Journal of Computing and Applied Informatics (JoCAI) Vol. 04, No. 1, 2020 | 27 – 46

talenta PUBLISHER

DATA SCIENCE
Journal of Computing and Applied Informatics

Adaptive Moment Estimation To Minimize Square Error In Backpropagation Algorithm

R N Singarimbun[1], E B Nababan[2*], and Opim Salim Sitompul[3]

[1]Graduate School of Computer Science
[2,3]Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

Abstract. Backpropagation Neural Network has weaknesses such as errors of gradient descent training slowly of error function, training time is too long and is easy to fall into local optimum. Backpropagation algorithm is one of the artificial neural network training algorithm that has weaknesses such as the convergence of long, over-fitting and easy to get stuck in local optima. Backpropagation is used to minimize errors in each iteration. This paper investigates and evaluates the performance of Adaptive Moment Estimation (ADAM) to minimize the squared error in backpropagation gradient descent algorithm. Adaptive Estimation moment can speed up the training and achieve the level of acceleration to get linear. ADAM can adapt to changes in the system, and can optimize many parameters with a low calculation. The results of the study indicate that the performance of adaptive moment estimation can minimize the squared error in the output of neural networks.

Keywords: Gradient Descent Backpropagation, Adaptive Moment Estimation, Minimize Square Error

Received 20 September 2019 | Revised 25 December 2019 | Accepted 29 January 2020

1. Introduction

ADAM is the Moment Adaptive Estimation, which is one method for optimizing parameters. In the study [1] using the algorithm ADAM compensated Asynchronous Delay (DC - Adam) to train Deep Neural Network (DNN). DC - ADAM can get a more accurate gradients and faster in training progress, and easy to implement with minimal memory requirements. This research [2] on online learning algorithm by using Group Method Of Data Handling Based Proportional - Integral - Derivative (GMDH - PID) for non-linear systems. With online setting method using GMDH - PID using Adaptive Estimation Moments (ADAM), which is one method of

*Corresponding author at:Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

E-mail address: ernabrn@usu.ac.id

ISSN: 2580-6769 | e-ISSN: 2580-829X | DOI : 10.32734/jocai.v4.i1-1160
Journal Homepage: https://jocai.usu.ac.id

optimization that can adapt to changes in the system and also can optimize many parameters with a low calculation.

A study about the Levenberg Marquardt - Backpropagation (LM - BP) Based Operation Quality Assessment Method For OTN (Optical Transmission Network) in the Smart Grid used in the smart grid to improve transmission speed and capacity efficient network transmission. Backpropagation Neural Network has weaknesses such as errors from gradient descent training on slowly of error function, training time is too long and is easy to fall into local optimum. While, LM algorithm convergence speed and robustness of the best on the network that are robustness resistant. The results of this study with the backpropagation algorithm Levenberg Marquardt (LM) have more optima prediction accuracy, have a better network structure and accuracy errors increased significantly from the standard back propagation neural network [3].

Artificial Neural Network (ANN) has a problem in determining the weight to the right network. A study comparing the Gradient Descent and Genetic Algorithm (GA) based training Artificial Neural Network (ANN). GA slightly better in Mean Square Error (MSE) in cancer datasets for classification errors are average, but better Gradient Descent in the dataset diabetes. From this study, it is still necessary to experiment with more datasets in ANN training [4].

In the study to improve the gradient descent in artificial neural network such as quickprop, backpropagation, Delta-Bar-Delta and Super SAB as the approximate error of function with quadratic polynomial to get the minimum squared error function. The partial derivative method until the process of update weights in the gradient descent on backpropagation can be modified to the level of learning at each weight to the neurons in the network. Improved gradient descent is better than standard gradient descent and gradient descent momentum [5].

In the study to evaluate and test the three gradient descent based on backpropagation to classify benign and malignant tumors in ultrasound imaging. In selecting the right learning rate, time complexity and network model are still important in the network system at the time of convergence in the classification process. Gradient descent (GD), the gradient descent with momentum (GDM) and adaptive gradient descent (AGD) is used for training the classification model for testing and validation. The results of this study, backpropagation based AGD is better in the process of classifying benign and malignant tumors, but AGD algorithms are very long in time complexity [6].

In the study of artificial neural networks, mean square error (MSE) is a problem that is used in learning. The training uses a theoretical backpropagation method, for correntropy conjugate gradient-based BP (CCG-BP). CCG - BP gets better results than MSP-based correntropy-based backpropagation and can minimize MSE [7]. A study of [8] to improve the convergence and global search capabilities of the network backpropagation (BP). BP has weaknesses such as long

convergence, over-fitting and easily trapped to local optima. Genetic Algorithm (GA) can improve BP by evaluating natural selection, genetic crossover and mutation gen that have advantages such as very high parallels, stochastic and global probability searches that can overcome BP shortcomings. The result is that GA can set specific targets in starting BP weight, adjusting BP training so that the epoch is smaller.

In the study using Backropagation Modified Adaptive Approach (AMBP) in improving the performance of the Modern Artificial Intelligence algorithm to accelerate convergence by adjusting the learning rate at each layer and epoch. AMBP algorithm with a learning rate variable for the process of classifying data quickly and getting a small MSE in a short time. To improved the learning rate with momentum can be added to the network, this algorithm can be known as the backpropagation with momentum algorithm (BPM). The results of previous studies, AMBP are much better and able to provide MSE that is less than SBP and BPM [9].

Artificial Neural Networks (ANN) using levenberg - marquardt training to optimize weights on ANN. Statistical methods and ANN are methods used to predict. However, ANN does not make the basic structure of the system compared to the statistical method. ANN is also a linear regression that is complex, nonlinear and dynamic. The levenberg - marquardt algorithm is close to the speed of training, so that the performance function will always be reduced at each algorithm iteration. The Levenberg - Marquardt algorithm is the fastest method for training artificial neural networks to several hundred weights. The result is that the Levenberg-Marquardt algorithm has errors that are relatively less than 3% Mammadli [10].

In the study of Optimizing the Backpropagation by using the Nguyen – Windrow method on the input layer of the feed – forward process and adjusting the learning rate parameter in the backward process. The influence of learning with adaptive learning rate changes using a randomly selected initial weight with the Nguyen – Windrow method. Backpropagation is used to minimize error in every iteration. The result in the feed – forward phase with Nguyen – Windrow's initial weight method was able to give close value to the error value affecting the weight update to the backward phase. In the results of the backward phase adaptive learning rate parameters can be pain number of iterations (epoch) [11].

In this paper, the learning process of the backpropagation algorithm is still slow in training gradient descent from error functions and requires a very long processing time. This is because the architecture, learning rate, overfitting and the number of epochs in the training are still high so that the solution is easy to fall into the optimum local. Gradient descent backpropagation is also still not good at minimizing squared errors, so a suitable approach is needed in order to improve the gradient descent backpropagation learning process. The performance of Adaptive Estimation moment to minimize the squared error in backpropagation gradient descent algorithm.

ADAM can update the parameters of the output torque which is a torque distribution first and second moments in backpropagation gradient descent. The purpose of this study is to minimize the squared error at each iteration (epoch) at the output of the neural network. The results showed that ADAM can minimize the squared error at each iteration (epoch) at the output of neural networks.

2. Adaptive Moment Estimation And Gradient Descent Backpropagation Algorithms

2.1 Adaptive Moment Estimation

ADAM is used for optimizing a gradient descent learning algorithm to minimize the objective function (often called the loss function E (x)) on various parameters such as weights and biases. Error in backpropagation is the mechanism used to modify network parameters before initialization parameters to get optimized and can produce output is approaching the target output. In the error back propagation neural network used, the process of calculating the feedforward output one by one and calculate the error component obtained in the last layer. Gradient is calculated on backpropagation to get the network to be optimized.Here are the steps - steps ADAM as follows [12]:

- Initialization m_{weight} (t), m_{bias} (t), v_{weight} (t) and v_{bias} (t) = 0
  If the first iteration is t = 1, t = 1-1 = 0 (time step / early iterations on the input)
- Gradient calculation for estimating the first moment in time step = $g_t \frac{\partial E}{\partial W_{ik}^o}$
- The calculation of the estimated first moment (**mt**) weight and bias can be done after receiving the derivative calculation squared error in the output layer by the following equation:

$$m_{weight(t)} = \beta_1 * m_{weight\,(t-1)} + (1-\beta_1) * g_t \quad (1)$$

$$\mathrm{m_{bias(t)}} = \beta_1 * \mathrm{m_{bias\,(t-1)}} + (1-\beta_1) * \mathrm{g_t}$$

- The calculation of weight and bias correction estimation of the first moment $\acute{m}_t$ by the following equation:

$$\acute{m}_{weight(t)} = \frac{m_{weight(t)}}{1-\beta_1^t} \quad (2)$$

$$\acute{m}_{bias(t)} = \frac{m_{bias(t)}}{1-\beta_1^t}$$

- Gradient calculation for estimating the first moment in time step = $g_t \frac{\partial E}{\partial W_{ij}^h}$

- The calculation of the estimated second moment (**vt**) weight and bias can be done after receiving the derivative calculation output to the hidden layer by the following equation:

$$v_{weight(t)} = \beta_2 v_{weight(t-1)} + (1-\beta_2) * (g_t)^2 \quad (3)$$

$$v_{bias(t)} = \beta_2 v_{bias(t-1)} + (1-\beta_2) * (g_t)^2$$

- The calculation of weight and bias correction the estimated second moment $\tilde{\nabla}_t$ by the following equation:

$$\tilde{\nabla}_{weight(t)} = \frac{v_{weight(t)}}{1-\beta_2^t} \quad (4)$$

$$\tilde{\nabla}_{bias(t)} = \frac{v_{bias(t)}}{1-\beta_2^t}$$

- Parameter updates weight and bias by the following equation:

$$w_{weight(t)} = w_{weight(t-1)} - \alpha * \frac{\acute{m}_{weight(t)}}{\sqrt{\tilde{\nabla}_{weight(t)} + \varepsilon}} \quad (5)$$

$$w_{bias(t)} = w_{bias(t-1)} - \alpha * \frac{\acute{m}_{bias(t)}}{\sqrt{\tilde{\nabla}_{bias(t)} + \varepsilon}}$$

2.2 Gradient Descent Backpropagation

Backpropagation gradient descent algorithm to minimize Square Error (SE) for the multilayer feedforward neural network. The learning rule to change the weights and bias on the output neuron layer and hidden layer neurons The following steps - steps in the gradient descent backpropagation ADAM [13]:

- Initialize the weights randomly on each neuron located in the input layer, hidden layer and output layer.
- Phase feed forward propagation:
  1. Calculate each neuron in the hidden layer to the equation:

$$net_{ij}^h = \sum_{i=1} X_i W_{ij}^h + w_{bias\_j}^h \quad (6)$$

  2. Calculate each neuron's activation function in the hidden layer with sigmoid equation:

$$f_{in_{net}n} = \frac{1}{1+e^{-(net_{ij}^h)}} \quad (7)$$

  3. Calculates the total value of the output layer to the equation:

$$net_{ik}^O = \sum_{i=1}^{n} (f_{in\_net\ n} * w_{ik}^O) + w_{bias\_k}^O \quad (8)$$

  4. Calculates the sigmoid activation function in the output layer to the equation:

$$f_{out\_net\ n} = \frac{1}{1+e^{-(net_{ik}^O)}} \quad (9)$$

  5. Calculates the error in the output layer based on the difference between the target and output by the equation:

$$e_{output} = \text{Target} - f_{out\_net\ n} \quad (10)$$

  6. Calculates the square error at the output layer to the equation:

$$\text{Square Error} = \frac{1}{2} * \sum_{k} (\text{Target} - f_{out\_net\,n})^2 \quad (11)$$

7. Calculates the partial derivative of the weight and bias for each neuron in the output layer with the equation:

$$\frac{\partial E}{\partial W_{ik}^{o}} = -(\text{Target} - f_{out\_net\,n}) * f'_{out\_net\,n} * f_{in_{net}n} \quad (12)$$

$$\frac{\partial E}{\partial W_{bias}^{o}} = -(\text{Target} - f_{out\_net\,n}) * f'_{out\_net\,n}$$

8. Calculates the partial derivative of the weight and the bias for each neuron in the hidden layer with the equation:

$$\frac{\partial E}{\partial W_{weight\,ij}^{h}} = -\sum_{k} (\text{T} - f_{out\_net\,n}) * f'_{out\_net\,n} * w_{ik}^{o} * f'_{in_{net}n} * \text{X}_i \quad (13)$$

$$\frac{\partial E}{\partial W_{bias\,i}^{h}} = -\sum_{k} (\text{T} - f_{out\_net\,n}) * f'_{out\_net\,n} * w_{ik}^{o} * f'_{in_{net}n}$$

3. Methodology

The methodology is divided into two parts on the backpropagation network architecture, that is, giving a feed forward for weighting and part of the backward feed for error values. Starting from output, so that each neuron has a corresponding error value that roughly represents its contribution to the original output. The steps to make a research design are as follows:

- Prepare data as enter 699 data which has 9 variables and 1 target variable for class.
- The pattern of designing network architecture is the number of neurons in the input layer, the number of neurons in the hidden layer and the number of neurons in the output layer.
- Run backpropagation with adaptive moment estimation with random initial weights.
- Analysis.

