	0,11380	0,00230	0,0000052888
12	0,10010	0,10500	-0,00490	0,0000239906
13	0,69111	0,65390	0,03721	0,0013843034
14	0,58538	0,46940	0,11598	0,0134522793
15	0,43779	0,60310	-0,16531	0,0273266285
16	0,23600	0,24030	-0,00430	0,0000184599
17	0,10521	0,10710	-0,00189	0,0000035869
18	0,12419	0,12070	0,00349	0,0000121954
19	0,10921	0,11050	-0,00129	0,0000016565
20	0,14923	0,13200	0,01723	0,0002967810
21	0,12312	0,12220	0,00092	0,0000008526
22	0,12522	0,11900	0,00622	0,0000386690
23	0,10070	0,10510	-0,00440	0,0000193858
24	0,10728	0,10870	-0,00142	0,0000020223
25	0,10000	0,10500	-0,00500	0,0000250000
26	0,21339	0,19640	0,01699	0,0002886174
27	0,10323	0,10760	-0,00437	0,0000190539
28	0,14316	0,14750	-0,00434	0,0000188223
29	0,15424	0,12560	0,02864	0,0008200153
30	0,13430	0,13500	-0,00070	0,0000004904
31	0,13071	0,13770	-0,00699	0,0000488448
32	0,14840	0,13900	0,00940	0,0000884307
33	0,10215	0,10690	-0,00475	0,0000225569
34	0,16433	0,17000	-0,00567	0,0000321723
35	0,10189	0,10620	-0,00431	0,0000185998
36	0,11580	0,11220	0,00360	0,0000129711
37	0,11395	0,11570	-0,00175	0,0000030791
38	0,10031	0,10520	-0,00489	0,0000239140
39	0,13433	0,13530	-0,00097	0,0000009346
40	0,10258	0,10650	-0,00392	0,0000153426
41	0,10215	0,10680	-0,00465	0,0000215810
42	0,10025	0,10510	-0,00485	0,0000235183
43	0,21061	0,16980	0,04081	0,0016655207
44	0,10116	0,10570	-0,00454	0,0000205954
45	0,20583	0,24340	-0,03757	0,0014118404
46	0,12590	0,12370	0,00220	0,0000048287
47	0,10516	0,10740	-0,00224	0,0000049962
48	0,12636	0,11530	0,01106	0,0001223141
49	0,10186	0,10620	-0,00434	0,0000188568
			Sum SSE	0,0489988383
			MSE/Perf	0,0009999763

Table 13. Test Results Model 5-15-1 (trainigd)

