Journal of Computer Science

Original Research Paper

Predicting Depression Levels using Back Propagation Neural Network

[1]Eddy Muntina Dharma, [1]Yaya Heryadi, [2]Lukas, [3]Wayan Suparta and [1]Antoni Wibowo

[1]Department of Computer Science Bina Nusantara University, Jakarta, Indonesia
[2]Department of Cognitive Engineering Research Group, Atmajaya University, Jakarta, Indonesia
[3]Department of Electrical Engineering, Institut Teknologi Nasional Yogyakarta, Sleman, Indonesia

Article history
Received: 21-11-2022
Revised: 15-02-2022
Accepted: 19-02-2022

Corresponding Author:
Eddy Muntina Dharma
Department of Computer Science Bina Nusantara University, Jakarta, Indonesia
Email: eddy.dharma@binus.ac.id

Abstract: Depression is a mood disorder characterized by feelings of deep sadness and a sense of indifference. When depression recurs in moderate or severe intensity, it can be a serious health condition. The most effective way to deal with this problem is to predict the symptoms of depression at an early stage. In this study, a Back Propagation Neural Network (BPNN) model is proposed to predict whether a person is categorized as mild, moderate, or severe depression based on Beck's Depression Inventory (BDI) data. There are 21 BDI data items that are used as predictors in the built BPNN model. The dataset used is 227 patients and divided into 2 categories, namely 181 observations (80%) as training data and 46 observations (20%) as test data. While the BPNN model that is built has 21 neurons in the input layer, one hidden layer with 21 hidden neurons and 4 neurons in the output layer. After testing, it was found that the BPNN model is able to predict the level of depression with F1-Score of 100, 95.65, 90.91 and 95.24% for the classification of normal, mild depression, moderate depression and severe depression, respectively. Overall, the accuracy level reached 95.65%. This study concluded that the proposed model can help doctors or psychiatrists to predict depression at an early stage, whether it is classified as mild, moderate, or severe depression, so that the patient can receive appropriate treatment.

Keywords: Depression, Back Propagation Neural Network, Beck's Depression Inventory

Introduction

Based on data from the World Health Organization (WHO), depression is a mental disorder that affects more than 280 million people worldwide (WHO, 2021). It is characterized by persistent deep sadness, a sense of indifference and a lack of interest or pleasure in activities that were previously beneficial or pleasurable. A person is declared depressed if she/he has felt sad, hopeless, or worthless for 2 weeks. Depression is different from the usual fluctuations in mood and short-lived emotional responses toward challenges in everyday life. Especially when it is repeated and of moderate or severe intensity, depression can be a serious health condition. This can cause the affected person to suffer greatly and function poorly at work, at school and in the family, changes in appetite and weight, changes in sleep and activity quality, weakness, feelings of guilt, problems with thinking and make decisions (difficulty concentrating). This condition is very worrying, where at worst, depression can lead to suicide. Therefore, the ability to predict the depression at an early stage, whether it is classified as mild, moderate or severe depression, would be considered as the most effective method to overcome this problem. Thus, the patient can receive appropriate treatment.

Related Work

Several methods have been proposed to predict whether a person is categorized as depressed or not, including Lam *et al*. (2019), using a Convolutional Neural Network (CNN) algorithm with acoustic features as the predictor. This study produces performances including F1-Score of 0.87, Precision of 0.91 and Recall of 0.83. Furthermore, Sen *et al*. (2018), used s-MRI and rs-fMRI as predictors and produced an accuracy of 64.3%. In the study of Pinaya *et al*. (2019), also used s-MRI as a predictor and resulted in an accuracy of 63.9%. Next, Aghdam *et al*. (2018), using s-MRI and rs-fMRI data as predictors) and Deep Belief Network (DBN) algorithms. The dataset used is a combination of Autism Brain

Imaging Data Exchange I and II (ABIDE I and ABIDE II). DBN was used to focus on a combination of resting-state fMRI (rs-fMRI), Gray Matter (GM) and white matter (WM) data. The performance obtained is accuracy = 65.56%, sensitivity = 84%, specificity = 32.96%, F1 score = 74.76%. In the following year, Aghdam *et al*. (2019) again refined their study by adding Adam's optimization, so that the results obtained were showing an accuracy of = 0.7045, sensitivity = 0.679, specificity = 0.7421. Another study that also used the rs-fMRI predictor was by Wang *et al*. (2019) where they used prediction models such as SVM-RFE and Stacked Sparse Auto-Encoder (SSAE) and the accuracy obtained was 93.6%. Zhao *et al*. (2019) has used a Hierarchical Attention Transfer Network (HATN) approach, which analysed depression based on the emotions contained in the speech. The results of the trial on the Patient Health Questionnaire (PHQ)-8 scale which has a scale of [0, 24] show that the technique used Reaches Mean Square Error (RMSE) = 5.51 and Mean Absolute Error (MAE) = 4.20.

