A Deep Neural Network (Dnn) For Prediction Of Percentage Of Gross Domestic Product Distribution At Current Price By Industry Sector

Emmilya Umma Aziza Gaffar, Achmad fanani Onnelia Gaffar

Abstract: Economic growth as measured by GDP growth rates and economic growth set as an increase in GDP strongly helps government predictions about the economic situation and the formation of economic development strategies. This measurement is done by combining mathematical and computer technology to make qualitative and quantitative predictions scientifically and appropriately for economic growth trends. It is a good practical sense to use scientific and proven methods to predict future GDP development trends of a particular economy. In some cases, machine learning methods have proven to be better forecasting results than statistical methods. A Deep Neural Network (DNN) is one type of ANN (Artificial Neural network) architecture based on deep MLP (Multi Layer Perceptron), which uses Deep Learning training techniques. This study proposes the use of DNN to predict the percentage of GDP distribution at current prices by industry sector. In this case, the DNN used will have multiple outputs as many industry sectors. The aim of this study is how to predict for the next period with the smallest possible prediction errors by using DNN.

Index Terms: Prediction, GDP, ANN, MLP, DNN.

1. INTROUCTION

The basic measure of the value added arising from economic activity is known as Gross Domestic Product at the national level and Gross Regional Domestic Product (GRDP) at the regional level (provinces/regencies/municipalities). To compile these statistics, two approaches have been used, i.e. production approach and expenditure approach. The first approach is to measure value added produced by various kinds of economic activities, while the second approach is to measure final uses of the country's output. In other words, GDP/GRDP is the sum of total value added produced by all economic industries (activities) and the way of using it. GDP and its aggregations are presented in two forms: at current market prices and at constant base year market prices. In presenting current market prices, all aggregates are valued at current market prices, while base year constant market prices are shown by valuing all aggregates at fixed base year prices. GDP by industrial sector is detailed according to the total added value of all existing economic sectors [1-10]. In this study, GDP by industrial sector covers: (1). agriculture, livestock, forestry, & fishery, (2). mining & quarrying, (3). manufacturing industry, (4). electricity, gas, & water supply, (5). construction, (6). trade, hotel, & restaurant, (7). transportation & communication, (8). finance, real estate, & business services, (9). other services. There are three ways to determine a country's GDP, all of which must be done, although in principle it gives the same result. The easiest way is to approach the product, where the total number of outputs of each company class. The expenditure approach works based on the principle that all products must be purchased by a person. In this case the total value of the product must equal the total expenditure.

- *Emmilya Umma A G is currently lecturer in Department of Economic Study, Mulawarman University, East Kalimantan, Indonesia PH- E-mail: emmilya.gaffars@gmail.com*
- *Achmad Fanani O. G is currently lecturer in Department of Information Technology, State Polytechnic of Samarinda, East Kalimantan, Indonesia. E-mail: onnygaffar212@gmail.com*

The income approach works on the principle that the earning income factor must be equal to the value of the product, where GDP is determined by finding the total revenue of all producers [11]. A measured economic growth with GDP growth rate and defined economic growth as an increase of GDP is helpful to government's reasonable prediction of the economic situation and establishing of the economic development strategy. This measurement is performed by combining mathematics and computer technology to make scientific and precise qualitative and quantitative prediction result for the trend of economic growth. It is of great practical meaning to use scientific and proven methods to predict GDP development trend of a certain economy in future[12]. In some forecasting cases, statistical methods have yielded good results. Many statistical forecasting methods are based on the assumption that time series can be given stationary through the use of transformations. Once the prediction is made on a stationary time series, it can be converted back to the original series using the same transformation that makes it stationary. However, the researchers tried to improve the forecasting results by applying machine learning methods (Fuzzy Logic, Genetic Algorithm, Neural Network, etc.) [11, 13-15]. In some cases machine learning methods have proven to be better forecasting results than statistical methods. Hybrid methods, both between statistics and machine learning or between machine learning methods, are also commonly used to improve performance of forecasting results or to compare performance among several methods [16-18]. Artificial Neural Network (ANN) has been successfully applied in various activities of identification and control of dynamic systems. The universal approximation capability of Multi Layer Perceptron (MLP) makes it a popular choice for modeling various nonlinear systems and for implementing a general purpose nonlinear system. One of the most common goals is prediction[19]. Various forecasting activities have been carried out by applying MLP, such as the application of time series model ARIMA seasonal models [20], flood forecasting [21], MLP and RBF (Radial Base Function) Network combination for rainfall prediction [22], etc. This study proposes the use of MLP networks to predict the percentage of GDP distribution at current prices by industry sector using a Deep Learning approach. This is commonly known as Deep Neural Network

(DNN), a deep MLP network that uses deep learning training techniques. The aim of this study is to obtain the predicted percentage of GDP distribution at current prices by industry sector for the next period with the smallest possible prediction errors. In this case, the MLP network which used will have multiple outputs as many industry sectors.