Y	**Target**	***Output***	**Error**	**SSE**	**Note**
1	0,15982	0,16490	-0,00508	0,0000258177	1
2	0,10774	0,10960	-0,00186	0,0000034531	1
3	0,10758	0,10800	-0,00042	0,0000001763	1
4	0,10138	0,10570	-0,00432	0,0000186790	1
5	0,31661	0,46720	-0,15059	0,0226784755	1
6	0,10019	0,10520	-0,00501	0,0000251470	1
7	0,10007	0,10500	-0,00493	0,0000242669	1
8	0,10837	0,11090	-0,00253	0,0000063788	1
9	0,10130	0,10560	-0,00430	0,0000185256	1
10	0,12901	0,12310	0,00591	0,0000349364	1
11	0,14082	0,12790	0,01292	0,0001669935	0
12	0,10008	0,10500	-0,00492	0,0000241846	1
13	0,67881	0,56480	0,11401	0,0129991055	0
14	0,86545	0,52630	0,33915	0,1150259067	0
15	0,67471	0,51420	0,16051	0,0257648152	0
16	0,29804	0,36860	-0,07056	0,0049784431	1
17	0,12285	0,11580	0,00705	0,0000496893	1
18	0,17867	0,14110	0,03757	0,0014112113	0
19	0,11392	0,11260	0,00132	0,0000017317	1
20	0,22698	0,18880	0,03818	0,0014578975	0
21	0,11666	0,11940	-0,00274	0,0000075100	1
22	0,12324	0,12510	-0,00186	0,0000034729	1
23	0,10437	0,10720	-0,00283	0,0000080026	1
24	0,12164	0,11510	0,00654	0,0000427297	1
25	0,10000	0,10500	-0,00500	0,0000249582	1
26	0,35597	0,32260	0,03337	0,0011138331	0
27	0,10363	0,10730	-0,00367	0,0000134702	1
28	0,10101	0,11100	-0,00999	0,0000997679	1
29	0,21125	0,19280	0,01845	0,0003403320	0
30	0,14321	0,13580	0,00741	0,0000549420	1
31	0,11929	0,12460	-0,00531	0,0000281633	1
32	0,18740	0,17070	0,01670	0,0002787490	0
33	0,10171	0,10590	-0,00419	0,0000175469	1
34	0,18628	0,17850	0,00778	0,0000604798	1
35	0,10262	0,10640	-0,00378	0,0000142598	1
36	0,13060	0,12420	0,00640	0,0000409139	1
37	0,11582	0,11540	0,00042	0,0000001758	1
38	0,10291	0,10630	-0,00339	0,0000114865	1
39	0,13772	0,13230	0,00542	0,0000293406	1
40	0,10287	0,10680	-0,00393	0,0000154526	1
41	0,10112	0,10580	-0,00468	0,0000219257	1
42	0,10057	0,10530	-0,00473	0,0000223475	1
43	0,36826	0,32010	0,04816	0,0023197684	0
44	0,10121	0,10560	-0,00439	0,0000192767	1
45	0,38454	0,29370	0,09084	0,0082519677	0
46	0,12884	0,12620	0,00264	0,0000069586	1
47	0,10720	0,10920	-0,00200	0,0000040011	1
48	0,23992	0,16680	0,07312	0,0053463588	0
49	0,10210	0,10630	-0,00420	0,0000176061	1
			Sum SSE	0,2029316320	**76%**
			MSE/Perf	0,0041414619	

b. Cyclical rule Algorithm
The display of the results of the 5-15-1 architectural model network training with the Cyclical rule algorithm can be seen in Figure 7. It is explained that the training results using the Cyclical rule algorithm with the tansig and purelin activation functions in the 5-15-1 model produce 151 iterations of epoch with only 19 (nineteen) seconds of training time. It means that the network convergence value of the 5-15-1 model occurs in the 151 iteration.


Neural Network Training (nntraintool)

Neural Network

Algorithms

Training: Cyclical Weight/Bias Rule (trainc)
Performance: Mean Squared Error (mse)
Derivative: Default (defaultderiv)

Progress

Epoch:	0	151 iterations	2000
Time:		0:00:19	
Performance:	0.166	9.89e-05	0.000100

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

✓ Performance goal met.

Figure 7. Network Model 5-15-1 Training on Cyclical rule (trainc)

The results of the training and testing of the 5-15-1 model network on the Cyclical rule (trainc), are presented in Tables 14 and 15. In Table 14, it can be seen that the training targets were obtained from the training data normalization table. The output is obtained from the results of training using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE/Perf is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.0048387573 and an MSE/Perf of 0.0000987501. In Table 15 it can be explained that the test target is obtained from the normalization table of the test data. The output is obtained from the test results using Matlab. Error obtained from Target-Output. SSE is obtained from Error ^ 2. The SSE Sum is obtained from the total SSE as a whole. MSE is obtained from Sum SSE divided by the number of data (Sum SSE / 49). This results in a total SSE of 0.3061234306 and an MSE/Perf of 0.0062474170. Note (information) is obtained from: If the value of Error $<=$ 0.01 then the Note is worth 1 (True), whereas if not it will be worth 0 (False). Results The accuracy rate of 80% obtained from Total data is correct / 49 $*$ 100.