From those studies above, it is mostly limited to yield a prediction whether a person is categorized as depressed or not but also to predict whether a person was categorized as having mild, moderate, severe or normal depression. The model proposed in this study is called the Back Propagation Neural Network (BPNN) model accompanied by the use of a predictor called Beck's Depression Inventory (BDI) data, for a total of 21 items. BPNN was selected in this study because the dataset used has been classified, therefore, supervised learning is the suitable type of learning model to be applied where BPNN is one of the models of supervised learning.

While BDI is a psychometric test that is used to determine the symptoms that arise in someone who may be experiencing depression. The BDI was developed by the United States psychiatrist, Aaron T. Beck, with his colleagues and was first published in 1961. This test is one of the most frequently used psychometric tests and has been tested for validity in various studies in several countries and is considered consistent and feasible to use (Wang and Gorenstein, 2021).

Methods

The stages in this study is described in Fig. 1, starting from dataset collection then continued to training process, model testing and finally it is ended by calculating the accuracy.

Dataset Determination

The dataset used in this study was taken from the Denpasar Mental Health Centre, Bali, Indonesia with the number of dataset is 227 patients, within 4 classification patient's condition such as: 63 People are normal patients, 50 mild patients, 68 Moderate patients and 46 Severe patients. Those 227 data, then divided into 181 data (80%) as training data and 46 data (20%) as test data. Sample of the used dataset chunk is shown on the Fig. 2.

The number of predictor features used as input data is 21 predictors representing the scale of: Feelings, pessimism, feeling of failure, satisfaction, feeling guilty, feeling punished, hate for yourself, self-pattern, wishes of self-denial, cry irritability, social withdrawal, indecisiveness, physical identity, reduce productivity, sleep disorders, easy fatigue, loss of appetite, weight loss, subchondral disturbances and loss of libido.

Furthermore, all the predictor values above are normalized to a value between [0,1].

The number of target (output) patterns is one, with different 4 classifications as follows:

1. NORMAL (Normal)
2. RINGAN (Mild depression)
3. SEDANG (Moderate depression)
4. BERAT (Severe Depression)

The target feature is encoded using one-hot encoding, as it shown on the Table 1.

One Hot Encoding is a process in which data processing is applied towards categorical data, converted into a binary vector representation and applied on machine learning algorithms. In this study, One Hot Encoding was used because of the target (output) patterns of the used dataset was in the form of categorical data within 4 categories (Normal, Mild, Moderate, Severe). In fact, the use of other encodings such as text encoding or integer encoding on categorical data may result in poor performance or unexpected results, therefore, one hot encoding is selected. So that, the dataset in Fig. 2 will change as shown in Fig. 3.

Determining the BPNN Architecture

The BPNN architecture used consists of 3 layers, called Input Layer, Hidden Layer and Output Layer (Fig. 4).

The number of neurons in each layer can be briefly described as follows:

- The Input Layer consists of 21 neurons, which receive 21 predictor inputs (X_1, X_2, X_3, ..., X_{21})
- Hidden Layer consists of 21 neurons. There is only one hidden layers used in this study. According to Fausett (1993), one hidden layer is sufficient for a backpropagation net to approximate any continuous mapping from then input patterns to the out patterns to an arbitrary degree of accuracy (Fausett, 1993). In addition, within one hidden layer, it can speed up the training process. In addition, in determining the optimal number of neurons in the hidden layer, Genetic Algorithm (GA) is used as it applied by

Ding *et al*. (2011). Finally, ffter applying GA, the optimal neuron obtained at the hidden layer is 21.