2 METHODS

2.1 DNN

A Deep Neural Network (DNN) is a deep MLP (with multiple layers), which uses Deep Learning training techniques. Data input is transformed from a low-level feature to a high-level feature. The input layer is characterized by having multiple inputs. In each hidden layer, data is encoded into features with fewer or more dimensions with non-linear transformations. The existence of the hidden layer is to map the non-linear nature of the training data. The more hidden layers will be the increased networking capability in terms of mapping non-linear properties of data. Each subsequent hidden layer will refine the learning pattern in high-level features, and so on until it is able to learn the complex patterns of training data used. For networks with multiple outputs, usually hidden layers will have dimensions at least equal to the dimensions of data input. A DNN with two hidden layer and multi outputs is shown in Fig. 1. In general, there are two direction processes that are forward and backward as follows.

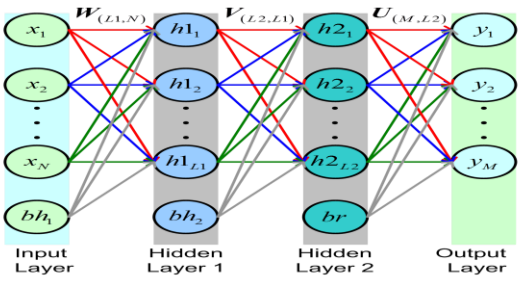

***Figure 1**. A DNN with two hidden layer and multi outputs*

Like the MLP network, back-propagation techniques are also used to train DNN. In principle, back-propagation training techniques consist of two stages: forward and backward as shown in Fig. 2.

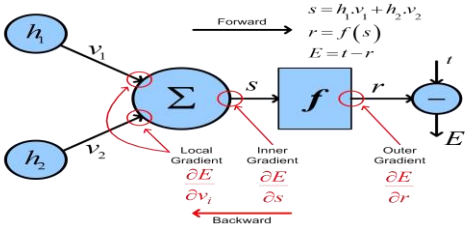

***Figure 2**. Back-propagation training technique*

Forward direction stage

The forward direction is the stage of completing the amount of data weighting of each layer down to the output layer according to the planned network architecture. Refer to Fig. 1 the process is mathematically as follows.

The nodes of 1st hidden layer are the weighted sums of the input nodes and its biases which generally expressed by

$$h1_i = \sum_{j=1}^{N} x_j . w_{ij} + b_{h1(i)} \; ; \; i = 1...L1$$

$$\begin{bmatrix} h1_1 \\ h1_2 \\ \dots \\ h1_{L1} \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1,N} \\ w_{21} & w_{22} & \dots & w_{2,N} \\ \dots & \dots & \dots & \dots \\ w_{L1,1} & w_{L1,2} & \dots & w_{L1,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} + \begin{bmatrix} b_{h1(1)} \\ b_{h1(2)} \\ \dots \\ b_{h1(L1)} \end{bmatrix} \quad (1)$$

$$\boldsymbol{h1}_{(L1,1)} = \boldsymbol{W}_{(L1,N)} \square \boldsymbol{x}_{(N,1)} + \boldsymbol{b}_{h1(L1,1)}$$

The final part is the canonical form where N is the number of input node, L1 is the number of 1st hidden layer node, $x_{(N,1)}$ is the column vector of input, $W_{(L1,N)}$ is the weight matrix of between input layer and 1st hidden layer, $b_{h1(L1,1)}$ is the column vector of the 1st hidden layer's biases, and $h1_{(L1,1)}$ is the column vector of the weighted sums of the input nodes. The 1st hidden layer output is the result of a transformation using a certain activation function which expressed by

$$\begin{bmatrix} H1_1 \\ H1_2 \\ \dots \\ H1_{L1} \end{bmatrix} = \begin{bmatrix} f(h1_1) \\ f(h1_2) \\ \dots \\ f(h1_{L1}) \end{bmatrix} \qquad \boldsymbol{H1}_{(L1,1)} = \boldsymbol{f}\left(\boldsymbol{h1}_{(L1,1)}\right) \quad (2)$$

where $H1_{(L1,1)}$ is the column vector of the 1st hidden layer output.