Table 14. Training Results Model 5-15-1 (trainc)

X	**Target**	***Output***	**Error**	**SSE**
1	0,16168	0,17050	-0,00882	0,0000777633
2	0,10540	0,10600	-0,00060	0,0000003588
3	0,10194	0,10150	0,00044	0,0000001972
4	0,10148	0,09810	0,00338	0,0000114199
5	0,26816	0,24950	0,01866	0,0003481129
6	0,10039	0,10010	0,00029	0,0000000818
7	0,10002	0,09960	0,00042	0,0000001759
8	0,10978	0,11330	-0,00352	0,0000123567
9	0,10060	0,10010	0,00050	0,0000002490
10	0,11372	0,11180	0,00192	0,0000036790
11	0,11610	0,11780	-0,00170	0,0000028909
12	0,10010	0,09980	0,00030	0,0000000912
13	0,69111	0,68070	0,01041	0,0001082896
14	0,58538	0,56050	0,02488	0,0006192115
15	0,43779	0,47290	-0,03511	0,0012325491
16	0,23600	0,24500	-0,00900	0,0000809370
17	0,10521	0,10630	-0,00109	0,0000011967
18	0,12419	0,12260	0,00159	0,0000025351
19	0,10921	0,10970	-0,00049	0,0000002372
20	0,14923	0,14940	-0,00017	0,0000000298
21	0,12312	0,12680	-0,00368	0,0000135177
22	0,12522	0,13260	-0,00738	0,0000544874
23	0,10070	0,10000	0,00070	0,0000004859
24	0,10728	0,10570	0,00158	0,0000024899
25	0,10000	0,09960	0,00040	0,0000001600
26	0,21339	0,20790	0,00549	0,0000301263
27	0,10323	0,10260	0,00063	0,0000004031
28	0,14316	0,15980	-0,01664	0,0002768385
29	0,15424	0,14490	0,00934	0,0000871592
30	0,13430	0,10390	0,03040	0,0009241424
31	0,13071	0,14820	-0,01749	0,0003058619
32	0,14840	0,14590	0,00250	0,0000062688
33	0,10215	0,10230	-0,00015	0,0000000223
34	0,16433	0,17100	-0,00667	0,0000445165
35	0,10189	0,10150	0,00039	0,0000001500
36	0,11580	0,11580	0,00000	0,0000000000
37	0,11395	0,11940	-0,00545	0,0000297541
38	0,10031	0,10000	0,00031	0,0000000960
39	0,13433	0,13770	-0,00337	0,0000113348
40	0,10258	0,10300	-0,00042	0,0000001739
41	0,10215	0,10160	0,00055	0,0000003074
42	0,10025	0,10000	0,00025	0,0000000627
43	0,21061	0,19470	0,01591	0,0002531533
44	0,10116	0,10110	0,00006	0,0000000038
45	0,20583	0,22200	-0,01617	0,0002616133
46	0,12590	0,12980	-0,00390	0,0000152300
47	0,10516	0,10340	0,00176	0,0000031144
48	0,12636	0,12250	0,00386	0,0000148963
49	0,10186	0,10170	0,00016	0,0000000248
			Sum SSE	0,0048387573
			MSE/Perf	0,0000987501

Table 15. Test Results Model 5-15-1 (trainc)