3.1 Data Input

The data used in this research is data about Wisconsin Breast Cancer dataset from the University of California Irvine (UCI) Machine Learning Repository. Data has 9 attributes were rated visually with the appropriate class variables and defined for each record in the dataset. All values on 9 attributes are indexed from 1 - 10 interval ranges, while the range class value on breast cancer cells is 2 for the benign category and 4 for the malignant category. The following Table 1 descriptions of datasets WBCD:

Table 1. Descriptions of datasets WBCD

Attribute	Range interval
Clump Thickness	1-10
Uniformity of Cell Size	1-10
Uniformity of Cell Shape	1-10
marginal Adhesion	1-10
Single Epithelial Cell Size	1-10
Bare Nuclei	1-10
Bland Chromatin	1-10
normal Neucleoli	1-10
Mitoses	1-10
Class variable	Benign cells (2) and malignant cells (4)

3.2 Block Diagram

The following figure 1 is a block diagram which aims to research done in the process is not out of the specified path. Block diagram shown in Figure 1.

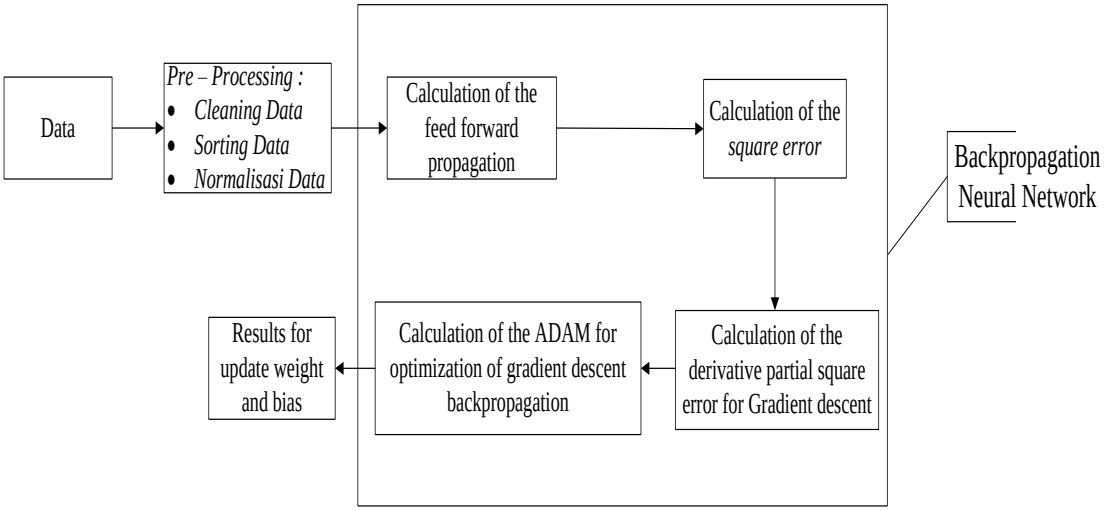


Figure 1. Block diagram

Figure 1 can be explained that the block diagram above is the backpropagation neural network architecture with multiple processes, namely:

- The data used is the data derived from Wisconsin Breast Cancer dataset from the University of California Irvine (UCI) Machine Learning Repository.
- Pre-processing stage has three processes, namely:
  1. Data Cleaning: used to fill in missing data values in as many as 16 data on the bare nuclei variable using the equation Paulin & Santhakumaran median method [14]:

$$\text{MEDIAN} = \text{size of } \frac{(N+1)}{2} \quad (14)$$

2. Sorting the data: is used to separate the data according to the class that is benign and malignant.
3. Normalization of data: aims to change the value on the data value range 1-10 into a value range of 0-1 by using the following equation [15]:

$$X_i' = \frac{X_i - X_{min}}{X_{max} - X_{min}} \quad (15)$$

3.3 Backpropagation Neural Networks Architecture

The data used for the research is binary value after cleaning data, the dataset is 699 data and has 9 variables and 1 target variable for the class. After the dataset is carried out in the pre-processing stage, the data is then used as a dataset during testing to the input layer of the network that will be calculated for each neuron in the hidden layer and output layer. The results (output) of the error crater will be used as test data to see the minimization of the squared error. To test dataset, the following figure 2. Backpropagation neural network architecture:

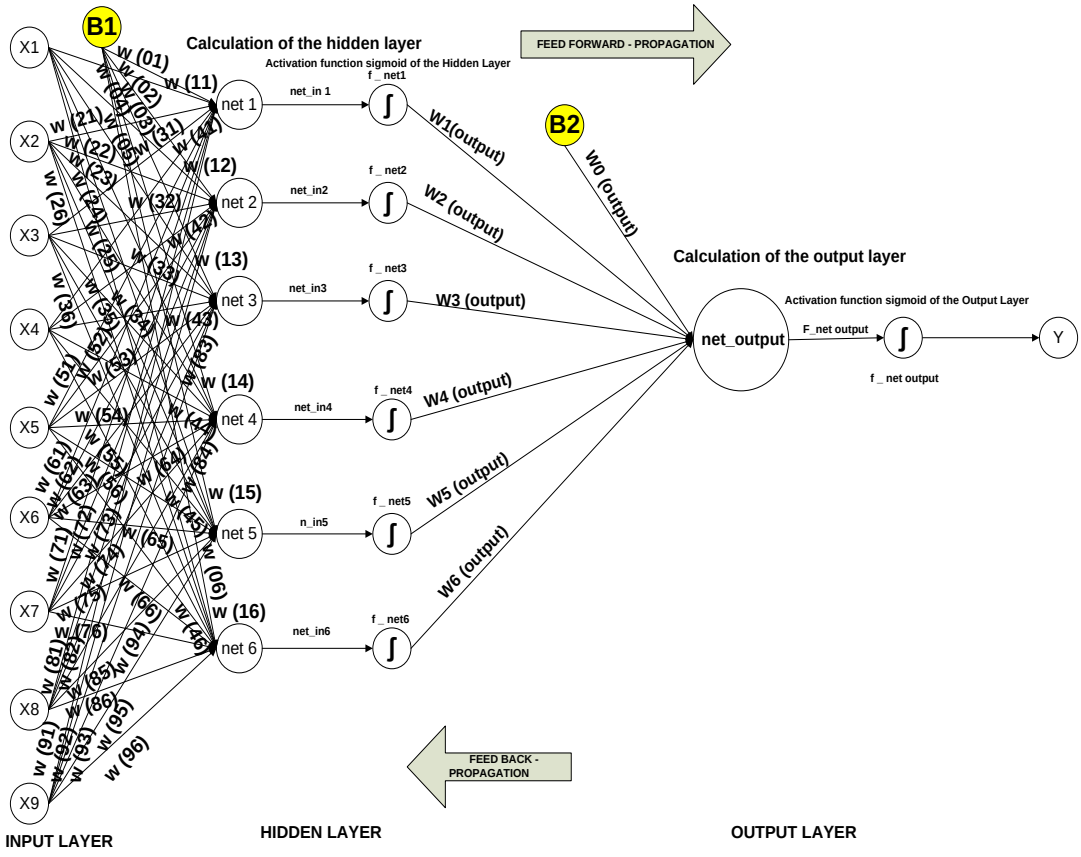


Figure 2. Backpropagation neural network architecture

Figure 2 is a back propagation neural network design. Input on this draft architecture adapted to the feature dataset from the UCI Machine Learning is 9 neurons in the input layer. In the hidden layer based on the calculation that has been calculated and determined, then the neurons in the

hidden layer of 6 neurons. And to output according to the data layer, called the output layer as many as 1 neurons.

3.4 ADAM Architecture Design In Gradient Descent Backpropagation

The image used is 24-bit color image. The calculation of MSE and PSNR aims to determine how much the image changes after message insertion. There is 1-bit storage up to 4-bit LSB to be performed, each stego-image will be calculated MSE and PSNR values to determine which image is better or how many better bits to store information or messages. The following is the formula used to calculate MSE and PSNR. Using (2.2) & (2.3)

Here is a figure 3 is the architecture of ADAM on a gradient descent backpropagation which aim to test dataset:

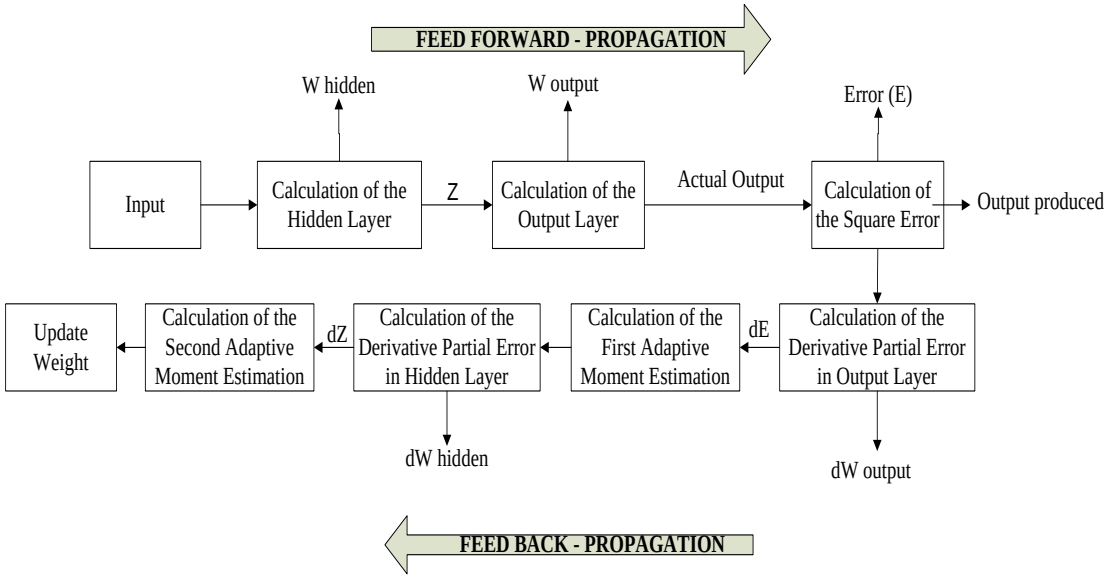


Figure 3. ADAM architecture on gradient descent backpropagation

Figure 3 is an architecture design of ADAM on a gradient descent backpropagation. In the feed forward propagation process, the input data that will be calculated on a hidden layer and output layer computations on. In the calculation results will be summed output layer to the actual value output (target value) in the data, after the reduction process will be conducted gradient descent to minimize the squared error propagation. In the propagation process takes partial derivative calculation process of error in the weight and bias in the output layer. ADAM The first will be done after getting the calculation of partial derivatives of the error in the output layer of weights and biases. After getting the results of the partial derivative calculation on the weights and biases, calculation to estimate the first moment and the moment that the first results will be corrected to gain weight and bias in the output layer. ADAM second will be carried out after obtaining the partial derivative calculation of error in hidden layer of weights and biases. After getting the results of the partial derivative calculation on the weights and biases, carried out calculations to

estimate the second moment and the second moment results will be corrected to get the weights and biases at the hidden layer, so that updates the weights and update the bias obtained.

