Adaptive Moment Estimation To Minimize Square Error In Backpropagation Algorithm

R N Singarimbun\textsuperscript{1}, E B Nababan\textsuperscript{2*}, and Opim Salim Sitompul\textsuperscript{3}

\textsuperscript{1}Graduate School of Computer Science
\textsuperscript{2,3}Department of Information Technology, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

\section*{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.

\section*{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

Copyright © 2020 Published by Talenta Publisher  
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 Backpropagation 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% Mamadli [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 $= \frac{\partial E}{\partial W(k)}$.

- 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:
  \[
  m_{weight(t)} = \beta_1 * m_{weight(t-1)} + (1 - \beta_1) * g_t \\
  m_{bias(t)} = \beta_1 * m_{bias(t-1)} + (1 - \beta_1) * g_t
  \] (1)

- The calculation of weight and bias correction estimation of the first moment $\hat{m}_t$ by the following equation:
  \[
  \hat{m}_{weight(t)} = \frac{m_{weight(t)}}{1 - \beta_1^t} \\
  \hat{m}_{bias(t)} = \frac{m_{bias(t)}}{1 - \beta_1^t}
  \] (2)

- **Gradient calculation** for estimating the first moment in time step $= \frac{\partial E}{\partial W(k)}$.

- 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:
  \[
  v_{weight(t)} = \beta_2 v_{weight(t-1)} + (1 - \beta_2) * (g_t)^2 \\
  v_{bias(t)} = \beta_2 v_{bias(t-1)} + (1 - \beta_2) * (g_t)^2
  \] (3)
The calculation of weight and bias correction the estimated second moment $\tilde{V}_t$ by the following equation:

$$
\tilde{V}_{\text{weight}}(t) = \frac{v_{\text{weight}}(t)}{1 - \beta_1^t}
$$

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

Parameter updates weight and bias by the following equation:

$$
w_{\text{weight}}(t) = w_{\text{weight}}(t-1) - \alpha * \frac{\dot{m}_{\text{weight}}(t)}{\sqrt{\tilde{V}_{\text{weight}}(t)} + \epsilon}
$$

$$
w_{\text{bias}}(t) = w_{\text{bias}}(t-1) - \alpha * \frac{\dot{m}_{\text{bias}}(t)}{\sqrt{\tilde{V}_{\text{bias}}(t)} + \epsilon}
$$

### 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:

$$
\text{net}_{ij}^h = \sum_{i=1}^n X_i W_{ij}^h + W_{\text{bias}_j}^h
$$

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

$$
f_{\text{in}} = \frac{1}{1 + e^{-\text{net}_{ij}^h}}
$$

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

$$
\text{net}_{ik}^o = \sum_{i=1}^n (f_{\text{in},n} \cdot W_{ik}^o) + W_{\text{bias}_k}^o
$$

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

$$
f_{\text{out},n} = \frac{1}{1 + e^{-\text{net}_{ik}^o}}
$$

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

$$
e_{\text{output}} = \text{Target} - f_{\text{out},n}
$$

6. Calculates the square error at the output layer to the equation:
Square Error = \[
\frac{1}{2} \sum_k (\text{Target} - f_{out\_net\_n})^2
\] (11)

7. Calculates the partial derivative of the weight and bias for each neuron in the output layer with the equation:
\[
\frac{\partial E}{\partial W_{ik}} = -(\text{Target} - f_{out\_net\_n}) \cdot f'_{out\_net\_n} \cdot f'_{in\_net
}
\] (12)
\[
\frac{\partial E}{\partial W_{bias}} = -(\text{Target} - f_{out\_net\_n}) \cdot f'_{out\_net\_n}
\]

8. Calculates the partial derivative of the weight and the bias for each neuron in the hidden layer with the equation:
\[
\frac{\partial E}{\partial W_{weight\_ij}} = - \sum_k (T - f_{out\_net\_n}) \cdot f'_{out\_net\_n} \cdot w_{ik}^0 \cdot f'_{in\_net\_n} \cdot X_i
\] (13)
\[
\frac{\partial E}{\partial W_{bias\_i}} = - \sum_k (T - f_{out\_net\_n}) \cdot f'_{out\_net\_n} \cdot w_{ik}^0 \cdot f'_{in\_net\_n}
\]

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Range interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clump Thickness</td>
<td>1-10</td>
</tr>
<tr>
<td>Uniformity of Cell Size</td>
<td>1-10</td>
</tr>
<tr>
<td>Uniformity of Cell Shape</td>
<td>1-10</td>
</tr>
<tr>
<td>marginal Adhesion</td>
<td>1-10</td>
</tr>
<tr>
<td>Single Epithelial Cell Size</td>
<td>1-10</td>
</tr>
<tr>
<td>Bare Nuclei</td>
<td>1-10</td>
</tr>
<tr>
<td>Bland Chromatin</td>
<td>1-10</td>
</tr>
<tr>
<td>normal Nucleoli</td>
<td>1-10</td>
</tr>
<tr>
<td>Mitoses</td>
<td>1-10</td>
</tr>
<tr>
<td>Class variable</td>
<td>Benign cells (2) and malignant cells (4)</td>
</tr>
</tbody>
</table>

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.