Y	Target	*Output*	Error	SSE	Note
1	0,15982	0,17040	-0,01058	0,0001119598	1
2	0,10774	0,10840	-0,00066	0,0000004333	1
3	0,10758	0,10670	0,00088	0,0000007746	1
4	0,10138	0,10090	0,00048	0,0000002286	1
5	0,31661	0,39580	-0,07919	0,0062716490	1
6	0,10019	0,10000	0,00019	0,0000000343	1
7	0,10007	0,09970	0,00037	0,0000001398	1
8	0,10837	0,10960	-0,00123	0,0000015022	1
9	0,10130	0,10100	0,00030	0,0000000875	1
10	0,12901	0,12800	0,00101	0,0000010215	1
11	0,14082	0,13610	0,00472	0,0000223029	1
12	0,10008	0,09970	0,00038	0,0000001461	1
13	0,67881	0,67030	0,00851	0,0000724817	1
14	0,86545	0,35130	0,51415	0,2643550497	0
15	0,67471	0,65800	0,01671	0,0002793652	0
16	0,29804	0,31900	-0,02096	0,0004392412	1
17	0,12285	0,12200	0,00085	0,0000007209	1
18	0,17867	0,14460	0,03407	0,0011604987	0
19	0,11392	0,11460	-0,00068	0,0000004679	1
20	0,22698	0,17520	0,05178	0,0026814194	0
21	0,11666	0,12110	-0,00444	0,0000197176	1
22	0,12324	0,12890	-0,00566	0,0000320761	1
23	0,10437	0,10490	-0,00053	0,0000002797	1
24	0,12164	0,11960	0,00204	0,0000041485	1
25	0,10000	0,09960	0,00040	0,0000001634	1
26	0,35597	0,28990	0,06607	0,0043657918	0
27	0,10363	0,10400	-0,00037	0,0000001370	1
28	0,10101	0,09310	0,00791	0,0000625936	1
29	0,21125	0,16600	0,04525	0,0020473897	0
30	0,14321	0,14420	-0,00099	0,0000009756	1
31	0,11929	0,12220	-0,00291	0,0000084501	1
32	0,18740	0,17280	0,01460	0,0002130367	0
33	0,10171	0,10120	0,00051	0,0000002612	1
34	0,18628	0,18030	0,00598	0,0000357230	1
35	0,10262	0,10250	0,00012	0,0000000153	1
36	0,13060	0,13070	-0,00010	0,0000000107	1
37	0,11582	0,11790	-0,00208	0,0000043292	1
38	0,10291	0,10300	-0,00009	0,0000000080	1
39	0,13772	0,14250	-0,00478	0,0000228800	1
40	0,10287	0,10300	-0,00013	0,0000000172	1
41	0,10112	0,10090	0,00022	0,0000000473	1
42	0,10057	0,10030	0,00027	0,0000000744	1
43	0,36826	0,29490	0,07336	0,0053822726	0
44	0,10121	0,10060	0,00061	0,0000003715	1
45	0,38454	0,28620	0,09834	0,0096708228	0
46	0,12884	0,13240	-0,00356	0,0000126884	1
47	0,10720	0,10750	-0,00030	0,0000000902	1
48	0,23992	0,14590	0,09402	0,0088395346	0
49	0,10210	0,10210	0,00000	0,0000000000	1
			Sum SSE	0,3061234306	**80%**
			MSE/Perf	0,0062474170	

4. Evaluation and Analysis

Based on the results of training and testing of the 5-5-1, 5-10-1, and 5-15-1 network models using the traditional Back-propagation algorithm (traind) with the activation functions of tansig and logsig, as well as the Cyclical rule (trainc) algorithm. With the activation function of tansig and purelin, it can be seen the ability, performance, accuracy of each algorithm and model, as presented in Table 16 below.

Table 16. Evaluation and analysis of capability, performance and accuracy of each algorithm and model

Algorithms	Models	Iterations	Times	Functions	Train Performances	Test Performances	Accuracy
Back-propagation Tradisional (traingd)	5-5-1	55182	5.39	*tansig, logsig*	0,0010001492	0,0023187121	76%
	5-10-1	23972	1.57		0,0010001584	0,0011422010	88%
	5-15-1	20119	1.43		0,0009999763	0,0041414619	76%
Cyclical rule (trainc)	5-5-1	1534	4.05	*tansig, purelin*	0,0000997796	0,0011042531	82%
	5-10-1	63	0.08		0,0000989520	0,0025085862	92%
	5-15-1	151	0.19		0,0000987501	0,0062474170	80%

Based on Table 16, it can be seen that of the 3 (three) models used with the Cyclical rule (trainc) algorithm training function combined with the tansig and purelin activation functions are able to perform better forecasting optimization than using the traditional Back-propagation algorithm training function (training). with the activation function of tansig and logsig. Its advantages can be seen from everything, including: smaller iterations, faster time, lower training performance, to a higher percentage of accuracy. Only the performance of the test is better than the traditional Back-propagation algorithm training function (training) with the tansig and logsig activation functions, but not too significant (not too much different). To further clarify the advantages of optimization with the Cyclical rule (trainc) algorithm training function, an accuracy graph will be presented which can be seen in Figure 8.