In the same way, for next layers are expressed by

$$\boldsymbol{h2}_{(L2,1)} = \boldsymbol{V}_{(L2,L1)} \square \boldsymbol{H1}_{(L1,1)} + \boldsymbol{b}_{h2(L2,1)} \quad (3)$$

$$\boldsymbol{H2}_{(L2,1)} = \boldsymbol{f}\left(\boldsymbol{h2}_{(L2,1)}\right) \quad (4)$$

$$\boldsymbol{y}_{(M,1)} = \boldsymbol{U}_{(M,L2)} \square \boldsymbol{H2}_{(L2,1)} + \boldsymbol{b}_{r(M,1)} \quad (5)$$

$$\boldsymbol{Y}_{(M,1)} = \boldsymbol{f}\left(\boldsymbol{y}_{(M,1)}\right) \quad (6)$$

The error function is declared by SSE (Sum Squared Error) as follows

$$SSE = E = \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{M}\left(t_j^{(k)} - y_j^{(k)}\right)^2 = \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{M}\left(e_j^{(k)}\right)^2 \quad (7)$$

Where K is the number of training data, M is the number of net outputs

Backward direction stage

The backward direction stage is the back propagation process of the gradient descent of network error to adjust the weighting of each layer. By using the gradient descent algorithm for learning, the gradient of the error function will computed with respect to the weights and biases. Mathematically, the processes are as follows.

The outer gradients of the output layer are expressed by

$$\Delta Y^{(k)} = -\frac{\delta E^{(k)}}{\delta Y^{(k)}} = -\frac{\delta}{\delta Y^{(k)}}\left(\frac{1}{2}\sum_{j=1}^{N}\left(t_j^{(k)} - y_j^{(k)}\right)^2\right) = t^{(k)} - y^{(k)} = e^{(k)} \quad (8)$$

In canonical form can be expressed by

$$\Delta \boldsymbol{Y}_{(N,1)}{}^{(k)} = \frac{\delta \boldsymbol{E}^{(k)}}{\delta \boldsymbol{Y}_{(N,1)}{}^{(k)}} = \boldsymbol{e}_{(N,1)}{}^{(k)} = \boldsymbol{T}_{(N,1)}{}^{(k)} - \boldsymbol{Y}_{(N,1)}{}^{(k)} \quad (9)$$

By using the principle of chain rule (chain rules), the inner gradient of the output layer nodes are expressed by

$$\frac{\delta E}{\delta y^{(k)}} = \frac{\delta E}{\delta Y^{(k)}} . \frac{\delta Y^{(k)}}{\delta y^{(k)}} \Rightarrow \Delta y^{(k)} = e^{(k)} . f'\left(y\right)^{(k)} \quad (10)$$

In canonical form can be expressed by

$$\Delta \boldsymbol{y}_{(N,1)}^{(k)} = \frac{\delta \boldsymbol{E}^{(k)}}{\delta \boldsymbol{y}_{(N,1)}^{(k)}} = \frac{\delta \boldsymbol{E}^{(k)}}{\delta \boldsymbol{Y}_{(N,1)}^{(k)}} \square \frac{\delta \boldsymbol{Y}_{(N,1)}^{(k)}}{\delta \boldsymbol{y}_{(N,1)}^{(k)}} = \boldsymbol{e}_{(N,1)}^{(k)} \square * f'\left(\boldsymbol{y}_{(N,1)}\right)^{(k)} \quad (11)$$

The local gradient of the output layer nodes are expressed by

$$\Delta u_{jk}^{(k)} = \frac{\delta E}{\delta u_{jk}^{(k)}} = \frac{\delta E}{\delta y_j^{(k)}} \cdot \frac{\delta y^{(k)}}{\delta u_{jk}^{(k)}} = \frac{\delta E}{\delta y_j^{(k)}} \cdot \frac{\delta}{\delta u_{jk}^{(k)}} \left(\sum_{k=1}^{M} H2_k . u_{jk} + br_{(j)} \right)$$

$$= \Delta y_j^{(k)} . H2_k \quad (12)$$

$$\Delta br_{(j)}^{(k)} = \frac{\delta E}{\delta br_{(j)}^{(k)}} = \frac{\delta E}{\delta y_j^{(k)}} \cdot \frac{\delta y_j^{(k)}}{\delta br_{(j)}^{(k)}} = \frac{\delta E}{\delta y_j^{(k)}} \cdot \frac{\delta}{\delta br_{(j)}^{(k)}} \left(\sum_{k=1}^{M} H2_k . u_{jk} + br_{(j)} \right)$$

$$= \Delta y_j^{(k)}$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{U}_{(M,L2)}^{(k)} = \Delta \boldsymbol{y}_{(M,1)}^{(k)} \square \left(\boldsymbol{H2}_{(L2,1)} \right)^T$$
$$\Delta \boldsymbol{b}_{\boldsymbol{r}(M,1)}^{(k)} = \Delta \boldsymbol{y}_{(M,1)}^{(k)} \quad (13)$$