- Output Layer consists of 4 neurons (Y_1, Y_2, Y_3, Y_4)

Training Using BPNN Algorithm

A. **Prepare Dataset.** In this study, the data set is stored in a formal MS-Excel file. The following is part of the program to read files from excel using matlab.

```
% Read from excel file using matlab
filename    = 'dataset_bdi.xlsx';
sheet       = 2;
xlRange     = 'D2:AB182';
Data        = xlsread(filename, sheet, xlRange);
```

B. **Initialization of** Target Mean Square Error (MSE), Learning rate (α) and assigning values of synaptic weights V_{ij} and W_{jk} randomly.

```
% V: weight between input layer and hidden layer
V           = rand 21,21);
% W: weight between hidden layer and output layer
W           = rand (4,21);
% b_hid: bias on hidden layer
b_hid       = rand (21,1);
% b_out: bias on hidden layer
b_out       = rand (4,1);
% MSE target
target_MSE  = 0.001;
% lr:learning rate
lr          = 0.1;
% Save to text file
writetable(table(V), 'param_V.txt')
writetable(table(W), 'param_W.txt')
writetable(table(b_hid), 'param_b_hid.txt')
writetable(table(b_out), 'param_b_out.txt')
```

C. Employing Training to the Network

The parameters set during training are: Epoch = 0 and MSE = 1.

Performing these following steps during (MSE>Target error)

1. Epoch = Epoch + 1
2. For each pair of elements to be studied, do:

Forward Propagation

i. Each input neuron (X_i) receives a signal and forwards the signal to all neurons in the hidden layer.
ii. Each neuron in the hidden layer (Z_in_j) adds up the weighted input signals (Russell and
iii. Norvig, 2002):

$$Z\_in_j = b_j^{hid} + \sum_{i=1}^{n} V_{ij} . X_i \quad (1)$$

V_{ij} = Weight between the i neuron input with the *i* hidden neuron
b^{hid}_j = Bias weight of the *j* hidden neuron on hidden layer
X_i = Entry from the *i* neuron input
n = Total number of neuron input
Z_in_j = The sum of weighing output signal from the hidden layer of the *j* neuron unit

Use the sigmoid activation function to calculate the output signal (Russell and Norvig, 2002):

$$Z_j = \frac{1}{1 + e^{-z\_inj}} \quad (2)$$

where, Z_j would be the output of hidden layer on the *j* hidden neuron.