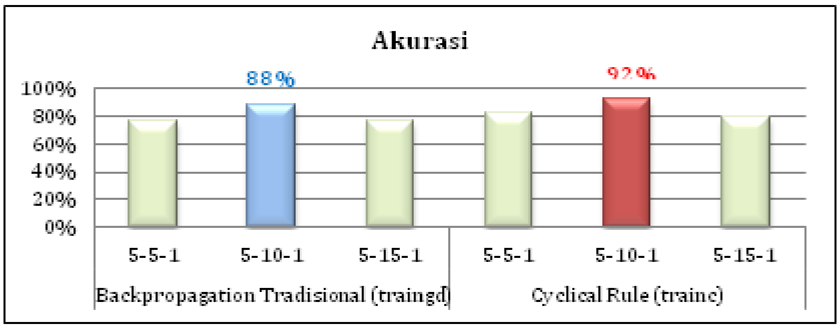


Figure 8. Graphics Accuracy

4. CONCLUSION

Based on the results and analysis, it is concluded that the novelty obtained from this study is in the form of the best forecasting model of the cyclical rule algorithm (cyclical order) with a 5-10-1 architecture that utilizes parameter changes and optimizes it to obtain better results. In addition, this research is different from previous research, both algorithms and science are used [34, 35], as well as the model and its parameters [36]. The cyclical rule algorithm in all architectural models used has better performance than traditional back-propagation, especially the 5-101 model with forecasting accuracy of 92% compared to 88% or 4% superior. So that it can be used and utilized to forecast the development of Covid-19 in Asia. The achievement of convergence and the training time of the cyclical rule (trainc) algorithm is faster with a smaller error rate than the traditional Back-propagation algorithm (training). But the parameters and training functions used also affect the performance of this algorithm. Suggestions for further research can use the training function with a different combination of parameters, such as changing the value of Maximum number of epochs to train, Performance goal, Maximum validation failures or the value of Epochs between displays. In addition, further research can also use different models with input layer and hidden layer values apart from this research.

5. ACKNOWLEDGEMENTS

The Acknowledgments section is optional. Research sources can be included in this section.

6. DECLARATIONS

AUTHOR CONTIBUTION

Anjar Wanto: Person in charge of research ideas and concepts, collects research data, develops theory and performs calculations and is responsible for writing research report papers, submitting them until they are published. Ni Luh Wiwik Sri Rahayu Ginantra: Validate research data, and separate data into two parts (training and test data), to determine and analyze a network architecture model that is suitable and appropriate for use. Surya Hendraputra: Look for and collect references related to the research topic and check the suitability of the grammar of the paper that has been written.Ika Okta Kirana: Analyze the results of each network architecture model used to choose the best one, then make a comparison chart of each architectural model used. Abdi Rahim Damanik: Normalize data, transform data and help check paper plagiarism using Turnitin.

FUNDING STATEMENT

Research funding was carried out independently by each author.

COMPETING INTEREST

We have no competing financial or personal relationship interests that could influence the work reported in this paper.

REFERENCES

[1] R. Lu, X. Zhao, J. Li, P. Niu, B. Yang, H. Wu, W. Wang, H. Song, B. Huang, N. Zhu, Y. Bi, X. Ma, F. Zhan, L. Wang, T. Hu, H. Zhou, Z. Hu, W. Zhou, L. Zhao, J. Chen, Y. Meng, J. Wang, Y. Lin, J. Yuan, Z. Xie, J. Ma, W. J. Liu, D. Wang, W. Xu, E. C. Holmes, G. F. Gao, G. Wu, W. Chen, W. Shi, and W. Tan, "Genomic Characterisation and Epidemiology of 2019 Novel Coronavirus: Implications for Virus Origins and Receptor Binding," *The Lancet*, vol. 6736, no. 20, pp. 1–10, 2020.

[2] Y. Yin and R. G. Wunderink, "MERS, SARS and Other Coronaviruses as Causes of Pneumonia," *Respirology*, vol. 23, no. 2, pp. 130–137, 2018.