The outer gradient of each node in 2nd hidden layer will receive the inner gradient of the output layer's nodes, so that

$$\frac{\delta E}{\delta H2_k^{(k)}} = \frac{\delta}{\delta H2_k^{(k)}} \left(\sum_{j=1}^{N} \frac{\delta E}{\delta y_j^{(k)}} \right) = \sum_{j=1}^{N} \frac{\delta y_j^{(k)}}{\delta H2_k^{(k)}} \cdot \frac{\delta E}{\delta y_j^{(k)}}$$

$$\Delta H2_k^{(k)} = \sum_{j=1}^{N} \frac{\delta}{\delta H2_k^{(k)}} \left(\sum_{k=1}^{L2} H2_k . u_{jk} + br_{(j)} \right) . \Delta y_j^{(k)} \quad (14)$$

$$= \sum_{j=1}^{N} u_{jk} . \Delta y_j^{(k)}$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{H2}_{(L2,1)}^{(k)} = \boldsymbol{U}_{(N,L2)}^{T} \square \Delta \boldsymbol{y}_{(N,1)}^{(k)} \quad (15)$$

The inner gradient of the 2nd hidden layer nodes are expressed by

$$\frac{\delta E}{\delta h2_k^{(k)}} = \frac{\delta E}{\delta H2_k^{(k)}} \cdot \frac{\delta H2_k^{(k)}}{\delta h2_k^{(k)}} \Rightarrow \Delta h2_k^{(k)} = \Delta H2_k^{(k)} . f'\left(h2_k\right) \quad (16)$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{h2}_{(L2,1)}^{(k)} = \Delta \boldsymbol{H2}_{(L2,1)}^{(k)} \square * \boldsymbol{f}'\left(\boldsymbol{h2}_{(L2,1)}\right) \quad (17)$$

The local gradient of the 2nd hidden layer nodes are expressed by

$$\Delta v_{ki}^{(k)} = \frac{\delta E}{\delta h2_k^{(k)}} \cdot \frac{\delta h2_k^{(k)}}{\delta v_{ki}^{(k)}}$$

$$= \Delta h2_k^{(k)} . \frac{\delta}{\delta v_{ki}^{(k)}} \left(\sum_{i=1}^{L1} H1_i . v_{ki} + b_{h2(k)} \right)$$

$$= \Delta h2_k . H1_i$$

$$\Delta b_{h2(k)}^{(k)} = \frac{\delta E}{\delta h2_k^{(k)}} \cdot \frac{\delta h2_k^{(k)}}{\delta b_{h2(k)}^{(k)}}$$

$$= \Delta h2_k^{(k)} . \frac{\delta}{\delta b_{h2(k)}^{(k)}} \left(\sum_{i=1}^{L1} H1_i . v_{ki} + b_{h2(k)} \right) \quad (18)$$

$$= \Delta h2_k^{(k)}$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{V}_{(L2,L1)}^{(k)} = \Delta \boldsymbol{h2}_{(L2,1)}^{(k)} \square \left(\boldsymbol{H1}_{(L1,1)} \right)^T$$
$$\Delta \boldsymbol{b}_{h2(L2,1)}^{(k)} = \Delta \boldsymbol{h2}_{(L2,1)}^{(k)} \quad (19)$$

The outer gradient of each node in 1st hidden layer will receive the inner gradient of the 2nd hidden layer's nodes, so that

$$\frac{\delta E}{\delta H1_i^{(k)}} = \frac{\delta}{\delta H1_i^{(k)}} \left(\sum_{k=1}^{L2} \frac{\delta E^{(k)}}{\delta h2_k^{(k)}} \right) = \sum_{k=1}^{L2} \frac{\delta h2_k^{(k)}}{\delta H1_i^{(k)}} \cdot \frac{\delta E^{(k)}}{\delta h2_k^{(k)}}$$

$$\Delta H1_i^{(k)} = \sum_{k=1}^{L2} \frac{\delta}{\delta H1_i^{(k)}} \left(\sum_{j=1}^{M} H2_k . u_{jk} + br_{(j)} \right) \cdot \left(\sum_{j=1}^{L1} H1_i . v_{ki} + b_{h2(k)} \right) . \Delta h2_k^{(k)} \quad (20)$$

$$= \sum_{k=1}^{L2} v_{ki} . \Delta h2_k^{(k)}$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{H1}_{(L1,1)}^{(k)} = \boldsymbol{V}_{(L2,L1)}^{T} \square \Delta \boldsymbol{h2}_{(L2,1)}^{(k)} \quad (21)$$