![Block diagram](image)

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]:

\[
\text{MEDIAN} = \text{size of } \frac{(N+1)}{2} \tag{14}
\]
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]:

$$X'_i = \frac{X_i - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$  \hspace{1cm} (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](image)

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>
<thead>
<tr>
<th>Parameter</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Of Hidden Neurons</td>
<td>6</td>
</tr>
<tr>
<td>Activation Function</td>
<td>Binary sigmoid</td>
</tr>
<tr>
<td>maximum Epoch</td>
<td>5</td>
</tr>
<tr>
<td>minimum Error</td>
<td>0:01</td>
</tr>
<tr>
<td>Learning Rate</td>
<td>0001</td>
</tr>
<tr>
<td>Initialization bias and weight to network</td>
<td>Random</td>
</tr>
<tr>
<td>Architecture</td>
<td>Multilayer Network:</td>
</tr>
<tr>
<td></td>
<td>Input: 9 Neuron</td>
</tr>
<tr>
<td></td>
<td>Hidden Layer: 6 neurons</td>
</tr>
<tr>
<td></td>
<td>Output: 1 neuron (second class)</td>
</tr>
<tr>
<td>Optimization In Backpropagation</td>
<td>Adaptive Moment Estimation</td>
</tr>
<tr>
<td></td>
<td>$\varepsilon : 0.00000001 (10^{-8})$</td>
</tr>
</tbody>
</table>

The value of the Exponential for estimation of the first moment ($\beta_1$): 0.9
The value of the Exponential for the estimation of the second moment ($\beta_2$): 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 processing 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:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Square Error</th>
<th>Iteration</th>
<th>Square Error</th>
<th>Iteration</th>
<th>Square Error</th>
<th>Iteration</th>
<th>Square Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.20134031420257</td>
<td>111</td>
<td>0.20045162592768</td>
<td>211</td>
<td>0.2043808098234572</td>
<td>311</td>
<td>0.2075909014710052</td>
</tr>
<tr>
<td>2</td>
<td>0.20134031420257</td>
<td>112</td>
<td>0.20045162592768</td>
<td>212</td>
<td>0.2043808098234572</td>
<td>312</td>
<td>0.2075909014710052</td>
</tr>
<tr>
<td>3</td>
<td>0.20134031420257</td>
<td>113</td>
<td>0.20045162592768</td>
<td>213</td>
<td>0.2043808098234572</td>
<td>313</td>
<td>0.2075909014710052</td>
</tr>
<tr>
<td>4</td>
<td>0.20134031420257</td>
<td>114</td>
<td>0.20045162592768</td>
<td>214</td>
<td>0.2043808098234572</td>
<td>314</td>
<td>0.2075909014710052</td>
</tr>
<tr>
<td>5</td>
<td>0.20134031420257</td>
<td>115</td>
<td>0.20045162592768</td>
<td>215</td>
<td>0.2043808098234572</td>
<td>315</td>
<td>0.2075909014710052</td>
</tr>
</tbody>
</table>

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 0.218344391432067 up to 0.290001510417409. In the iteration of 467 until iteration 586 gets a decrease from the square of the error which is valued at 0.2892814850209544 it decreases to 0.2097125407447288. In the 587 iteration up to 699 iterations occurred an increase from the result of the error squared value which was 0.2104010482343566 up to 0.32437955970086. 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](image)

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:

![Figure 6. The squared error value in the output layer at epoch 2 starts from 700 iterations up to iterations 1399](image)
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.

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](image)

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:
Figure 8. The squared error value in the output layer at epoch 3 starts from iterations 1400 to iteration 2099. In the results, it can be seen in the third test on epoch 3, in 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 iteration 1980, the decrease / minimization of the error error value of 0.22549797498133206 decreased to 0.140336589092475. In the iteration 1981 to 2099 iterations get an increase from the results of the error squared value which is 0.14072494335883678 up to 0.220596133259007. 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:

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 iteration 1980, the decrease / minimization of the error error value of 0.22549797498133206 decreased to 0.140336589092475. In the iteration 1981 to 2099 iterations get an increase from the results of the error squared value which is 0.14072494335883678 up to 0.220596133259007. 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:
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:

![Figure 10. The squared error value in the output layer at epoch 4 starts from the iteration 2100 to iteration 2799](image-url)
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:

![Graph of the fourth test at Epoch 4](image)

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:
In the results, it can be seen in the fifth test on epoch 5 in each iteration from iteration 2800 to iteration 3494 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 0.14711608349727018. In the iteration 3258 up to iteration 3378, the decline / minimization of the value of the results of the squared error of 0.1466647665674243 decreased to 0.07891650158907507. 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:

![Graph of Epoch 5](image)

**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:

![Graph of Epoch 1 to Epoch 5](image)

**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