[3] C. Drosten, S. Günther, W. Preiser, S. Van der Werf, H. R. Brodt, S. Becker, H. Rabenau, M. Panning, L. Kolesnikova, R. A. Fouchier, A. Berger, A. M. Burguière, J. Cinatl, M. Eickmann, N. Escriou, K. Grywna, S. Kramme, J. C. Manuguerra, S. Müller, V. Rickerts, M. Stürmer, S. Vieth, H. D. Klenk, A. D. Osterhaus, H. Schmitz, and H. W. Doerr, "Identification of A Novel Coronavirus in Patients with Severe Acute Respiratory Syndrome," *New England Journal of Medicine*, vol. 348, no. 20, pp. 1967–1976, 2003.

[4] A. M. Zaki, S. Van Boheemen, T. M. Bestebroer, A. D. Osterhaus, and R. A. Fouchier, "Isolation of A Novel Coronavirus From A Man with Pneumonia in Saudi Arabia," *New England Journal of Medicine*, vol. 367, no. 19, pp. 1814–1820, 2012.

[5] S. Zanganeh, N. Goodarzi, M. Doroudian, and E. Movahed, "Potential COVID-19 Therapeutic Approaches Targeting Angiotensin-Converting Enzyme 2; An Updated Review," *Reviews in Medical Virology*, vol. 32, no. 4, pp. 1–14, 2022.

[6] S. Alsammarraie and N. K. Hussein, "A New Hybrid Grasshopper Optimization - Backpropagation for Feedforward Neural Network Training," *Tikrit Journal of Pure Science*, vol. 25, no. 1, pp. 118–127, 2020.

[7] E. Bas, E. Egrioglu, and U. Yolcu, "A Hybrid Algorithm Based on Artificial Bat and Backpropagation Algorithms for Multiplicative Neuron Model Artificial Neural Networks," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–9, 2020.

[8] I. T. Sui Kim, V. Sethu, S. K. Arumugasamy, and A. Selvarajoo, "Fenugreek Seeds and Okra for The Treatment of Palm Oil Mill Effluent (POME) Characterization Studies and Modeling with Backpropagation Feed Forward Neural Network (BFNN)," *Journal of Water Process Engineering*, vol. 37, no. 101500, pp. 1–16, 2020.

[9] I. C. Afolabi, S. I. Popoola, and O. S. Bello, "Modeling Pseudo-Second-Order Kinetics of Orange Peel-Paracetamol Adsorption Process Using Artificial Neural Network," *Chemometrics and Intelligent Laboratory Systems*, vol. 203, no. 104053, pp. 1–47, 2020.

[10] Isha, A. S. Chaudhary, and D. K. Chaturvedi, “Effects of Activation Function and Input Function of ANN for Solar Power Forecasting,” in *Lecture Notes in Networks and Systems*, M. L. Kolhe, S. Tiwari, M. C. Trivedi, and K. K. Mishra, Eds. Springer, 2020, vol. 94, ch. Advances i, pp. 329–342.

[11] A. Panyafong, N. Neamsorn, and C. Chaichana, “Heat Load Estimation Using Artificial Neural Network,” *Energy Reports*, vol. 6, pp. 742–747, 2020.

[12] K. Kumar, V. Singh, and T. Roshni, “Efficacy of Hybrid Neural Networks in Statistical Downscaling of Precipitation of The Bagmati River Basin,” *Journal of Water and Climate Change*, vol. 11, no. 4, pp. 1302–1322, 2020.

[13] M. Žic, V. Subotić, S. Pereverzyev, and I. Fajfar, “Solving CNLS Problems Using Levenberg-Marquardt Algorithm: A New Fitting Strategy Combining Limits and a Symbolic Jacobian Matrix,” *Journal of Electroanalytical Chemistry*, vol. 866, no. 114171, pp. 1–9, 2020.

[14] J. Bilski, B. Kowalczyk, A. Marchlewska, and J. M. Zurada, “Local Levenberg-Marquardt Algorithm for Learning Feedforwad Neural Networks,” *Journal of Artificial Intelligence and Soft Computing Research*, vol. 10, no. 4, pp. 299–316, 2020.