The inner gradient of the 1st hidden layer nodes are expressed by

$$\frac{\delta E}{\delta h1_i^{(k)}} = \frac{\delta E}{\delta H1_i^{(k)}} \cdot \frac{\delta H1_i^{(k)}}{\delta h1_i^{(k)}} \Rightarrow \Delta h1_i^{(k)} = \Delta H1_i^{(k)} . f'\left(h1_i\right) \quad (22)$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{h1}_{(L1,1)}^{(k)} = \Delta \boldsymbol{H1}_{(L1,1)}^{(k)} .* \boldsymbol{f}'\left(\boldsymbol{H1}_{(L1,1)}\right) \quad (23)$$

The local gradient of the 1st hidden layer nodes are expressed by

$$\Delta w_{ij}^{(k)} = \frac{\delta E}{\delta h1_i^{(k)}} \cdot \frac{\delta h1_i^{(k)}}{\delta w_{ij}^{(k)}} \quad (15)$$

$$= \Delta h1_i^{(k)} . \frac{\delta}{\delta w_{ij}^{(k)}} \left(\sum_{j=1}^{N} x_j . w_{ij} + b_{h1(i)} \right)$$

$$= \Delta h1_i . x_j \quad (16)$$

$$\Delta b_{h1(i)}^{(k)} = \frac{\delta E}{\delta h1_i^{(k)}} \cdot \frac{\delta h1_i^{(k)}}{\delta b_{h1(i)}^{(k)}} \quad (17)$$

$$= \Delta h1_i^{(k)} . \frac{\delta}{\delta b_{h1(i)}^{(k)}} \left(\sum_{j=1}^{N} x_j . w_{ij} + b_{h1(i)} \right) \quad (24)$$

$$= \Delta h1_i^{(k)}$$

Generally in canonical form can be expressed by

$$\Delta \boldsymbol{W}_{(L1,N)}^{(k)} = \Delta \boldsymbol{h1}_{(L1,1)}^{(k)} \square \left(\boldsymbol{X}_{(N,1)} \right)^T$$
$$\Delta \boldsymbol{b}_{h1(L1,1)}^{(k)} = \Delta \boldsymbol{h1}_{(L1,1)}^{(k)} \quad (25)$$

All the weights on each layer (W, V, U and their biases) are updated using the following formula

$$\omega_{new} = \omega_{old} + m.\Delta\omega_{prev} + \eta.\Delta\omega$$

Where ω_{new} is the new weights, ω_{old} is the old weights, $\Delta\omega_{prev}$ is the gradient error of the weights on previous iteration, $\Delta\omega$ is the gradient error of the weights on current

iteration, and η is learning rate which controls how much the updating steps can affect the current values of the weights [23]. In certain situation, maybe the steepness of gradient trapped into local minima. To help the network out of local minima is by using the momentum (m) [24].

2.2 Preparation Strategi

Convergence is usually faster if the average of each input variable over the training set is close to zero. In this case, the data with the smallest value will have a negative sign, while the data with the largest value will have a positive sign. This heuristic need to be applied to all layers so that the average of the outputs of a node to be close to zero, because these outputs are the inputs to the next layer. This problem can be addressed by coordinating how the inputs are transformed with the choice of the tangent sigmoid activation function which expressed by

$$\tanh(x) = f(x) = \frac{\sinh(x)}{\cosh(x)} = \left(\frac{e^{x} - e^{-x}}{2}\right) / \left(\frac{e^{x} + e^{-x}}{2}\right) \quad (27)$$

The derivative of the tangent sigmoid function is expressed by

$$f'(x) = 1 - \left(f(x)\right)^{2} \quad (28)$$

To do this, the data samples need to be normalized using the following formula

$$\mu_x = \frac{1}{r}\sum_{i=0}^{r} d_i \qquad \sigma_x = \sqrt{\frac{1}{r}\sum_{i=0}^{r}\left(d_i - \mu_x\right)^2} \qquad xn_i = \frac{x_i - \mu_x}{\sigma_x} \quad (29)$$

Where d_i is the ith data samples, μ_x is the mean of data samples, σ_x is the variance of data samples, r is the number of data samples, x_i is the data samples selected as input, xn_i is the ith normalized data. The target values need to be set within in the range of the sigmoid so as to avoid saturating the output units. The simplest way is to normalize the target value by its maximum value, as follows

$$tn_i = \frac{t_i}{\max(d)} \quad (30)$$

Where t_i is the ith data samples selected as output (target value).