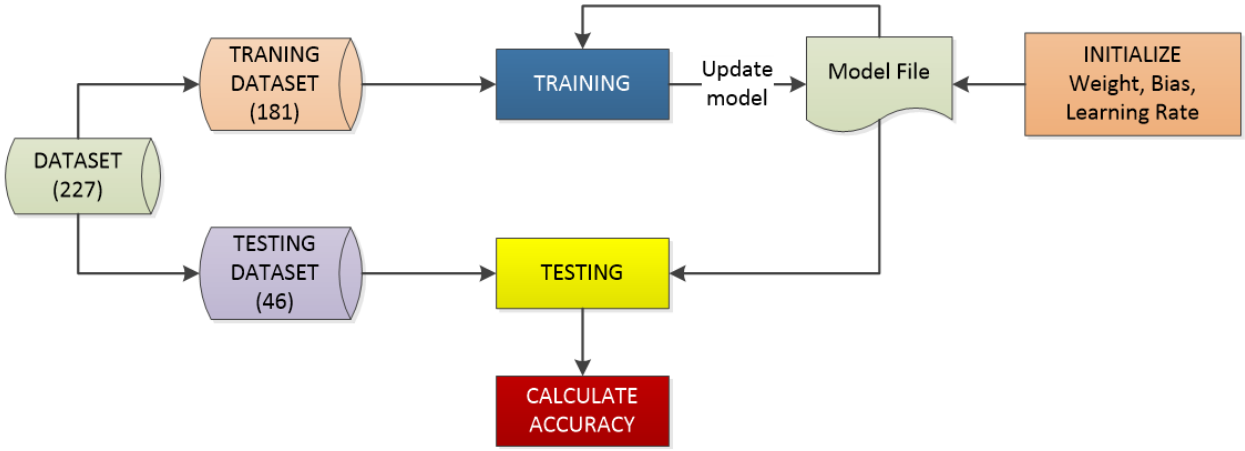


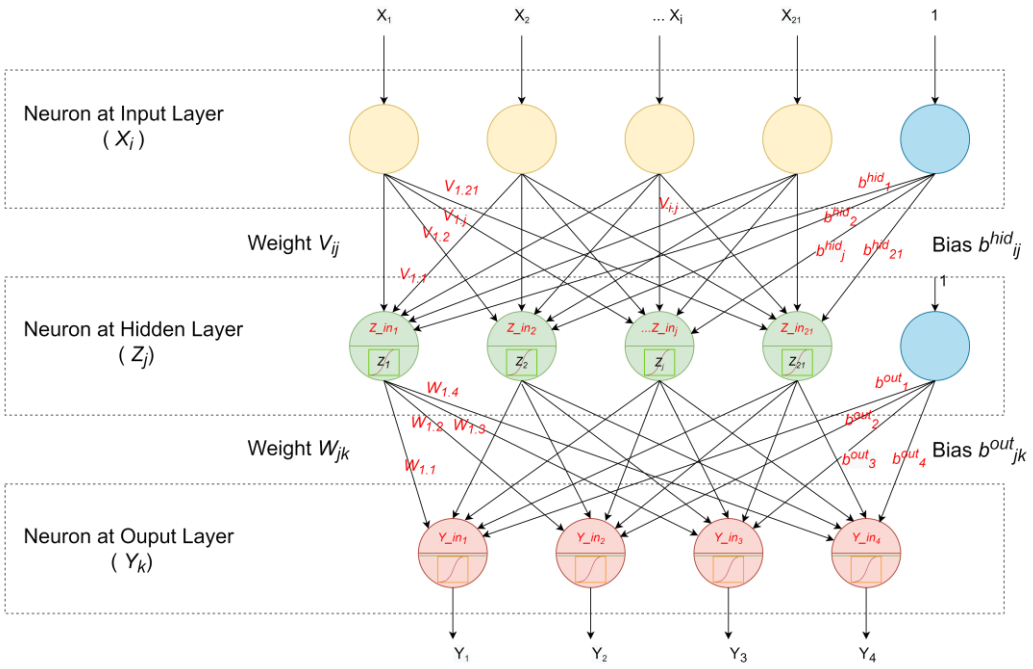
Fig. 1: Research flow prediction of depression level with BPNN

Nama	umur	jk	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15	p16	p17	p18	p19	p20	p21	Diagnosa
***	24	P	1	2	2	2	1	1	2	2	1	1	2	2	2	2	2	1	2	2	1	3	1	BERAT
***	27	P	1	0	0	1	2	1	1	0	0	3	0	0	0	2	2	1	1	0	0	0	0	RINGAN
***	27	L	1	0	1	1	2	1	1	0	0	0	1	0	0	1	0	1	1	1	0	1	0	RINGAN
***	35	L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NORMAL
***	26	P	2	1	2	1	2	1	2	1	1	3	1	2	1	1	1	2	1	0	0	2	2	SEDANG
***	55	P	1	1	0	1	2	1	1	2	0	1	1	1	1	2	2	2	1	2	1	1	2	SEDANG
***	29	P	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	0	0	SEDANG
***	11	P	0	0	0	0	2	0	0	1	0	0	1	0	2	0	1	1	0	1	0	1	1	RINGAN
***	28	P	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	1	NORMAL
***	31	P	0	1	1	0	0	0	0	1	0	0	1	0	1	1	1	0	0	0	0	0	0	NORMAL
***	27	P	1	1	1	1	1	2	3	1	1	0	2	1	2	3	1	1	2	2	0	1	1	SEDANG
***	30	L	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	NORMAL
***	25	L	1	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	NORMAL
***	25	P	1	1	0	0	2	1	1	0	0	0	1	1	2	0	1	1	1	0	0	1	2	RINGAN
***	23	P	1	1	1	3	0	1	1	0	1	0	1	1	0	3	1	1	1	0	0	0	3	SEDANG
***	24	P	1	2	2	2	1	1	2	2	1	1	2	2	2	2	2	1	2	2	1	3	1	BERAT
***	26	L	3	1	2	1	2	3	3	3	3	1	1	1	2	0	2	1	2	1	0	1	0	BERAT
***	28	P	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	2	NORMAL
***	20	L	3	3	1	1	1	1	1	2	1	1	1	2	2	0	2	1	1	0	0	0	0	SEDANG