[15] N. L. W. S. R. Ginantra, M. A. Hanafiah, A. Wanto, R. Winanjaya, and H. Okprana, “Utilization of the Batch Training Method for Predicting Natural Disasters and Their Impacts,” *IOP Conf. Series: Materials Science and Engineering*, vol. 1071, no. 1, p. 012022, 2021.

[16] R. Jayaseelan, G. Pandulu, and G. Ashwini, “Neural Networks for The Prediction of Fresh Properties and Compressive Strength of Flowable Concrete,” *Journal of Urban and Environmental Engineering*, vol. 13, no. 1, pp. 183–197, 2019.

[17] H. Espitia, I. Machon, and H. Lopez, “Control of A Microturbine Using Neural Networks,” in *Communications in Computer and Information Science*, J. C. Figueroa-García, M. Duarte-González, S. Jaramillo-Isaza, A. D. Orjuela-Cañon, and Y. D.-G. (Eds.), Eds. Springer, 2019, vol. 1052, ch. Applied Co, pp. 202–213.

[18] D. Gong, J. Feng, W. Xiao, and S. Sun, “Spectral Reconstruction Based on Bayesian Regulation Neural Network,” in *Smart Innovation, Systems and Technologies*, R. Kountchev, S. Patnaik, J. Shi, and M. N. Favorskaya, Eds. Springer, 2019, vol. 179, ch. Advances i, pp. 77–85.

[19] U. G. Inyang, E. E. Akpan, and O. C. Akinyokun, “A Hybrid Machine Learning Approach for Flood Risk Assessment and Classification,” *International Journal of Computational Intelligence and Applications*, vol. 19, no. 2, pp. 1–20, 2020.

[20] T. Afriliansyah and Z. Zulfahmi, “Prediction of Life Expectancy in Aceh Province by District City Using The Cyclical Order Algorithm,” *International Journal of Information System & Technology*, vol. 3, no. 2, pp. 268–275, 2020.

[21] G. S. Rao, S. S. Rani, and B. P. Rao, “Computed Tomography Medical Image Compression Using Conjugate Gradient,” *2019 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 169–173, 2019.

[22] Q. H. Nguyen, H. B. Ly, V. Q. Tran, T. A. Nguyen, V. H. Phan, T. T. Le, and B. T. Pham, “A Novel Hybrid Model Based on a Feed Forward Neural Network and One Step Secant Algorithm for Prediction of Load-Bearing Capacity of Rectangular Concrete-Filled Steel Tube Columns,” *Molecules*, vol. 25, no. 15, pp. 1–26, 2020.

[23] M. Zandieh, A. Azadeh, B. Hadadi, and M. Saberi, “Application of Artificial Neural Networks for Airline Number of Passenger Estimation in Time Series State,” *Journal of Applied Sciences*, vol. 9, no. 6, pp. 1001–1013, 2009.

[24] A. Perera, H. Azamathulla, and U. Rathnayake, “Comparison of Different Artificial Neural Network (ANN) Raining Algorithms to Predict The Atmospheric Temperature in Tabuk, Saudi Arabia,” *Journal MAUSAM*, vol. 71, no. 2, pp. 233–244, 2020.

[25] C. Perez, *Big Data and Deep Learning Examples with Matlab*. Lulu Press, Inc, 2020.

[26] P. Parulian, M. H. Tinambunan, S. Ginting, M. Khalil Gibran, A. Wanto, L. O. Muharram, N. Nurmawati, and G. W. Bhawika, “Analysis of Sequential Order Incremental Methods in Predicting The Number of Victims Affected by Disasters,” in *Journal of Physics: Conference Series*, vol. 1255, no. 1. Institute of Physics Publishing, sep 2019.

[27] C. K. Arthur, V. A. Temeng, and Y. Y. Ziggah, "Performance Evaluation of Training Algorithms in Backpropagation Neural Network Approach to Blast-Induced Ground Vibration Prediction," *Ghana Mining Journal*, vol. 20, no. 1, pp. 20–33, 2020.

[28] H. K. Ghritlahre and R. K. Prasad, "Prediction of Thermal Performance of Unidirectional Flow Porous Bed Solar Air Heater with Optimal Training Function Using Artificial Neural Network," *Energy Procedia*, vol. 109, pp. 369–376, 2017.