4. Result and Discussion

The main objective of this research is to focus on the minimization of the squared error in the feed forward propagation. Initial weights randomly in feed forward propagation and update weighs in at feedbackward propagation process. The process of the feed forward propagation on network performance and update of the buffer weight is affected by the provision of learning rate parameter value on network performance can be seen in Table 2.

Table 2. Parameter Value for adaptive estimation moment in gradient descent backpropagation

Parameter	information
Number Of Hidden Neurons	6
Activation Function	Binary sigmoid
maximum Epoch	5
minimum Error	0:01
Learning Rate	0001
Initialization bias and weight to network	Random
Architecture	Multilayer Network: Input: 9 Neuron Hidden Layer: 6 neurons Output: 1 neuron (second class)
Optimization In Backpropagation	Adaptive Moment Estimation ε : 0.00000001 (10 ^ - 8**)** The value of the Exponential for estimation of the first moment (β1): 0.9 The value of the Exponential for the estimation of the second moment (β2): 0999

The data consists of 699 data, the number of variables in the input layer 9 and 1 variable to the target (have 2 classes). Later in the input data before, performed a pre proccesing to get the value range of 0 - 1. The result of the implementation of this program is to minimization of the squared error and the performance of the gradient descent back propagation neural network using ADAM, so I know how the system back propagation neural network to recognize a given pattern. Tests carried out with 5 testing, first on epoch 1, until the fifth test of the epoch 5. In the first test in epoch 1 consists of 699 iteration process, so that by the fifth test at the epoch 5 has the overall iterative process as much as 3494 iterations.

4.1 The First Test On EPOCH 1

The results of ADAM testing on gradient descent backpropagation for the first test on EPOCH 1 can be seen in Figure 4:

Error

Iteration	Square Error
1	0.2183443914320067
2	0.21833426534162517
3	0.21831754609634085
4	0.2183017266126326
5	0.21828144360287252
6	0.2182599009705588
7	0.2182357025427679
8	0.21820982212860704
9	0.21818371993013594
10	0.2181577779019354
11	0.21813291509924626
12	0.21811060973154958
13	0.21809117680189408
14	0.21807547275643813
15	0.2180643898090109
16	0.21805765830132776
17	0.2180555265341538
18	0.21805873586664898
19	0.2180677381706457
20	0.21808217436254637
21	0.21810225028688568
22	0.21812819263374175
23	0.2181601938936229
24	0.21819844286081216
25	0.2182430866346172
26	0.21829370300003864
27	0.21835212601839613
28	0.21842510510790553
29	0.2185027968265684
30	0.21858529723082262
31	0.21867377745486807
32	0.2187671866558931
33	0.2188652985332788
34	0.21896901304775349
35	0.21907774850203118
36	0.21919254002789665
37	0.21931383146701555
38	0.21944011970893126
39	0.21957101394059167
40	0.21970784871846252
41	0.21984932122790785
42	0.2199948440373561
43	0.22014496333359027
44	0.2202994480642018
45	0.22045886250065064
46	0.22062245334364802
47	0.2207899676254977
48	0.2209609412702453

Error

Iteration	Square Error
433	0.2908356936767408
434	0.29093001507672833
435	0.29102417519378726
436	0.2911178986891049
437	0.29121140685837754
438	0.29130435590894627
439	0.29139760803185555
440	0.2914901013472916
441	0.2915827121983036
442	0.2916747253761926
443	0.2917665056561608
444	0.2918575297620595
445	0.29194797568149294
446	0.2920385034076308
447	0.29212832353105417
448	0.2922174821036761
449	0.2923060206637339
450	0.2923939766883878
451	0.29248197971452383
452	0.2925697817401076
453	0.2926553469769873
454	0.2927407199430027
455	0.29282595571699144
456	0.2929108181280084
457	0.2929955911348958
458	0.29308008851001044
459	0.29302972724127624
460	0.29287307122771
461	0.29260394578205656
462	0.29224969977922344
463	0.2918021923610752
464	0.2912710372532344
465	0.2906650459245583
466	0.2900015104017409
467	0.28928014850209544
468	0.2884982692951264
469	0.28766749546111114
470	0.28678620767291246
471	0.2858617550401845
472	0.28489724102946595
473	0.2838837626535493
474	0.28284377623786794
475	0.2817727269309319
476	0.28066707968013793
477	0.27952568245051485
478	0.27835632098039303
479	0.2771685623143681
480	0.27595662999501563

Error

Iteration	Square Error
577	0.20463806082345726
578	0.20508514004861933
579	0.2055672537873786
580	0.20607395618505692
581	0.20660439991475316
582	0.20716852461310428
583	0.20776477114052064
584	0.2083913961368368
585	0.209040315240835
586	0.20971254074474288
587	0.2104010408243566
588	0.21109213992087086
589	0.21179043792140098
590	0.21251137829608513
591	0.21324588522448748
592	0.21399165489344618
593	0.21475975859499913
594	0.21554055911020037
595	0.21634059918128565
596	0.21714488690632475
597	0.21795506858084307
598	0.21878538809164508
599	0.2196288154288003
600	0.22048192381407175
601	0.22134013141240177
602	0.22222143910369702
603	0.22310341359329272
604	0.22401351539783182
605	0.22494284463513914
606	0.22589886645784346
607	0.22685786625490298
608	0.2278289338634118
609	0.228815779272003
610	0.22981743465892182
611	0.23083136825597572
612	0.23185514665745374
613	0.23289220933309812
614	0.23394211284368446
615	0.23499168962047676
616	0.2360518840046439
617	0.23712689476283833
618	0.2381986276550333
619	0.2392974681874346
620	0.240394899330636
621	0.24148503721972372
622	0.2425860255192229
623	0.24367992340253444
624	0.24478754422937726

Error

Iteration	Square Error
652	0.27508500707917555
653	0.2761884674389882
654	0.27729919710794615
655	0.27840070708570175
656	0.27949303495148825
657	0.280579662035238
658	0.28166328907049043
659	0.28275814466062965
660	0.2838584801403564
661	0.28496070607119595
662	0.2860644311671636
663	0.28715144137608245
664	0.288236434263162
665	0.2893079704538922
666	0.29039114058067295
667	0.29146182102240886
668	0.29253135229786253
669	0.29359400651471657
670	0.29464857090967395
671	0.2957011727308291
672	0.2967481262936963
673	0.2977773926614608
674	0.2988038148982141
675	0.2998362416901186
676	0.3008776234368967
677	0.3019330824060276
678	0.3029999755018039
679	0.3040594008962237
680	0.30513668011910405
681	0.30620355323868664
682	0.3072578842628227
683	0.3083202185525004
684	0.3093693794449982
685	0.31041127753027803
686	0.3114529925837359
687	0.3124854079081913
688	0.3135005431036652
689	0.3145077647513129
690	0.31552079263530025
691	0.3165332848105071 3
692	0.31753033793888913
693	0.31852670007530987
694	0.31951229076333454
695	0.32049352018635585
696	0.32147163087473796
697	0.3224377045464258
698	0.323407589786683
699	0.324379559070086

Figure 4. The squared error value in the output layer at epoch 1 starts from iteration 1 to iteration 699

In the results, it can be seen in each iteration of the first test on epoch 1 from iteration 1 to iteration 466 getting an increase from the result of the squared error which is 0.2183443914320067 up to 0.2900015104017409. In the iteration of 467 until iteration 586 gets a decrease from the square of the error which is valued at 0.28928014850209544 it decreases to 0.20971254074474288. In the 587 iteration up to 699 iterations occurred an increase from the results of the error squared value which was 0.2104010408243566 up to 0.324379559070086. So that the first test on epoch 1 in each iteration starting from iteration 1 to iteration 699 does not get a decrease / minimization

of the squared error. Here is a Fig. 5 which is a graph to see the results of the first test on Epoch 1:

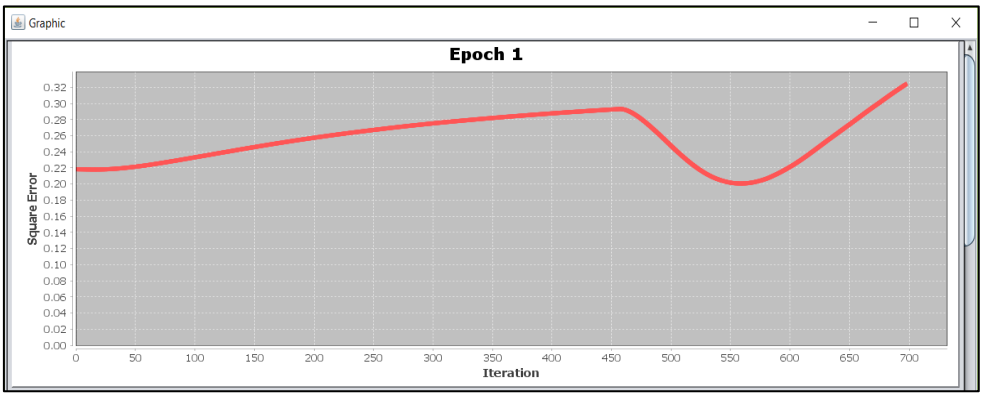


Figure 5. Graph of the first test at epoch 1

4.2 The Second Test On EPOCH 2

The results of ADAM testing on gradient descent backpropagation for the second test on EPOCH 2 can be seen in Figure 6:

Error		Error		Error		Error		Error	
Iteration	Square Error	Iteration	Square Error	Iteration	Square Error	Iteration	Square Error	Iteration	Square Error
700	0.32498026050398193	844	0.18706570848805767	1132	0.2524126391024443	1276	0.17518868212264227	1352	0.23804270453411513
701	0.32524674657946856	845	0.1871800654720399	1133	0.2525768768556188	1277	0.17562407798602347	1353	0.2390138487552314
702	0.3252139843457372	846	0.18730102723456563	1134	0.2527413644687922	1278	0.1760943366532307	1354	0.23997176489618258
703	0.32490119059597367	847	0.18742516593874772	1135	0.2529049126536898	1279	0.17658317101976032	1355	0.2409162611657418
704	0.32434851040136914	848	0.1875541826599815	1136	0.2530682468914368	1280	0.17709027618204443	1356	0.2418520279000807
705	0.32358418267627215	849	0.1876876366712546	1137	0.2532302710922167	1281	0.17763122145299018	1357	0.24278246400533793
706	0.32262804262670386	850	0.1878255242640544	1138	0.25339361397633053	1282	0.17820598093920742	1358	0.2437277161797117
707	0.3214977702503516	851	0.1879670570032362	1139	0.2535550273394963	1283	0.17880708946777346	1359	0.24468469887046687
708	0.32020773676618364	852	0.18811123092488593	1140	0.2537168946071046	1284	0.1794268266766979	1360	0.24564427196918778
709	0.3187848367567527	853	0.18825847346939573	1141	0.2538772687664784	1285	0.18006584703194672	1361	0.24660637804623498
710	0.3172400280231929	854	0.18840941998697738	1142	0.25403726954742045	1286	0.18071500892703712	1362	0.2475481299881 6
711	0.3155913742538877	855	0.18856385354663	1143	0.2541953580010502 5	1287	0.18135853238515393	1363	0.2484892853702188
712	0.3138443246918631	856	0.18872219203210436	1144	0.2543520529292499	1288	0.18199923876382249	1364	0.24941405981721523
713	0.3120128682499858	857	0.18888291982092187	1145	0.2545093938004973	1289	0.1826619729811274	1365	0.25035593896999825
714	0.31010480814182384	858	0.18904632514796774	1146	0.2546649606872704	1290	0.18333135559786587	1366	0.25128406742898207
715	0.3081336556621836	859	0.18921225537000388	1147	0.2548188900112744	1291	0.18400824153864564	1367	0.25221045107751816
716	0.3061136596914087	860	0.18939180967764455	1148	0.2549713047905482	1292	0.18470472912815142	1368	0.2531277963235839
717	0.30404642477610805	861	0.18957902266806365	1149	0.2551223159714037	1293	0.18540937144848324	1369	0.2540399373548878
718	0.3019373824493913	862	0.18977283438208956	1150	0.2552738241018639	1294	0.18612996089114073	1370	0.2549500263535 24
719	0.2997987980370959	863	0.1899676752888724	1151	0.25542504594704685	1295	0.18685056118606055	1371	0.2558531967275634
720	0.29763764120200725	864	0.1901636290061076	1152	0.25557214137749784	1296	0.18756724292779323	1372	0.2567356816898142
721	0.2954574593128584 4	865	0.19036076923160136	1153	0.2557186813743273 5	1297	0.18830250628565878	1373	0.2576198712305287
722	0.29326600046641965	866	0.190559160223756	1154	0.2558649310211098	1298	0.18904436951451162	1374	0.2585175242520587
723	0.29106877788143254	867	0.19076099840730867	1155	0.2560105782058837	1299	0.1897923625925934	1375	0.2594271108001845
724	0.2888656759401566 6	868	0.19096479206645472	1156	0.2561563012156435 3	1300	0.19054421340534541	1376	0.2603572998127841
725	0.2866620299228302	869	0.1911694773107427	1157	0.2563013255982416 5	1301	0.1913195289859942	1377	0.2613030480918635
726	0.2844617745627975	870	0.1913751329372047	1158	0.2564835902701433 6	1302	0.1920903447866018	1378	0.2622451058276911
727	0.2822701995506466 4	871	0.19158156187835168	1159	0.256119094060028 6	1303	0.1928872879535562	1379	0.2632113789084879
728	0.280089984761488	872	0.19180596790744436	1160	0.2558118934714513 3	1304	0.19370390740156126	1380	0.264164744519929
729	0.27792273736135564	873	0.19203162527139628	1161	0.2553894384341086 6	1305	0.19455044140867137	1381	0.2651020449192270 6
730	0.2757691769109124 6	874	0.19226814369584025	1162	0.2548482411673053	1306	0.19539671592716845	1382	0.2660492549206936
731	0.2736331318248363	875	0.1925050069459928	1163	0.2541988790060580 3	1307	0.19625291279282533	1383	0.26698372739972315
732	0.2715160578051427	876	0.1927423225369567	1164	0.2534598055819089	1308	0.19712545066934434	1384	0.2679024159454884
733	0.2694161276579598	877	0.19297904144105904	1165	0.2526440031321531	1309	0.19801251025102656	1385	0.26882350498941326
734	0.267338847117244	878	0.1932188262650295	1166	0.2517531387626407 7	1310	0.1989092366081599	1386	0.26973858528478467
735	0.2652794046823062	879	0.19345790689297773	1167	0.2507929196234525 7	1311	0.19981391423129383	1387	0.270635188014052
736	0.2632494892801007	880	0.19369846925643766	1168	0.2497847822986138 4	1312	0.2007309770551402	1388	0.27151990880113297
737	0.2612474050218997 5	881	0.1939430012476118	1169	0.248718462178868 3	1313	0.20165992952651293	1389	0.2724109730306017
738	0.2592722459450666	882	0.19418661097087617	1170	0.2476009674348291 5	1314	0.20258292762874014	1390	0.27330800620897167
739	0.25731673515028797	883	0.19442974470809873	1171	0.246436006329221 5	1315	0.2035235802193389	1391	0.2741887642710969
740	0.2553889007622018	884	0.19467196815752433	1172	0.2452240804417524 2	1316	0.20448164499708021	1392	0.27506979235199436
741	0.2534884196863988	885	0.19491401885090673	1173	0.2439834076819822 6	1317	0.20543010193385813	1393	0.2759339779230506
742	0.2516221126918673	886	0.19515631776958756	1174	0.2427226259889947	1318	0.20641769648774916	1394	0.2767921277930648
743	0.2497823525665616 4	887	0.19539834244454193	1175	0.241429387826498	1319	0.20739402978724716	1395	0.2776510711915995
744	0.24796780420927342	888	0.19564019896420395	1176	0.2401038206503716 7	1320	0.2083586182254224	1396	0.27849912461870785
745	0.24618808750008692	889	0.19588789239130722	1177	0.238747977080325 6	1321	0.2093345350352541	1397	0.27935634923909736
746	0.2444368609686952	890	0.19613593725609674	1178	0.2373735625354326 3	1322	0.21029279363212866	1398	0.28021954886291195
747	0.24271618624914312	891	0.1963865395831961	1179	0.2359815159795263 2	1323	0.21126244739993924	1399	0.2807332818651873

Figure 6. The squared error value in the output layer at epoch 2 starts from 700 iterations up to iterations 1399

In the results, it can be seen in the second test on epoch 2 in each iteration of 700 iterations up to iteration 1399 indicating a decrease / minimization of the squared error value of 700 iterations to 863 iterations which are 0.32498026050398193 decreased to 0.1899676752888724. In the iteration process 864 up to iteration 1167 get an increase from the results of the error squared value which is worth 0.1901636290061076 up to 0.25079291962345257. In the iteration process 1168 up to 1284 iterations get a decrease / minimization of the squared error which is 0.24978478229861384 decreases to 0.1794268266766979. In the iteration process 1285 up to iteration 1399 get an increase from the result of the squared error which is 0.18006584703194672 rising to 0.2807332818651873. So that the second test on epoch 2 in each iteration for the entire iteration of 700 to iteration 1399 gets a decrease / minimization of the squared error. Following figure 7 is a graph to see the results of the second test on Epoch 2:

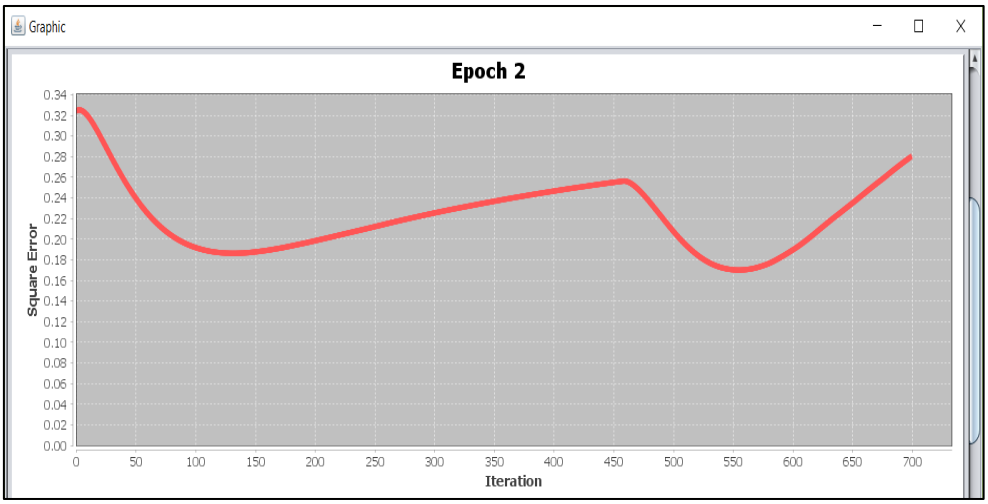
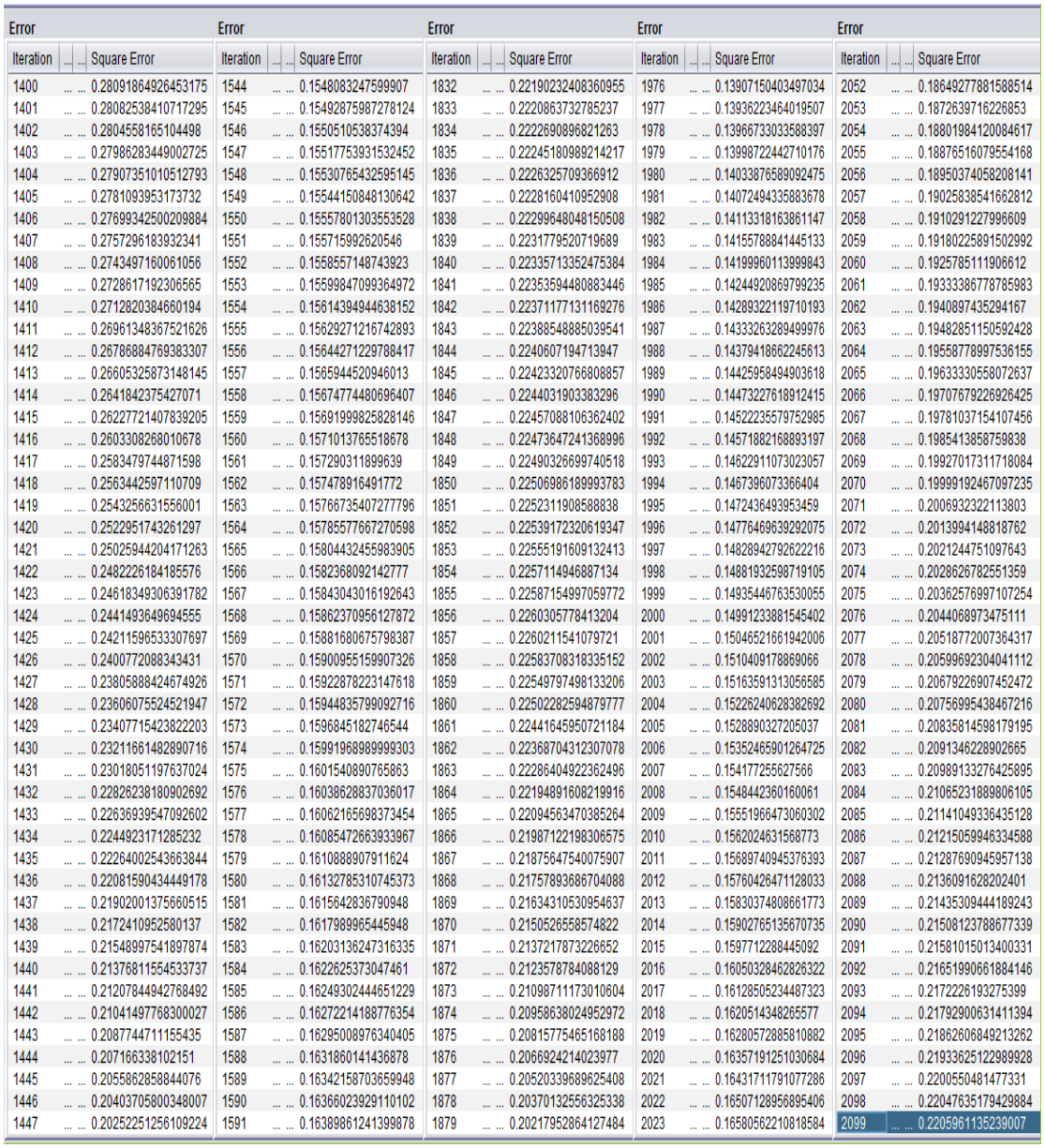


Figure 7. Graph of the second test at Epoch 2

4.3 The Third Test On EPOCH 3

The results of ADAM testing on gradient descent backpropagation for the third test on epoch 3 can be seen in Figure 8:



Error			Error			Error			Error			Error		
Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error
1400	... ...	0.28091864926453175	1544	... ...	0.1548083247599907	1832	... ...	0.221902324083609 55	1976	... ...	0.13907150403497034	2052	... ...	0.18649277881588514
1401	... ...	0.28082538410717295	1545	... ...	0.15492875987278124	1833	... ...	0.2220863732785237	1977	... ...	0.13936223464019507	2053	... ...	0.1872639716226853
1402	... ...	0.2804558165104498	1546	... ...	0.1550510538374394	1834	... ...	0.2222690896821263	1978	... ...	0.13966733033588397	2054	... ...	0.18801984120084617
1403	... ...	0.27986283449002725	1547	... ...	0.15517753931532452	1835	... ...	0.222451809892142 17	1979	... ...	0.13998722442710176	2055	... ...	0.18876516079554168
1404	... ...	0.27907351010512793	1548	... ...	0.15530765432595145	1836	... ...	0.2226325709366912	1980	... ...	0.14033876589092475	2056	... ...	0.18950374058208141
1405	... ...	0.2781093953173732	1549	... ...	0.15544150848130642	1837	... ...	0.2228160410952908	1981	... ...	0.14072494335883678	2057	... ...	0.19025838541662812
1406	... ...	0.27699342500209884	1550	... ...	0.15557801303553528	1838	... ...	0.2229964804815050 8	1982	... ...	0.14113318163861147	2058	... ...	0.1910291227996609
1407	... ...	0.2757296183932341	1551	... ...	0.155715992620546	1839	... ...	0.2231779520719689	1983	... ...	0.14155788841445133	2059	... ...	0.19180225891502992
1408	... ...	0.2743497160061056	1552	... ...	0.1558557148743923	1840	... ...	0.223357133524753 84	1984	... ...	0.14199960113999843	2060	... ...	0.1925785111906612
1409	... ...	0.2728617192306565	1553	... ...	0.15599847099364972	1841	... ...	0.223535944808834 46	1985	... ...	0.14244920869799235	2061	... ...	0.19333386778785983
1410	... ...	0.2712820384660194	1554	... ...	0.15614394944638152	1842	... ...	0.223711771311692 76	1986	... ...	0.14289322119710193	2062	... ...	0.1940897435294167
1411	... ...	0.26961348367521626	1555	... ...	0.15629271216742893	1843	... ...	0.223885488850395 41	1987	... ...	0.14333263289499976	2063	... ...	0.19482851150592428
1412	... ...	0.26786884769383307	1556	... ...	0.15644271229788417	1844	... ...	0.224060719471394 7	1988	... ...	0.14379418662245613	2064	... ...	0.19558778997536155
1413	... ...	0.26605325873148145	1557	... ...	0.1565944520946013	1845	... ...	0.224233207668088 57	1989	... ...	0.14425958494903618	2065	... ...	0.19633330558072637
1414	... ...	0.2641842375427071	1558	... ...	0.15674774480696407	1846	... ...	0.224403190338329 6	1990	... ...	0.14473227618912415	2066	... ...	0.19707679226926425
1415	... ...	0.26227721407839205	1559	... ...	0.15691999825828146	1847	... ...	0.224570881063624 02	1991	... ...	0.14522235579752985	2067	... ...	0.19781037154107456
1416	... ...	0.2603308268010678	1560	... ...	0.1571013765518678	1848	... ...	0.224736472413689 96	1992	... ...	0.14571882168893197	2068	... ...	0.1985413858759838
1417	... ...	0.2583479744871598	1561	... ...	0.157290311899639	1849	... ...	0.224903269974051 8	1993	... ...	0.14622911073023057	2069	... ...	0.19927017311718084
1418	... ...	0.2563442597110709	1562	... ...	0.157478916491772	1850	... ...	0.225069861899937 83	1994	... ...	0.1467396073366404	2070	... ...	0.19999192467097235
1419	... ...	0.2543256631556001	1563	... ...	0.15766735407277796	1851	... ...	0.225231190858883 8	1995	... ...	0.1472436493953459	2071	... ...	0.2006932322113803
1420	... ...	0.2522951743261297	1564	... ...	0.15785577667270598	1852	... ...	0.225391723206193 47	1996	... ...	0.14776469639292075	2072	... ...	0.2013994148818762
1421	... ...	0.25025944204171263	1565	... ...	0.15804432455983905	1853	... ...	0.225551916091324 13	1997	... ...	0.14828942792622216	2073	... ...	0.2021244751097643
1422	... ...	0.2482226184185576	1566	... ...	0.1582368092142777	1854	... ...	0.2257114946887134	1998	... ...	0.14881932598719105	2074	... ...	0.20286267882551359
1423	... ...	0.24618349306391782	1567	... ...	0.15843043016192643	1855	... ...	0.225871549970597 72	1999	... ...	0.14935446763530055	2075	... ...	0.20362576997107254
1424	... ...	0.2441493649694555	1568	... ...	0.15862370956127872	1856	... ...	0.226030577841320 4	2000	... ...	0.14991233881545402	2076	... ...	0.20440689734751 11
1425	... ...	0.24211596533307697	1569	... ...	0.15881680675798387	1857	... ...	0.226021154107972 1	2001	... ...	0.15046521661942006	2077	... ...	0.20518772007364317
1426	... ...	0.2400772088343431	1570	... ...	0.15900955159907326	1858	... ...	0.225837083183351 52	2002	... ...	0.1510409178869066	2078	... ...	0.20599692304041112
1427	... ...	0.23805888424674926	1571	... ...	0.15922878223147618	1859	... ...	0.225497974981332 06	2003	... ...	0.15163591313056585	2079	... ...	0.2067922690745247 2
1428	... ...	0.23606075524521947	1572	... ...	0.15944835799092716	1860	... ...	0.225022825948797 77	2004	... ...	0.15226240628382692	2080	... ...	0.20756995438467216
1429	... ...	0.23407715423822203	1573	... ...	0.1596845182746544	1861	... ...	0.224416459507211 84	2005	... ...	0.1528890327205037	2081	... ...	0.20835814598179195
1430	... ...	0.23211661482890716	1574	... ...	0.15991968989999303	1862	... ...	0.223687043123070 78	2006	... ...	0.15352465901264725	2082	... ...	0.2091346228902665
1431	... ...	0.23018051197637024	1575	... ...	0.1601540890765863	1863	... ...	0.222864049223624 96	2007	... ...	0.154177255627566	2083	... ...	0.20989133276425895
1432	... ...	0.22826238180902692	1576	... ...	0.16038628837036017	1864	... ...	0.221948916082199 16	2008	... ...	0.1548442360160061	2084	... ...	0.21065231889806105
1433	... ...	0.22636939547092602	1577	... ...	0.16062165698373454	1865	... ...	0.220945634703852 64	2009	... ...	0.15551966473060302	2085	... ...	0.21141049336435128
1434	... ...	0.2244923171285232	1578	... ...	0.16085472663933967	1866	... ...	0.219871221983065 75	2010	... ...	0.1562024631568773	2086	... ...	0.21215059946334588
1435	... ...	0.22264002543663844	1579	... ...	0.1610888907911624	1867	... ...	0.218756475400759 07	2011	... ...	0.15689740945376393	2087	... ...	0.21287690945957138
1436	... ...	0.22081590434449178	1580	... ...	0.16132785310745373	1868	... ...	0.217578936867040 88	2012	... ...	0.15760426471128033	2088	... ...	0.2136091628202401
1437	... ...	0.21902001375660515	1581	... ...	0.1615642836790948	1869	... ...	0.216343105309546 37	2013	... ...	0.15830374808661773	2089	... ...	0.21435309444189243
1438	... ...	0.2172410952580137	1582	... ...	0.1617989965445948	1870	... ...	0.215052655857482 2	2014	... ...	0.15902765135670735	2090	... ...	0.21508123788677339
1439	... ...	0.21548997541897874	1583	... ...	0.16203136247316335	1871	... ...	0.213721787322665 2	2015	... ...	0.1597712288445092	2091	... ...	0.21581015013400331
1440	... ...	0.21376811554533737	1584	... ...	0.1622625373047461	1872	... ...	0.212357878408812 9	2016	... ...	0.16050328462826322	2092	... ...	0.21651990661884146
1441	... ...	0.21207844942768492	1585	... ...	0.16249302444651229	1873	... ...	0.210987111730106 04	2017	... ...	0.16128505234487323	2093	... ...	0.2172226193275399
1442	... ...	0.21041497768300027	1586	... ...	0.16272214188776354	1874	... ...	0.209586380249529 72	2018	... ...	0.1620514348265577	2094	... ...	0.21792900631411394
1443	... ...	0.2087744711155435	1587	... ...	0.16295008976340405	1875	... ...	0.208157546516818 8	2019	... ...	0.16280572885810882	2095	... ...	0.21862606849213262
1444	... ...	0.207166338102151	1588	... ...	0.1631860141436878	1876	... ...	0.206692421402397 7	2020	... ...	0.16357191251030684	2096	... ...	0.21933625122989928
1445	... ...	0.2055862858844076	1589	... ...	0.16342158703659948	1877	... ...	0.205203396896254 08	2021	... ...	0.16437111791077286	2097	... ...	0.2200550481477331
1446	... ...	0.20403705800348007	1590	... ...	0.16366023929110102	1878	... ...	0.203701325563253 38	2022	... ...	0.16507128956895406	2098	... ...	0.22047635179429884
1447	... ...	0.20252251256109224	1591	... ...	0.16389861241399878	1879	... ...	0.202179528641274 84	2023	... ...	0.16580562210818584	2099	... ...	0.2205961135239007

Figure 8. The squared error value in the output layer at epoch 3 starts from iterations 1400 to iterations 2099

In the results, it can be seen in the third test on epoch 3 in each iteration from iteration 1400 to iteration 2099 getting a decrease / minimization of the squared error value of iterations 1400 to iteration 1545 which value 0.28091864926453175 decreases to 0.15492875987278124. In the iteration of 1546 until iteration 1858 obtained an increase from the result of the squared error value of 0.1550510538374394, rising to 0.22583708318335152. In the iteration 1859 to the 1980 iteration, the decrease / minimization of the error error value of 0.22549797498133206 decreased to 0.14033876589092475. In the 1981 iteration up to 2099 iterations get an increase from the results of the error squared value which is 0.14072494335883678 up to 0.2205961135239007. So that the third test on epoch 3 in each iteration for the whole of iterations 1400 to iteration 2099 gets a decrease / minimization of the squared error. Following figure 9 is a graph to see the results of the third test on Epoch 3:

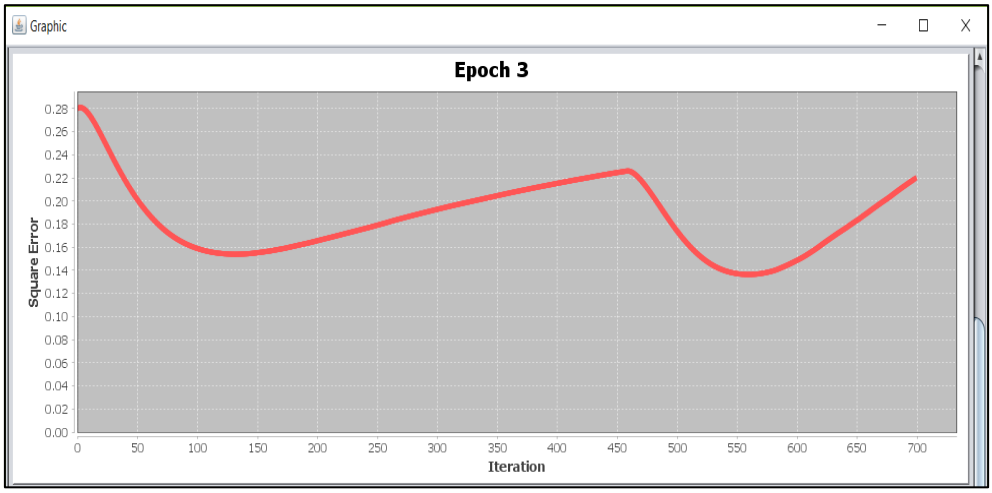


Figure 9. Graph of the third test at Epoch 3

4.4 The Fourth Test On EPOCH 4

The results of ADAM testing on gradient descent backpropagation for the fourth test on epoch 4 can be seen in Figure 10:

Error			Error			Error			Error			Error		
Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error
2100	... ...	0.22048338304646564	2244	... ...	0.12194312699269359	2532	... ...	0.18447054667366639	2676	... ...	0.10552402802431735	2752	... ...	0.13719207193449473
2101	... ...	0.22011577763822543	2245	... ...	0.12206380052970295	2533	... ...	0.18465587784809775	2677	... ...	0.10567761951793427	2753	... ...	0.13773266967940284
2102	... ...	0.21956120752944544	2246	... ...	0.1221871360806004	2534	... ...	0.18484139307743216	2678	... ...	0.10584155677228965	2754	... ...	0.13826239787079897
2103	... ...	0.21884155979666942	2247	... ...	0.12231250231366145	2535	... ...	0.18502419703768658	2679	... ...	0.10603047370913046	2755	... ...	0.13878744431038715
2104	... ...	0.21798222786164875	2248	... ...	0.12244013376122495	2536	... ...	0.18521129390301244	2680	... ...	0.10624969325947714	2756	... ...	0.13932067833649536
2105	... ...	0.21701137878617732	2249	... ...	0.12256874117142759	2537	... ...	0.18539416717356497	2681	... ...	0.1064849599465841	2757	... ...	0.1398730185370973
2106	... ...	0.2159186277626417	2250	... ...	0.12269702516013957	2538	... ...	0.1855787064066628	2682	... ...	0.10673276072931769	2758	... ...	0.14042639289172784
2107	... ...	0.21474094221138768	2251	... ...	0.12282536038810737	2539	... ...	0.18576007913232465	2683	... ...	0.1069935884472588	2759	... ...	0.1409820989263129
2108	... ...	0.213481362438156	2252	... ...	0.12295530365527461	2540	... ...	0.1859410486604255	2684	... ...	0.10726004862888978	2760	... ...	0.14151944789233992
2109	... ...	0.21215188843937896	2253	... ...	0.12308658916570384	2541	... ...	0.18611791176653839	2685	... ...	0.10752281659289625	2761	... ...	0.1420574129494973
2110	... ...	0.21075183465295627	2254	... ...	0.12321986685661317	2542	... ...	0.18629189747127964	2686	... ...	0.10778191086036455	2762	... ...	0.14258006927128167
2111	... ...	0.20929120579647345	2255	... ...	0.12335278590894926	2543	... ...	0.18646835413644827	2687	... ...	0.1080599032183265	2763	... ...	0.14312217791931592
2112	... ...	0.20777058924180627	2256	... ...	0.12348602709635535	2544	... ...	0.18664106700217262	2688	... ...	0.10833977077723786	2764	... ...	0.1436521594003414
2113	... ...	0.20621266816058592	2257	... ...	0.1236193986734071	2545	... ...	0.186810380520376	2689	... ...	0.10862564464418759	2765	... ...	0.1441797117812627
2114	... ...	0.2046333984745041	2258	... ...	0.12377675991938626	2546	... ...	0.18697660533044225	2690	... ...	0.10892465226991091	2766	... ...	0.14469789144040293
2115	... ...	0.2030272866728743	2259	... ...	0.12394384420169816	2547	... ...	0.1871400215752955	2691	... ...	0.1092278922525032	2767	... ...	0.1452151480928016
2116	... ...	0.20139307209124427	2260	... ...	0.12411856332103241	2548	... ...	0.1873053822774048	2692	... ...	0.10954139439417347	2768	... ...	0.14572981912706462
2117	... ...	0.19975024595454913	2261	... ...	0.12429119353124893	2549	... ...	0.18747062962719727	2693	... ...	0.10985521531074001	2769	... ...	0.14623807445323372
2118	... ...	0.19810268126795194	2262	... ...	0.1244619800055821	2550	... ...	0.18762985866129434	2694	... ...	0.11016314136315486	2770	... ...	0.14672922471060323
2119	... ...	0.19645255037949869	2263	... ...	0.12463115578466386	2551	... ...	0.1877880274363016	2695	... ...	0.11048414318428736	2771	... ...	0.14722619774467155
2120	... ...	0.19480437518504443	2264	... ...	0.12479894081626319	2552	... ...	0.1879457721004635	2696	... ...	0.1108066863210416	2772	... ...	0.14774311021979655
2121	... ...	0.19315995012279938	2265	... ...	0.12497067441313923	2553	... ...	0.18810292372399962	2697	... ...	0.11113292520271112	2773	... ...	0.14827146882390194
2122	... ...	0.19151669818753828	2266	... ...	0.12514223101059144	2554	... ...	0.1882610100855613	2698	... ...	0.11146431102729623	2774	... ...	0.1488240710835776
2123	... ...	0.18988428921322406	2267	... ...	0.12531190045281934	2555	... ...	0.18841771397787513	2699	... ...	0.11181423346661544	2775	... ...	0.14939348878240705
2124	... ...	0.18824699741817086	2268	... ...	0.12547992283689074	2556	... ...	0.1884191825643146	2700	... ...	0.11215981785250793	2776	... ...	0.1499647948188793
2125	... ...	0.18658596548881354	2269	... ...	0.12564620425242623	2557	... ...	0.18822533895035368	2701	... ...	0.11252246141521483	2777	... ...	0.15056357899420322
2126	... ...	0.18495451891145626	2270	... ...	0.12584713848723258	2558	... ...	0.18788067263630312	2702	... ...	0.11290083829727786	2778	... ...	0.15114954956556367
2127	... ...	0.18335026047313713	2271	... ...	0.12604696309794833	2559	... ...	0.18738617274336267	2703	... ...	0.11330734069141142	2779	... ...	0.151719180873641
2128	... ...	0.1817590330337565	2272	... ...	0.12626917230070564	2560	... ...	0.18676062550911754	2704	... ...	0.1137141878041584	2780	... ...	0.1522978029046528
2129	... ...	0.18019519709389384	2273	... ...	0.12648855024899697	2561	... ...	0.1860062497734057	2705	... ...	0.11412785887555173	2781	... ...	0.1528667035210243
2130	... ...	0.17866011676653346	2274	... ...	0.1267054348340233	2562	... ...	0.18516610297877079	2706	... ...	0.11455622969226306	2782	... ...	0.15341677124115624
2131	... ...	0.17714167416497742	2275	... ...	0.1269181905567533	2563	... ...	0.18422091067650992	2707	... ...	0.11499647116653132	2783	... ...	0.15397140395179182
2132	... ...	0.1756488374823423	2276	... ...	0.12713351072320822	2564	... ...	0.1831780954602736	2708	... ...	0.11544292383348433	2784	... ...	0.15452551308348259
2133	... ...	0.17416693618162904	2277	... ...	0.12734467233164204	2565	... ...	0.18207213621226762	2709	... ...	0.11589504662121979	2785	... ...	0.15506428498917713
2134	... ...	0.17270072207898865	2278	... ...	0.12755604357528605	2566	... ...	0.1809430190725432	2710	... ...	0.11635716594619168	2786	... ...	0.15559011944486828
2135	... ...	0.17126002239004834	2279	... ...	0.12777238625329213	2567	... ...	0.17975220908210351	2711	... ...	0.1168293792989618	2787	... ...	0.15612021134239928
2136	... ...	0.16984613346874095	2280	... ...	0.12798440291896027	2568	... ...	0.17849864390955444	2712	... ...	0.11729466975765435	2788	... ...	0.15666455935389131
2137	... ...	0.1684437931960222	2281	... ...	0.12819321859364158	2569	... ...	0.17718578306080338	2713	... ...	0.11778627588756763	2789	... ...	0.15719542174872156
2138	... ...	0.16706639763260725	2282	... ...	0.12839811037736393	2570	... ...	0.1758478027998617	2714	... ...	0.1182963090945455	2790	... ...	0.15772679836978523
2139	... ...	0.16571790963011565	2283	... ...	0.1286005417654138	2571	... ...	0.17447405088514303	2715	... ...	0.11879544159456228	2791	... ...	0.15824075244586921
2140	... ...	0.1643960510279362	2284	... ...	0.1288010559675051	2572	... ...	0.17311310713629544	2716	... ...	0.11934720770826227	2792	... ...	0.15874771456129472
2141	... ...	0.16309737266067111	2285	... ...	0.1289989125743197	2573	... ...	0.1717297717819596	2717	... ...	0.11988363434783922	2793	... ...	0.1592592992579107
2142	... ...	0.16181666078019766	2286	... ...	0.12919440382517766	2574	... ...	0.1703287378600535	2718	... ...	0.12040927544532416	2794	... ...	0.15976360071948345
2143	... ...	0.16056216363330264	2287	... ...	0.12939931526824403	2575	... ...	0.1688841367246602	2719	... ...	0.12094508953677402	2795	... ...	0.16028179967544992
2144	... ...	0.15933269056914776	2288	... ...	0.1296028659679498	2576	... ...	0.1674092056565582	2720	... ...	0.12146169934623187	2796	... ...	0.16080890550351484
2145	... ...	0.15813153500249574	2289	... ...	0.12980952065907275	2577	... ...	0.165929685247381	2721	... ...	0.12198374960637696	2797	... ...	0.16111412760810687
2146	... ...	0.15696376447262161	2290	... ...	0.13001510912883066	2578	... ...	0.16443868493498914	2722	... ...	0.12248805750741572	2798	... ...	0.1611675648401819
2147	... ...	0.15582185301231966	2291	... ...	0.13024918117684617	2579	... ...	0.16295176537788036	2723	... ...	0.12297911699074819	2799	... ...	0.16105322504274178

Figure 10. The squared error value in the output layer at epoch 4 starts from the iteration 2100 to iteration 2799

In the results, it can be seen in the fourth test on epoch 4 in each iteration from iteration 2100 until iteration 2799 gets a decrease / minimization of the square of error at the iteration of 2100 to iteration 2244 which is 0.22048338304646564 decreasing to 0.12194312699269359. At iteration 2245 until iteration 2566 gets an increase from the result of the squared error value of 0.12206380052970295 up to 0.1809430190725432. In the iteration 2567 up to iteration 2693, the decrease / minimization of error squares which is 0.17975220908210351 decreases to 0.10985521531074001. At iteration 2694 until iteration 2799 gets an increase from the result of the squared error value which is 0.11016314136315486 up to 0.16105322504274178. So that the fourth test on Epoch 4 in each iteration for the whole of the iteration 2100 until iteration 2799 gets a decrease / minimization of the squared error. Following figure 11 is a graph to see the results of the fourth test on Epoch 4:

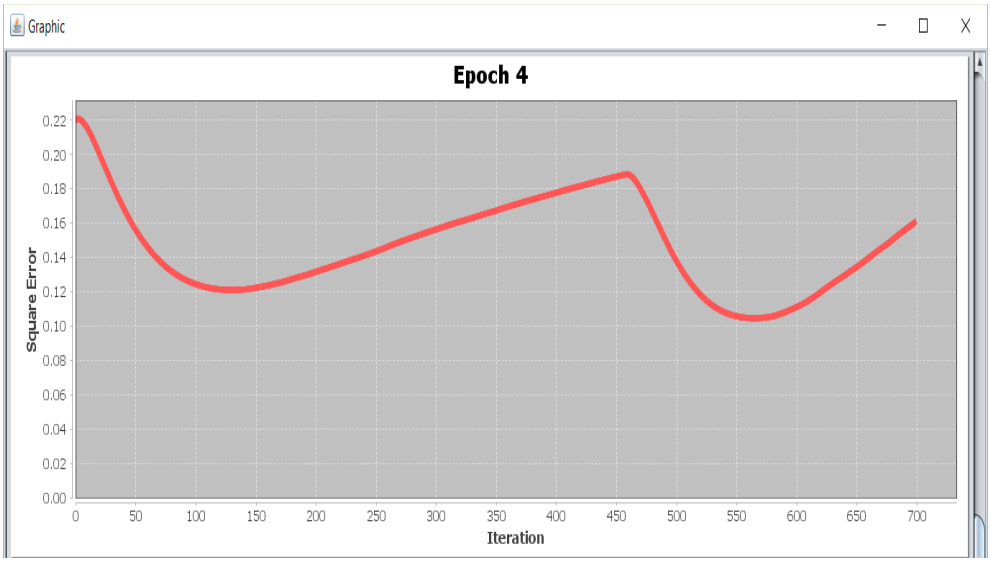
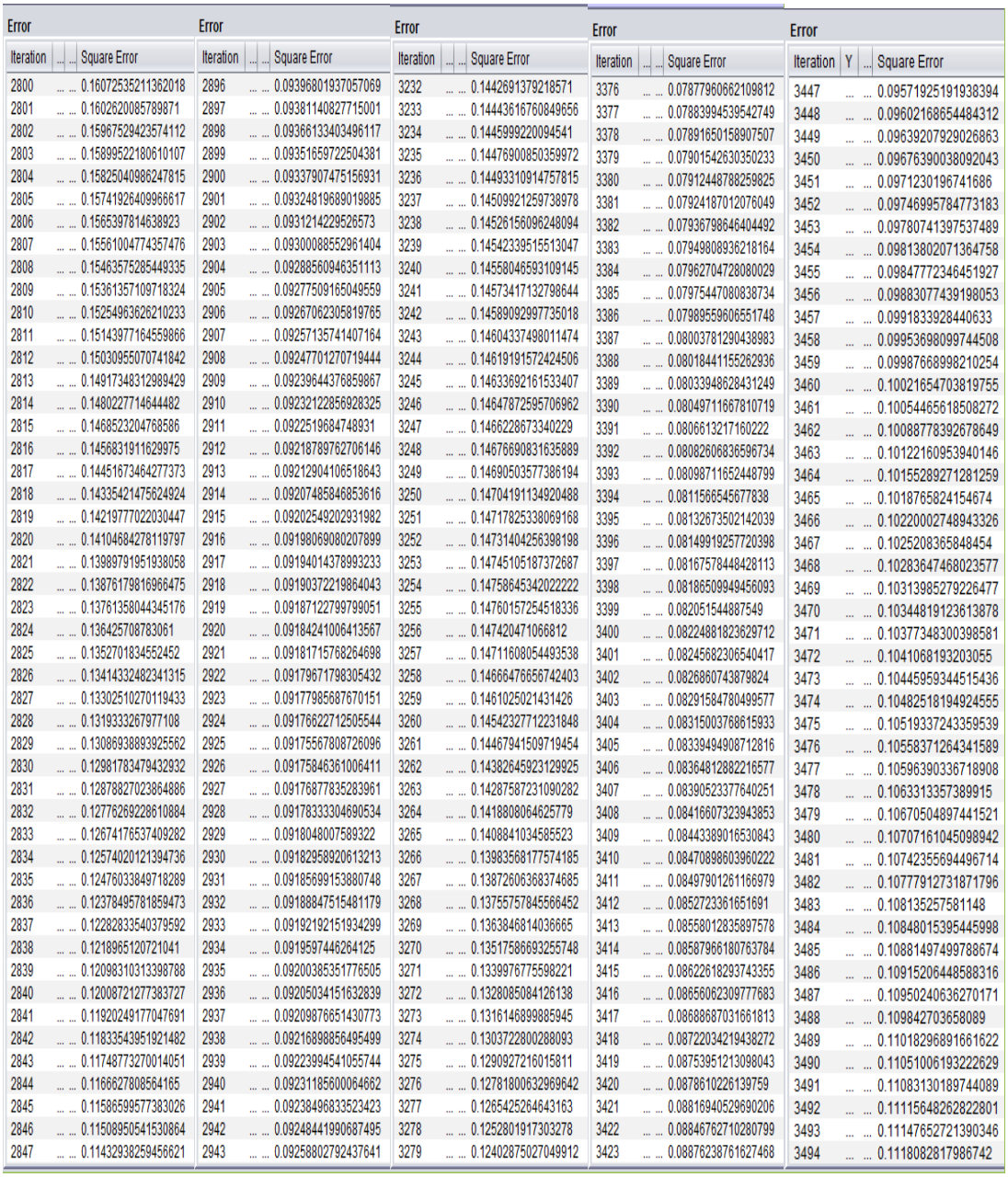


Figure 11. Graph of the fourth test at Epoch 4

4.5 The Fifth Test On EPOCH 4

The results of ADAM testing on gradient descent backpropagation for the fifth test in Epoch 5 can be seen in Figure 12:



Error			Error			Error			Error			Error		
Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	... ...	Square Error	Iteration	Y ...	Square Error
2800	... ...	0.16072535211362018	2896	... ...	0.09396801937057069	3232	... ...	0.1442691379218571	3376	... ...	0.07877960662109812	3447	... ...	0.0957192519193839
2801	... ...	0.1602620085789871	2897	... ...	0.09381140827715001	3233	... ...	0.14443616760849656	3377	... ...	0.07883994539542749	3448	... ...	0.0960216865448431
2802	... ...	0.15967529423574112	2898	... ...	0.09366133403496117	3234	... ...	0.1445999220094541	3378	... ...	0.07891650158907507	3449	... ...	0.0963920792902686
2803	... ...	0.15899522180610107	2899	... ...	0.09351659722504381	3235	... ...	0.14476900850359972	3379	... ...	0.07901542630350233	3450	... ...	0.0967639003809204
2804	... ...	0.15825040986247815	2900	... ...	0.09337907475156931	3236	... ...	0.14493310914757815	3380	... ...	0.07912448788259825	3451	... ...	0.097123019674168
2805	... ...	0.15741926409966617	2901	... ...	0.09324819689019885	3237	... ...	0.14509921259738978	3381	... ...	0.07924187012076049	3452	... ...	0.0974699578477318
2806	... ...	0.1565397814638923	2902	... ...	0.0931214229526573	3238	... ...	0.14526156096248094	3382	... ...	0.07936798646404492	3453	... ...	0.0978074139753748
2807	... ...	0.15561004774357476	2903	... ...	0.09300088552961404	3239	... ...	0.14542339515513047	3383	... ...	0.07949808936218164	3454	... ...	0.0981380207136475
2808	... ...	0.15463575285449335	2904	... ...	0.09288560946351113	3240	... ...	0.14558046593109145	3384	... ...	0.07962704728080029	3455	... ...	0.0984777234645192
2809	... ...	0.15361357109718324	2905	... ...	0.09277509165049559	3241	... ...	0.1457341713279864	3385	... ...	0.07975447080838734	3456	... ...	0.0988307743919805
2810	... ...	0.15254963626210233	2906	... ...	0.09267062305819765	3242	... ...	0.14589092997735018	3386	... ...	0.07989559606551748	3457	... ...	0.099183392844063
2811	... ...	0.15143977164559866	2907	... ...	0.09257135741407164	3243	... ...	0.14604337498011474	3387	... ...	0.08003781290438983	3458	... ...	0.0995369809974450
2812	... ...	0.15030955070741842	2908	... ...	0.09247701270719444	3244	... ...	0.14619191572424506	3388	... ...	0.08018441155262936	3459	... ...	0.0998766899821025
2813	... ...	0.14917348312989429	2909	... ...	0.09239644376859867	3245	... ...	0.1463368921615334O7	3389	... ...	0.08033948628431249	3460	... ...	0.1002165470381975
2814	... ...	0.1480227714644482	2910	... ...	0.09232122856928325	3246	... ...	0.14647872595706962	3390	... ...	0.08049711667810719	3461	... ...	0.1005446561850827
2815	... ...	0.1468523204768586	2911	... ...	0.0922519684748931	3247	... ...	0.1466228673340229	3391	... ...	0.0806613217160222	3462	... ...	0.1008877839267864
2816	... ...	0.1456831911629975	2912	... ...	0.09218789762706146	3248	... ...	0.14676690831635889	3392	... ...	0.08082606836596734	3463	... ...	0.1012216095394014
2817	... ...	0.14451673464277373	2913	... ...	0.09212904106518643	3249	... ...	0.14690503577386194	3393	... ...	0.08098711652448799	3464	... ...	0.1015528927128125
2818	... ...	0.14335421475624924	2914	... ...	0.09207485846853616	3250	... ...	0.14704191134920488	3394	... ...	0.0811566545677838	3465	... ...	0.101876582415467
2819	... ...	0.14219777022030447	2915	... ...	0.09202549202931982	3251	... ...	0.14717825338069168	3395	... ...	0.08132673502142039	3466	... ...	0.1022000274894332
2820	... ...	0.14104684278119797	2916	... ...	0.09198069080207899	3252	... ...	0.14731404256398198	3396	... ...	0.08149919257720398	3467	... ...	0.102520836584845
2821	... ...	0.13989791951938058	2917	... ...	0.09194014378993233	3253	... ...	0.14745105187372687	3397	... ...	0.08167578448428113	3468	... ...	0.1028364746802357
2822	... ...	0.13876179816966475	2918	... ...	0.09190372219864043	3254	... ...	0.14758645342022222	3398	... ...	0.08186509949456093	3469	... ...	0.1031398527922647
2823	... ...	0.13761358044345176	2919	... ...	0.09187122799799051	3255	... ...	0.14760157254518336	3399	... ...	0.082051544887549	3470	... ...	0.1034481912361387
2824	... ...	0.136425708783061	2920	... ...	0.09184241006413567	3256	... ...	0.147420471066812	3400	... ...	0.08224881823629712	3471	... ...	0.1037734830039858
2825	... ...	0.13527018345524S2	2921	... ...	0.09181715768264698	3257	... ...	0.14711608054493538	3401	... ...	0.08245682306540417	3472	... ...	0.104106819320305
2826	... ...	0.13414332482341315	2922	... ...	0.09179671798305432	3258	... ...	0.14666476656742403	3402	... ...	0.082686074879824	3473	... ...	0.1044595934451543
2827	... ...	0.13302510270119433	2923	... ...	0.09177985687670151	3259	... ...	0.1461025021431426	3403	... ...	0.08291584780499577	3474	... ...	0.1048251819492455
2828	... ...	0.13193332679771O8	2924	... ...	0.09176622712505544	3260	... ...	0.14542327712231848	3404	... ...	0.08315003768615933	3475	... ...	0.1051933724335953
2829	... ...	0.13086938893925562	2925	... ...	0.09175567808726096	3261	... ...	0.14467941509719454	3405	... ...	0.08339494908712816	3476	... ...	0.1055837126434158
2830	... ...	0.12981783479432932	2926	... ...	0.09175846361006411	3262	... ...	0.14382645923129925	3406	... ...	0.08364812882216577	3477	... ...	0.1059639033671890
2831	... ...	0.12878827023864886	2927	... ...	0.09176877835283961	3263	... ...	0.14287587231090282	3407	... ...	0.08390523377640251	3478	... ...	0.106331335738991
2832	... ...	0.12776269228610884	2928	... ...	0.09178333304690534	3264	... ...	0.1418808064625779	3408	... ...	0.08416607323943853	3479	... ...	0.1067050489744152
2833	... ...	0.12674176537409282	2929	... ...	0.0918048007589322	3265	... ...	0.140841034585523	3409	... ...	0.08443389016530843	3480	... ...	0.1070716104509894
2834	... ...	0.12574020121394736	2930	... ...	0.09182958920613213	3266	... ...	0.13983568177574185	3410	... ...	0.08470898603960222	3481	... ...	0.1074235569449671
2835	... ...	0.12476033849718289	2931	... ...	0.09185699153880748	3267	... ...	0.13872606368374685	3411	... ...	0.08497901261166979	3482	... ...	0.1077791273187179
2836	... ...	0.12378495781859473	2932	... ...	0.09188847515481179	3268	... ...	0.13755757845566452	3412	... ...	0.0852723361651691	3483	... ...	0.108135257581148
2837	... ...	0.12282833540379592	2933	... ...	0.09192192151934299	3269	... ...	0.1363846814036665	3413	... ...	0.08558012835897578	3484	... ...	0.1084801539544599
2838	... ...	0.1218965120721041	2934	... ...	0.0919597446264125	3270	... ...	0.13517586693255748	3414	... ...	0.08587966180763784	3485	... ...	0.1088149749978867
2839	... ...	0.12098310313398788	2935	... ...	0.09200385351776505	3271	... ...	0.1339976775598221	3415	... ...	0.0862261829374355	3486	... ...	0.1091520644858831
2840	... ...	0.12008721277383727	2936	... ...	0.09205034151632839	3272	... ...	0.1328085084126138	3416	... ...	0.08656062309777683	3487	... ...	0.1095024063627017
2841	... ...	0.11920249177047691	2937	... ...	0.09209876651430773	3273	... ...	0.1316146899885945	3417	... ...	0.08688687031661813	3488	... ...	0.109842703658089
2842	... ...	0.11833543951921482	2938	... ...	0.09216898856495499	3274	... ...	0.1303722800288093	3418	... ...	0.08722034219438272	3489	... ...	0.1101829689166162
2843	... ...	0.11748773270014051	2939	... ...	0.09223994541055744	3275	... ...	0.1290927216015811	3419	... ...	0.08753951213098043	3490	... ...	0.1105100619322262
2844	... ...	0.1166627808564165	2940	... ...	0.09231185600064662	3276	... ...	0.12781800632969642	3420	... ...	0.0878610226139759	3491	... ...	0.1108313018974408
2845	... ...	0.11586599577383026	2941	... ...	0.09238496833523423	3277	... ...	0.1265425264643163	3421	... ...	0.08816940529690206	3492	... ...	0.1111564826228028
2846	... ...	0.11508950541530864	2942	... ...	0.09248441990687495	3278	... ...	0.12528019173003278	3422	... ...	0.08846762710280799	3493	... ...	0.1114765272139034
2847	... ...	0.11432938259456621	2943	... ...	0.09258802792437641	3279	... ...	0.12402875027049912	3423	... ...	0.08876238761627468	3494	... ...	0.111808281798674