Assuming the inputs, x_i, to a unit is uncorrelated with variance 1, the standard deviation of the units weighted sums will be

$$\sigma_{x_i} = \sqrt{\sum_{j}\left(w_{ij}\right)^2} \quad (31)$$

To ensure that the σ_{x_i} are approximately 1, the weights should be randomly drawn from a normal distribution with mean zero and a standard deviation given by

$$\sigma_w = \sqrt{h} \quad (32)$$

where h is the number of inputs to unit.

2.3 Performance of DNN Training

The DNN training is done in such a way that the error function as in Eq. (7) can be minimized to close to or less than the target error. In order for network output to approximate the target value then the error target needs to be set close to zero. In this study use the error target = 1e-6. The net output of the training results needs to be normalized into the range of its original value. The performance of the result is measured by using MAPE (Mean Absolute Percentage Error) expressed by

$$MAPE = \frac{1}{M}\sum_{i=1}^{M}\frac{\left|y_{actual(i)} - y_{pred(i)}\right|}{y_{actual(i)}} \times 100\% \quad (33)$$

where M is the number of net outputs. This performance measure is also used in the validation stages of training results.

2.4 Datasets

In this study, the data of Percentage of GDP Distribution at Current Price by Industry Sector period 2001-2016 have been taken from the catalog of Statistical Yearbook of Indonesia. The dataset used is shown in Table 1. Data from 2001-2014 are used as training input data, while data from 2002-2015 are used as training targets. As data validation of training results used data in 2016. Table 1 shows that there are 9 industry sectors which contribute to GDP. Since the predicted is the percentage of GDP distribution per sector, there will be 9 inputs, as well as the number of outputs. Before use, training data input needs to be normalized using Eq. (29), while the training target is normalized using Eq. (30). The DNN used has 4 hidden layers as shown in Fig. 3. The MATLAB software has been used to implement DNN in the case of the intended prediction

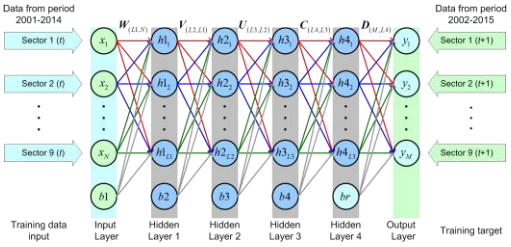

Figure 3. *DNN architecture used*

TABLE 1
UNITS FOR MAGENTIC PROPERTIES

Year	Agriculture, Livestock, Forestry, Fishery	Mining & Quarrying	Manufacturing Industry	Electricity, Gas, Water supply	Construction	Trade, Hotel, Restaurant	Transport & Communication	Finance, Real Estate & Business Services	Services	Total
	1	2	3	4	5	6	7	8	9	
2001	15.63	10.81	30.07	0.64	5.30	15.90	4.59	8.02	9.04	100.00
2002	15.46	8.83	28.72	0.84	6.07	17.14	5.38	8.48	9.08	100.00
2003	15.19	8.32	28.25	0.95	6.22	16.64	5.91	8.64	9.88	100.00
2004	14.34	8.94	28.07	1.03	6.59	16.05	6.20	8.47	10.31	100.00
2005	13.13	11.14	27.41	0.96	7.03	15.56	6.51	8.30	9.96	100.00
2006	12.97	10.98	27.54	0.91	7.52	15.02	6.93	8.06	10.07	100.00
2007	13.71	11.17	27.06	0.88	7.73	14.92	6.69	7.73	10.11	100.00
2008	14.40	10.97	27.87	0.82	8.46	13.97	6.31	7.43	9.77	100.00
2009	15.29	10.54	26.38	0.83	9.89	13.37	6.28	7.20	10.22	100.00
2010	15.29	11.16	24.80	0.76	10.25	13.69	6.56	7.24	10.25	100.00
2011	13.51	11.81	21.76	1.25	9.09	13.60	7.13	9.11	12.74	100.00
201	13.37	11.61	21.45	1.19	9.35	13.21	7.24	9.41	13.17	100.00

3 RESULT AND DISCUSSION

In this section, the testing results of percentage of GDP distribution per sector dataset in Indonesia Period 2001-2016 by using the DNN will be described. After the data preparation phase is performed, the DNN training is implemented using MATLAB software programming tool (M-File). Learning parameters used are alpha = 0.4, momentum = 0.005. By error target = 1e-6, the training is stopped after the 75584 iteration. The performance of training errors during the training process is shown in Fig. 4. Training result is shown in Fig. 5 with MAPE = 0.0192%. The trained DNN is then used to predict 2016 data using 2015 data as input. The predicted results are then validated using the 2016 data as shown in Fig. 6 and Table 2.