[29] E. Siregar, H. Mawengkang, E. B. Nababan, and A. Wanto, "Analysis of Backpropagation Method with Sigmoid Bipolar and Linear Function in Prediction of Population Growth," *Journal of Physics: Conference Series*, vol. 1255, no. 1, pp. 1–6, 2019.

[30] M. Tyrtaiou, A. Papaleonidas, A. Elenas, and L. Iliadis, "Accomplished Reliability Level for Seismic Structural Damage Prediction Using Artificial Neural Networks," in *Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference. EANN 2020. Proceedings of the International Neural Networks Society*, vol. 2. Springer International Publishing, 2020, pp. 85–98.

[31] B. Febriadi, Z. Zamzami, Y. Yunefri, and A. Wanto, "Bipolar Function in Backpropagation Algorithm in Predicting Indonesia's Coal Exports by Major Destination Countries," *IOP Conference Series: Materials Science and Engineering*, vol. 420, no. 1, p. 012087, 2018.

[32] N. Nasution, A. Zamsuri, L. Lisnawita, and A. Wanto, "Polak-Ribiere Updates Analysis with Binary and Linear Function in Determining Coffee Exports in Indonesia," *IOP Conference Series: Materials Science and Engineering*, vol. 420, no. 1, pp. 1–9, 2018.

[33] B. H. Hayadi, I. G. I. Sudipa, and A. P. Windarto, "Model Peramalan Artificial Neural Network pada Peserta KB Aktif Jalur Pemerintahan Menggunakan Artificial Neural Network Back-Propagation," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 21, no. 1, pp. 11–20, 2021.

[34] S. Mohan, A. John, A. Abugabah, M. Adimoolam, S. Kumar Singh, A. kashif Bashir, and L. Sanzogni, "An Approach to Forecast Impact of Covid-19 Using Supervised Machine Learning Model," *Software - Practice and Experience*, vol. 52, no. 4, pp. 824–840, 2022.

[35] R. Katoch and A. Sidhu, "An Application of ARIMA Model to Forecast The Dynamics of COVID-19 Epidemic in India," *Global Business Review*, vol. March, pp. 1–14, 2021.

[36] M. Shawaqfah and F. Almomani, "Forecast of The Outbreak of COVID-19 Using Artificial Neural Network: Case Study Qatar, Spain, and Italy," *Results in Physics*, vol. 27, no. June, p. 104484, 2021.

[37] Worldometer, "Reported Cases and Deaths by Country or Territory," 2021.

[38] A. L. Association, "Best Free Reference Web Sites 2011 13th Annual List RUSA Emerging Technologies in Reference Section (MARS)."

[39] M. O. Shabani and A. Mazahery, "Prediction Performance of Various Numerical Model Training Algorithms in Solidification Process of A356 Matrix Composites," *Indian Journal of Engineering and Materials Sciences*, vol. 19, no. 2, pp. 129–134, 2012.

[40] G. W. Bhawika, P. Purwantoro, A. D. GS, D. Sudrajat, A. Rahman, M. Makmur, R. A. Rohmah, and A. Wanto, "Implementation of ANN for Predicting The Percentage of Illiteracy in Indonesia by Age Group," *Journal of Physics: Conference Series*, vol. 1255, no. 1, pp. 1–6, 2019.

[41] M. K. Z. Sormin, P. Sihombing, A. Amalia, A. Wanto, D. Hartama, and D. M. Chan, "Predictions of World Population Life Expectancy Using Cyclical Order Weight / Bias," *Journal of Physics: Conference Series*, vol. 1255, no. 1, pp. 1–6, 2019.

[42] S. Setti and A. Wanto, "Analysis of Backpropagation Algorithm in Predicting The Most Number of Internet Users in The World," *JOIN (Jurnal Online Informatika)*, vol. 3, no. 2, pp. 110–115, 2018.

[43] A. Wanto, S. Defit, and A. P. Windarto, "Algoritma Fungsi Perlatihan pada Machine Learning berbasis ANN untuk Peramalan Fenomena Bencana," *RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 2, pp. 254–264, 2021.