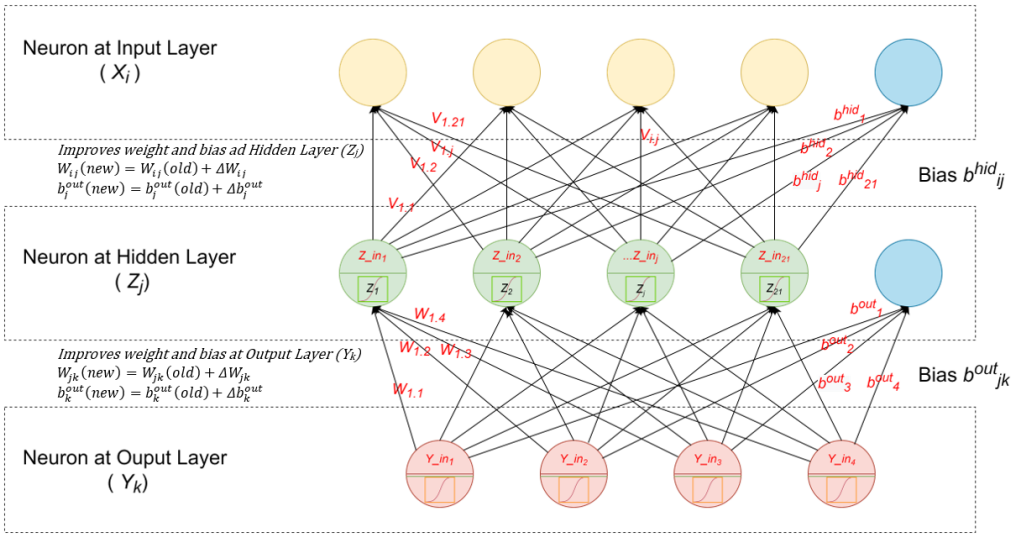
Fig. 2: Example of the dataset used

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	Normal	Mild	Moderate	Severe
0.33	0.67	0.67	0.67	0.33	0.33	0.67	0.67	0.33	0.33	0.67	0.67	0.67	0.67	0.67	0.33	0.67	0.67	0.33	1.00	0.33	0	0	0	1
0.33	0.00	0.00	0.33	0.67	0.33	0.33	0.00	0.00	1.00	0.00	0.00	0.00	0.67	0.67	0.33	0.33	0.00	0.00	0.00	0.00	0	1	0	0
0.33	0.00	0.33	0.33	0.67	0.33	0.33	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.33	0.33	0.33	0.00	0.33	0.00	0	1	0	0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1	0	0	0
0.67	0.33	0.67	0.33	0.67	0.33	0.67	0.33	0.33	1.00	0.33	0.67	0.33	0.33	0.33	0.67	0.33	0.00	0.00	0.67	0.67	0	0	1	0
0.33	0.33	0.00	0.33	0.67	0.33	0.33	0.67	0.00	0.33	0.33	0.33	0.33	0.67	0.67	0.67	0.33	0.67	0.33	0.33	0.67	0	0	1	0
0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.67	0.33	0.33	0.33	0.33	0.33	0.00	0.00	0	0	1	0
0.00	0.00	0.00	0.00	0.67	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.67	0.00	0.33	0.33	0.00	0.33	0.00	0.33	0.33	0	1	0	0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.33	0.33	0.33	0.00	0.00	0.33	1	0	0	0
0.00	0.33	0.33	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.33	0.33	0.33	0.00	0.00	0.00	0.00	0.00	0.00	1	0	0	0
0.33	0.33	0.33	0.33	0.33	0.67	1.00	0.33	0.33	0.00	0.67	0.33	0.67	1.00	0.33	0.33	0.67	0.67	0.00	0.33	0.33	0	0	1	0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.33	0.33	0.33	0.00	0.00	0.00	0.00	1	0	0	0
0.33	0.00	0.00	0.33	0.00	0.33	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.00	1	0	0	0
0.33	0.33	0.00	0.00	0.67	0.33	0.33	0.00	0.00	0.33	0.33	0.67	0.00	0.33	0.33	0.33	0.33	0.00	0.00	0.33	0.67	0	1	0	0
0.33	0.33	0.33	1.00	0.00	0.33	0.33	0.00	0.33	0.00	0.33	0.33	0.00	1.00	0.33	0.33	0.33	0.00	0.00	0.00	1.00	0	0	1	0
0.33	0.67	0.67	0.67	0.33	0.33	0.67	0.67	0.33	0.33	0.67	0.67	0.67	0.67	0.67	0.33	0.67	0.67	0.33	1.00	0.33	0	0	0	1
1.00	0.33	0.67	0.33	0.67	1.00	1.00	1.00	1.00	0.33	0.33	0.33	0.67	0.00	0.67	0.33	0.67	0.33	0.00	0.33	0.00	0	0	0	1
0.33	0.33	0.33	0.33	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.33	0.00	0.00	0.00	0.67	1	0	0	0
1.00	1.00	0.33	0.33	0.33	0.33	0.33	0.67	0.33	0.33	0.33	0.67	0.67	0.00	0.67	0.33	0.33	0.00	0.00	0.00	0.00	0	0	1	0
0.33	0.67	0.33	0.67	0.33	0.33	0.67	0.67	0.33	1.00	1.00	1.00	0.67	0.00	0.67	0.33	0.33	0.67	0.00	0.00	0.00	0	0	0	1