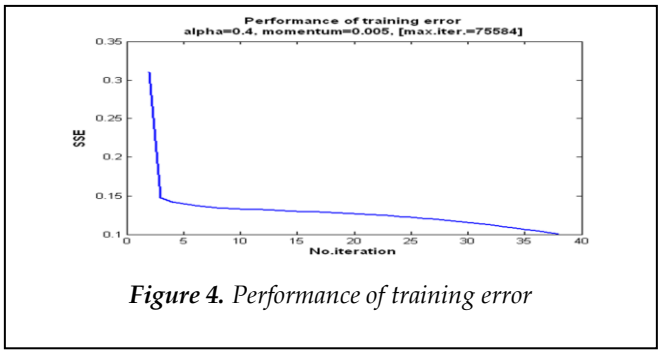

Figure 4. *Performance of training error*

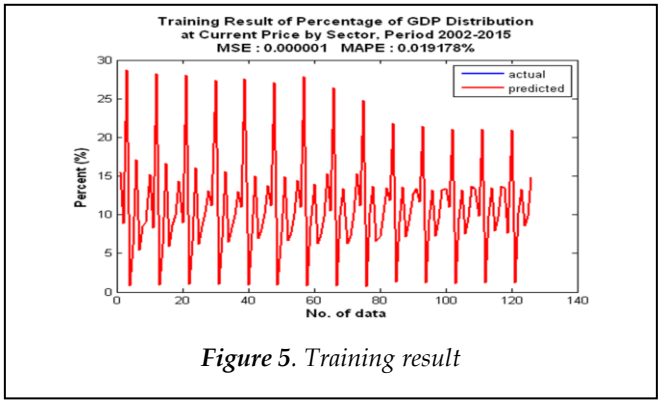

Figure 5. *Training result*

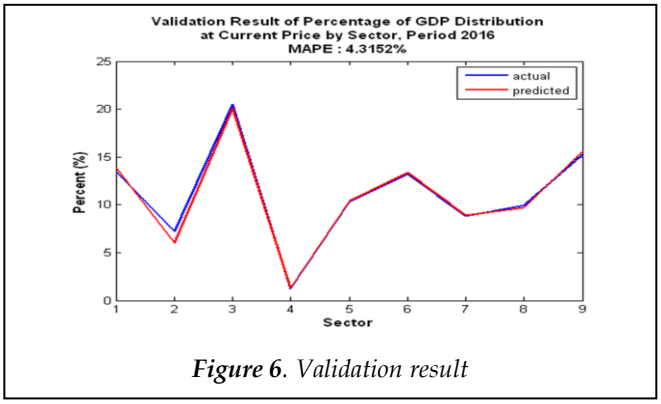

Figure 6. *Validation result*

From the validation of 2016 data that has been done, obtained MAPE = 4.3152%. From Table 2 it is found that the total predicted result is 99.43%. Since the total percentage of GDP distribution of all sectors should be 100%, it is necessary to normalize as shown in Table 3. The normalization as in Table 3 is obtained by the following formula

$$weight(i) = \text{Predict}(i) / \sum_{i=1}^{9} \text{Predict}(i)$$

$$\text{NewPredict}(i) = \text{Predict}(i) + weight(i) * \left(100 - \sum_{i=1}^{9} \text{Predict}(i) \right)$$

After normalization obtained NewMAPE = 4.53%. Using data in year 2016 as trained DNN inputs, then obtained a predicted percentage of GDP distribution per sector in year 2017 as shown in Table 4. By using the normalized predicted results in 2017 as DNN trained inputs, prediction results in 2018 will be obtained, and so on until the desired year

4 CONCLUSION

TABLE 2
DATA VALIDATION RESULT

Sector	Actual	Predict	APE (%)
1	13.45	13.84	2.86
2	7.21	6.04	16.17
3	20.51	20.13	1.87
4	1.22	1.34	10.11
5	10.38	10.44	0.58
6	13.19	13.39	1.49
7	8.84	8.92	0.95
8	9.93	9.69	2.37
9	15.27	15.64	2.44
Total		99.43	
MAPE (%)			4.32

TABLE 3
UNITS FOR MAGENTIC PROPERTIES

Sector	Predict	weight	New Pred.	Actual	New APE (%)
1	13.84	0.14	13.91	13.45	3.45
2	6.04	0.06	6.08	7.21	15.69
3	20.13	0.20	20.24	20.51	1.31
4	1.34	0.01	1.35	1.22	10.74
5	10.44	0.10	10.50	10.38	1.15
6	13.39	0.13	13.46	13.19	2.06
7	8.92	0.09	8.97	8.84	1.52
8	9.69	0.10	9.75	9.93	1.82
9	15.64	0.16	15.73	15.27	3.02
Total	99.43	1.00	100.00	New MAPE	4.53

In this paper, DNN has been used to predict the percentage of GDP distribution at current prices by industry sector. This research was used the data of Percentage of GDP Distribution at current price by industry sector in period 2001-2016 taken from the catalog of Statistical Yearbook of Indonesia. This experiment showed that the DNN is better results analysis in prediction activity. The experiment that has been done shows that DNN succeeded in predicting the next period quite well. This is evidenced by very small prediction errors. Nevertheless, the DNN training process in this study is still quite slow. Maybe for some prediction purposes, the accuracy of the results still needs to be improved. As future work, optimization methods get much better accuracy and speed up the training process is proposed.