Figure 12. The squared error value in the output layer at epoch 5 starts from the iteration 2800 to iteration 3494

In the results, it can be seen in the fifth test on epoch 5 in each iteration from iteration 2800 until iteration 3494 gets a decrease / minimization of the squared error value at iterations 2800 to iteration 2934 which is 0.16072535211362018 decreases to 0.0919597446264125. In the iteration 2935 until iteration 3257 get an increase from the results of the squared error value which is 0.09200385351776505 up to 0.14711608054493538. In the iteration 3258 up to iteration 3378, the decline / minimization of the value of the results of the squared error of 0.14666476656742403 decreased to 0.07891650158907507. In the 3379 iteration up to iteration 3494 get an increase from the results of the squared error value which is 0.07901542630350233 up to 0.1118082817986742. So that the fifth test on Epoch 5 in each iteration for the overall iteration

of 2800 until iteration 3494 gets a decrease / minimization of the squared error. Following figure 13 is a graph to see the results of the fifth test on Epoch 5:

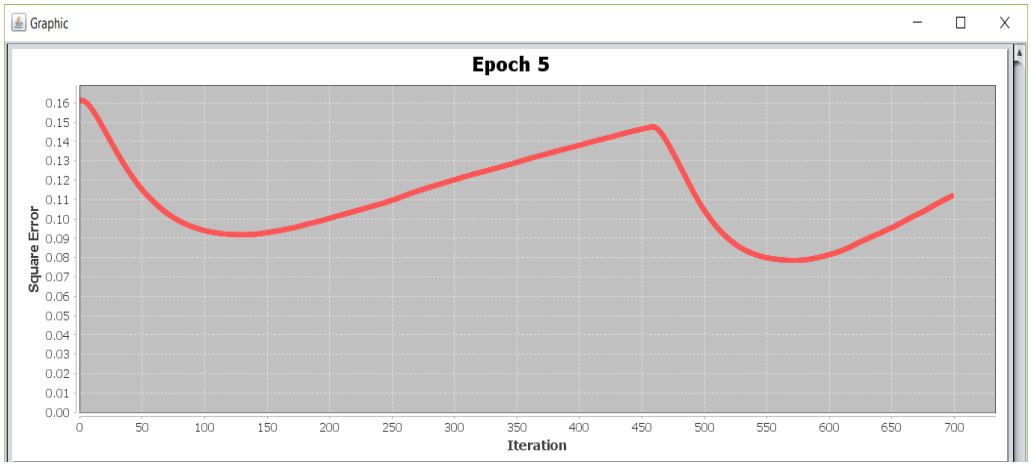


Figure 13. Graph of the fifth test at Epoch 5

From the graph of the first test on Epoch 1 to the fifth test on Epoch 5, can be seen in Figure Graph 14 which is the whole test above. The first test on EPO 1 to Fifth Test on EPO can be seen in Figure 14:

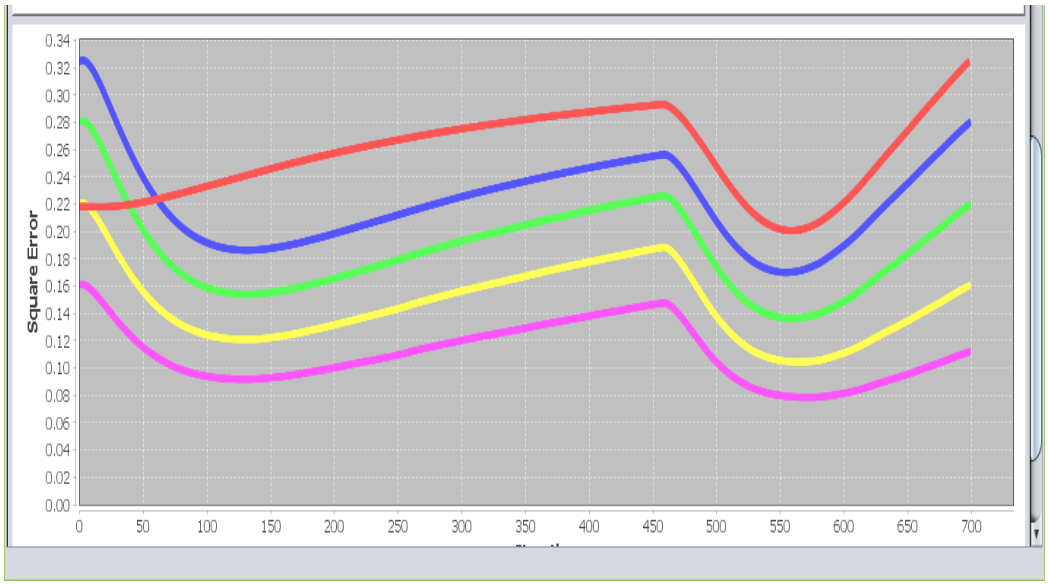


Figure 14. The first test chart until the fifth test in epoch 1 through epoch 5

5. Conclusion

The first test on Epoch 1 is an increase in the value of the squared error. In the second test on Epoch 2 to the fifth test at Epoch 5 there is minimization / decrease of squared error in each epoch test. The results of tests that have been conducted on neural network networks namely ADAM on gradient descent backpropagation can help the learning performance of neural network networks on gradient descent backpropagation to minimize / decrease squared errors. Furthermore, a new method analysis can be carried out for the learning level, so that it is expected that from the first

test on Epoch 1 to the fifth test at Epoch 5 it results in a decrease in the minimization of the squared error.

REFERENCES

[1] Guan, N., Shan, L., Yang, C., Xu W., & Zhang, M., 2017. Delay Compensated Asynchronous Adam Algorithm for Deep Neural Networks. *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications*, pp. 852 – 859.

[2] Wakitani, S., Yamamoto, T., & Ishimura, A., 2017. Study on an adaptive GMDH-PID controller using adaptive moment estimation. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1587 – 1591.

[3] Wu, M., Guo, S., Chen, X., Xing, N., & Zhong, C., 2016. LM–BP based operation quality assessment method for OTN in Smart grid. *Proceedings of the IEEE Network Operations and Management Symposium.*

[4] Ahmad, F., Isa, N., A., M., Osman M., K., & Hussain, Z., 2010. Performance comparison of gradient descent and Genetic Algorithm based Artificial Neural Networks training. *Proceedings of the IEEE International Conference on Intelligent Systems Design and Applications*, pp. 604 – 609.

[5] Popa, C-A., 2014. Enhanced Gradient Descent Algorithms for Complex-Valued Neural Networks. *Proceedings of the IEEE International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 272 – 279.

[6] Singh, B., K., Verma, K., & Thoke, A., S., 2015. Adaptive Gradient Descent Backpropagation for Classification of Breast Tumor in Ultrasound Imaging. *Proceedings of the Elsevier International Conference on Information and Communication Technologies* **46**(9): 1601 – 1609.

[7] Heravi, A., R., & Hodtani, G., A., 2018. A New Correntropy-Based Conjugate Gradient Backpropagation Algorithm for Improving Training in Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* **29**(12) : 6252 – 6263.

[8] Li, Y., Zhu, L., Zhou, L-j., & Jiang, J., 2011. Study on the BP–GA model and its application in water quality assessment. *Proceedings of the IEEE International Symposium on Water Resource and Environmental Protection*, pp. 2781 – 2784.

[9] Achkar, R., Geagea, R., Mehio., & Kmeish, W., 2016. SmartCoach personal gym trainer: An Adaptive Modified Backpropagation approach. *Proceedings of the IEEE International Multidisciplinary Conference on Engineering Technology*, pp. 1 – 6.

[10] Mammadli, S., 2017. Financial time series prediction using artificial neural network based on Levenberg–Marquardt algorithm. *Proceedings of the Elsevier International Conference on Theory and Application of Soft Computing, Computing with Words and Perception* **120**(6): 602 – 607.

[11] Andayani, U., Nababan, E., B., Siregar, B., Muchtar, M., A., Nasution, T., H., & Siregar, I., 2017. Optimization backpropagation algorithm based on Nguyen–Widrow adaptive weight and adaptive learning rate. *Proceeding of the IEEE International Conference on Industrial Engineering and Application*, pp. 363 – 367.

[12] Indolia, S., Goswami, A., K., Mishra, S., P., & Asopa, P., 2018. Conceptual Understanding of Convolutional Neural Network – A Deep Learning Approach. *Proceedings of the*

Elsevier International Conference on Computational Intelligence and Data Science **132**(10): 679 - 688.

[13] Srinivasan, N., Ravichandran, V., Chan, K., L., Vidhya, J., R., Ramakirishnan, S., & Krishnan, S., M., 2002. Exponentiated backpropagation algorithm for multilayer feedforward neural networks. *Proceedings of the IEEE Proceedings of the 9th International Conference on Neural Information Processing, 2002. ICONIP '02*, pp. 327 – 331.

[14] Paulin, F., & Santhakumaran, A., 2010. Back Propagation Neural Network by Comparing Hidden Neuron: Case study on Breast Cancer Diagnosis. *International Journal of Computer Application* **2**(4): 40 – 44.

[15] Chen, C., C., Kuo, C., Kuo, S., Y., & Chou, Y., H., 2015. Dynamic Normalization BPN for Stock Price Forecasting. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2855 - 2860.