Fig. 3: Example of Normalized and encoding dataset



(a) Forward propagation



(b) Backward Propagation

Fig. 4: BPNN architecture used

Table 1: One-hot encoding for target feature

Classification	Normal	Mild	Moderate	Severe
Normal	1	0	0	0
Mild	0	1	0	0
Moderate	0	0	1	0
Severe	0	0	0	1

Then send the signal to all units at the output layer. The sigmoid function is used as a transfer function with the consideration that it has a gradient proportional towards the output reflection.

iii. Each output neuron (Y_in_k) sums the weighing *input*, under this formula (Russell and Norvig, 2002):

$$Y\_in_k = b_k^{out} + \sum_{j=1`}^{n} W_{jk}.Z_j \quad (3)$$

W_{ij} = Weight between the j hidden neuron and the k output neuron
b^{out}_j = bobot bias weight of the j output = neuron on output layer
Z_j = hidden layer output towards the j hidden neuron
n = Total numbers of output neuron
Y_in_j = The sum of weighing output signal on the k output neuron unit

Perform the sigmoid activation function to calculate the output signals (Russell and Norvig, 2002).
where Y_k would be the output of hidden layer on the k hidden neuron:

$$Y_k = \frac{1}{1+e^{-Y\_in_k}} \quad (4)$$

iv. Calculating the Error Value Error (E) is the difference between the desired output value (T = Target of pattern) within the exact output from the learning result NN (Y_k) as follows (Russell and Norvig, 2010):

$$E = T - Y \quad (5)$$

Sum Square Error (SSE) In the output layer of the artificial neural network of advanced bait is (Russell and Norvig, 2002):

$$SSE = \sum_{k=1}^{n} \left(T_k - Y_k\right)^2 \quad (6)$$

where, n is the number of neurons at the output layer; T_k is the output target in the k-neuron.

The steps of forward propagation above, written in Matlab:

```
% Forward Propagation
% X: Input Layer
X    = Data(i,1:21)';
% T: Target
T    = Data(i,22:25)';
% Zin: hidden layer
% Z: sigmoid of Z_in
Zin  = V * X + b_hid;
```

```
Z     = sigmoid (dlarray(Z_in));
% Yin:output layer
% Y:sigmoid dari Y_in
Yin   = W * Z + b_out;
Y     = sigmoid (dlarray(Y_in));
% E: Error
E     = T - Y;
SSE = SSE + dot(E,E);
```

Backward Propagation

i. Each unit of output (Y_k) receives a target pattern related to the learning input pattern, calculating its error information (Russell and Norvig, 2002):

$$\delta_k = E * Y_k \left(1 - Y_k\right) \tag{7}$$

where, δ_k is the error information (delta) in the k-output neuron.

Then calculate the weight correction (which will be used to fix the value of W_{jk}) (Russell and Norvig, 2002):

$$\Delta W_{jk} = \alpha.\delta_k \tag{8}$$

where, ΔW_{jk} is the change in weight variable between hidden layer and output layer; α is the learning rate value;

Also calculate the correction of the bias in the output layer (which will later be used to fix the value $b^{out}{}_k$) (Russell and Norvig, 2002):

$$\Delta b_k^{out} = \alpha.\delta_k \tag{9}$$

where, $\Delta b^{out}{}_k$ is the change in refractive weight on the k-output neuron

ii. Each neuron has a hidden layer (Z_j) summing its input delta (from the units at the next layer). (Russell and Norvig, 2002):

$$\delta\_in_j = \sum_{k=1}^{n} \delta_k.W_{jk} \tag{10}$$

Multiply this value by a derivative of its activation function to calculate error information (Russell and Norvig, 2002):

$$\delta_j = \delta\_in_j * Z_j \left(1 - Z_j\right) \tag{11}$$

where, δ_j is the error information (delta) in the j-hidden neuron.