ACKNOWLEDGMENT

The authors would like to express their heartfelt thanks to The Modern Computing Research Center, Department of Information Technology, State Polytechnic of Samarinda for providing all their support.

TABLE 4
UNITS FOR MAGENTIC PROPERTIES

Sector	Predict	weight	New Pred.
1	13.94	0.14	14.07
2	6.05	0.06	6.11
3	19.78	0.20	19.97
4	1.36	0.01	1.37
5	10.49	0.11	10.59
6	13.50	0.14	13.63
7	8.94	0.09	9.03
8	9.53	0.10	9.62
9	15.45	0.16	15.60
Total	99.05	1.00	100.00

REFERENCES

[1] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2005/2006," vol. 315.598, B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2006.
[2] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2007," vol. 315.598, B.-S. Indonesia, Ed., ed. Jakarta: BPS - Statistics Indonesia, 2007
[3] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2008," vol. 315.598, B.-S. Indonesia, Ed., ed. Jakarta: BPS - Statistics Indonesia, 2008.
[4] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2009," B.-S. Indonesia, Ed., ed. Jakarta: BPS - Statistics Indonesia, 2009.
[5] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2010," B.-S. Indonesia, Ed., ed: BPS - Statistics Indonesia, 2010.
[6] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2011.
[7] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2012
[8] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2013
[9] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2014
[10] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2015

[11] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2016
[12] B.-S. Indonesia, "Statistical Yearbook of Indonesia 2011," B.-S. Indonesia, Ed., ed. Jakarta - Indonesia: BPS - Statistics Indonesia, 2017
[13] J. Vrbka, "Predicting Future GDP Development by Means of Artificial Intelligence," Littera Scripta [online], vol. 9, pp. 154-167, 2016.
[14] T. Wang, "Forecast of Economic Growth by Time Series and Scenario Planning Method—A Case Study of Shenzhen," Modern Economy, vol. 07, pp. 212-222, 2016.
[15] S. R. Y. Mayankkumar B Patel, "Stock Price Prediction Using Artificial Neural Network," International Journal of Innovative Research in Science, Engineering and Technology (IJIRSET), (An ISO 3297: 2007 Certified Organization), vol. 3, June, 2014 2014.
[16] E. U. A. Gaffar, "Prediction of Regional Economic Growth in East Kalimantan using Genetic Algorithm," International Journal of Computing and Informatics (IJCANDI) vol. 1, pp. 58-67 May, 2016 2016.
[17] A. Dingli and K. S. Fournier, "Financial Time Series Forecasting - A Machine Learning Approach," Machine Learning and Applications: An International Journal, vol. 4, pp. 11-27, 2017.
[18] B. Al-hnaity and M. Abbod. (2016, Predicting Financial Time Series Data Using Hybrid Model. Intelligent Systems and Applications 650, 19-41.
[19] X.-N. Mergani Khairalla, Nashat T. AL-Jallad, "Hybrid Forecasting Scheme for Financial Time-Series Data using Neural Network and Statistical Methods," (IJACSA) International Journal of Advanced Computer Science and Applications, vol. 8, 2017.
[20] A. Sapronova, C. Meissner, and M. Mana, "Short time ahead wind power production forecast," Journal of Physics: Conference Series, vol. 749, p. 012006, 2016.
[21] J. L. Hong and K. Hong, "Flood Forecasting for Klang River at Kuala Lumpur using Artificial Neural Networks," International Journal of Hybrid Information Technology, vol. 9, pp. 39-60, 2016.
[22] N. Vivekanandan, "Prediction of Rainfall Using MLP and RBF Networks," International Journal Advanced Networking and Applications, vol. 5, pp. 1974-1979 2014.
[23] M. Hassan and M. Hamada, "Performance Comparison of Feed-Forward Neural Networks Trained with Different Learning Algorithms for Recommender Systems," Computation, vol. 5, p. 40, 2017.
[24] N. Buduma, Fundamentals of Deep Learning - Designing Next Generation Artificial Intelligence Algorithms, First Edition ed. United States of America: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2015