Then calculate the weight correction (which will later be used to update the V_{ij} value) (Russell and Norvig, 2002):

$$\Delta V_{ij} = \alpha.\delta_j.X_i \tag{12}$$

where, ΔV_{ij} is the change in weight variable between the input layer and the hidden layer.

Also calculate the bias correction (which will later be used to change the value of $b^{hid}{}_j$) (Russell and Norvig, 2002):

$$\Delta b_j^{hid} = \alpha.\delta_j \tag{13}$$

where $\Delta\, b^{hid}{}_j$ is the change in the bias weight of j-neurons at the hidden layer

iii. Each unit at the output layer (Y_k) improves its weight and bias (Russell and Norvig, 2002):

$$W_{jk}(new) = W_{jk}(old) + \Delta W_{jk} \tag{14}$$

$$b_k^{out}(new) = b_k^{out}(old) + \Delta b_k^{out} \tag{15}$$

Each hidden *layer* unit (Z_j) improves its weight and bias (Russell and Norvig, 2002):

$$V_{jk}(new) = V_{jk}(old) + \Delta V_{jk} \tag{16}$$

$$b_j^{hid}\left(new\right) = b_j^{hid}(old)\Delta b_j^{hid} \tag{17}$$

Calculating MSE (Russell and Norvig, 2002):

$$MSE = \frac{1}{n}\sum_{i=1}^{n} SSE_i \tag{18}$$

where, SSE_i is the SSE of each epoch and n is the number of epochs.

The steps of backward propagation above is written in Matlab:

```
% Backward Propagation
% Update Bobot W & bias on output layer
delta_out = E.*Y.*(1-Y);
dW      = lr.*delta_out*Z';
db_out = lr.*delta_out;
W       = W + dW; % --> Update Bobot W
b_out   = b_out + db_out; % --> Update bias on Output Layer
% Update Bobot V & bias on hidden layer
delta_hid = (delta_out'*W)'.*Z.*(1-Z);
dV      = lr.*delta_hid*X';
```

```
db_hid = lr.*delta_hid;
V      = V + dV; % --> Update Bobot V
b_hid  = b_hid + db_hid; % --> Update bias on Hidden Layer
```

The steps for forward and backward calculations above are for one training cycle (one epoch) so they must be repeated until the specified number of epochs or the desired MSE has been reached. In this study, only MSE was used as a reference for the training cycle. The MSE target used is: 0.001.

The final result of the BPNN training is the obtaining of V_{ij}, W_{jk}, b^{hid}_j, b^{out}_k weights which are then stored for the testing phase.

BPNN training algorithm in complete source code using Matlab can be seen in the Appendix 1 of this study.

After the training process is complete, the next process is to test the result of BPNN model with 46 data testing.

Testing

The weights of V_{ij}, W_{jk} and bias b^{hid}_j, b^{out}_k generated in the training process will be used to test the test dataset, with the following steps:

1. Initialization of weights and biases according to the training results, namely V_{ij}, W_{jk}, b^{hid}_j, b^{out}_k
2. For each input pattern do the following steps:

    i. Each input neuron (X_i) in the input layer receives a signal and forwards the signal to all neurons in the hidden layer
    ii. Each neuron in the hidden layer (Z_in_j) adds up the weighted input signals (Russell and Norvig, 2002):

$$Z\_in_j = b_j^{hid} + \sum_{i=1}^{n} V_{ij}.X_i \tag{19}$$

Use the sigmoid activation function to calculate its output signal (Russell and Norvig, 2002):

$$Z_j = \frac{1}{1+e^{-z-in_j}} \tag{20}$$

    iii. Each *output* neuron (Y_in_k) sums up the weighted *input* signals (Russell and Norvig, 2002):

$$Y - in_k = b_k^{out} + \sum_{j=1}^{n} W_{jk}.Z_j \tag{21}$$

Use the sigmoid activation function to calculate its output signal (Russell and Norvig, 2002):

$$Y_k = \frac{1}{1+e^{-Y-in_k}} \tag{22}$$

where, Y_k is the output of the neural network.

Testing Algorithm in complete Matlab code can be seen in the Appendix 2 of this study.

Results and Discussion

The training process uses 181 training data, 21 predictors and one target with 4 classifications. Then the parameters used in the training process are MSE target of a validation set is 0.001 and learning rate value is 0.1. After the training process is carried out, the number of epochs = 1,695 to achieve an MSE of a validation set of 0.001, with a training time of 208 seconds. The training result can be shown in Fig. 5.

After the MSE target is achieved, the optimal weights and biases for the BPNN model are found. The optimal weights and biases will be tested on 46 test data. The performance of the BPNN model is presented in Table 2. Of the 46 data tested, 44 data were found to be classified correctly, or resulting an accuracy of 95.65%.

The result presented in Table 2 reveals that BPNN model is able to predict the level of depression with F1-Score of 100, 95.65, 90.91 and 95.24% for the classification of normal, mild depression, moderate depression and severe depression, respectively.

However, the model that was suggested would achieve the best accuracy if it precisely using 21 predictors. This result then supported as after conducting trial and error using 5 or 10 predictor or below 21 predictors, the accuracy was not really satisfying. This is also supported by BDI's suggestion that these 21 predictors are valid and have been tested by Beck *et al*. (1961).

Table 2: Performance of BPNN model

	Classified as								
Class	Normal	Mild	Moderate	Severe	TPR (%)	FNR (%)	PPV (%)	F1-Score (%)	Accuracy (%)
Normal	13	0	0	0	100	0	100	100	95.65
Mild	0	11	1	0	91.67	8.33	100	95.65	
Moderate	0	0	10	1	90.91	9.09	90.91	90.91	
Severe	0	0	0	10	100	0	90.91	95.24	

TPR = True Positive Rate (Recall); FNR = False Negative Rate; PPV = Positice Predictive Value (Precision)

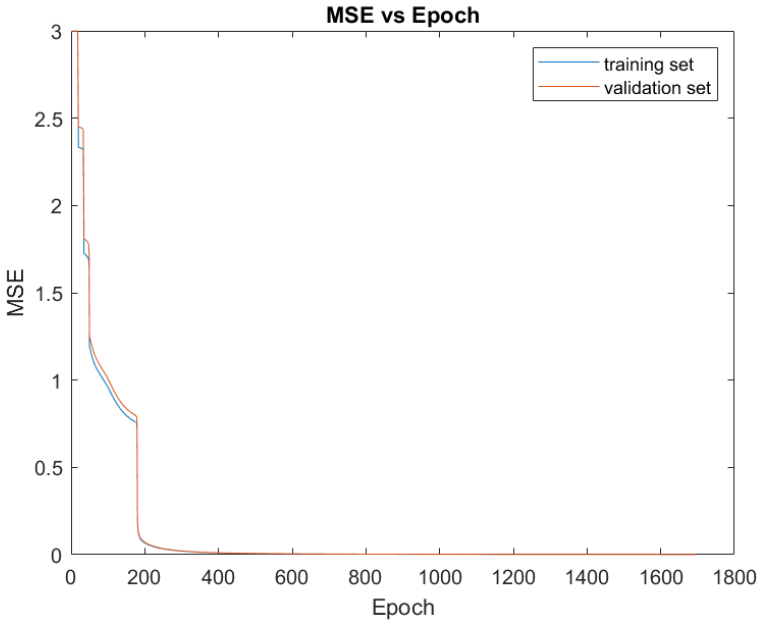


Fig. 5: Training Result (MSE vs Epoch)

Conclusion and Recommendation

Prediction of depression level with BPNN model with one hidden layer shows a good performance, reaching 95.65% accuracy, with 181 datasets as training data and 46 data as test data with 21 BDI data items that are used as predictors. In this study, the proposed model is able to predict the level of depression with F1-Score of 100 95.65 90.91 and 95.24% for the classification of depression: normal, mild, moderate and severe, respectively.

From the results achieved in this study, the proposed model can be applied to help doctors or psychiatrists to predict depression at an early stage, whether it is classified as mild, moderate, or severe depression, so that the patient can receive appropriate treatment.

Acknowledgement

Thanks to Denpasar Mental Health Center for the sample data and the doctor on duty, dr. I Putu Belly Sutrisna, M. Biomed, Sp.KJ and dr. Nyoman Widhyalestari Parwatha, Sp.KJ for all the input and suggestions.

Author's Contributions

Eddy Muntina Dharma: Contributed to collecting dataset, design the research methodology, data-analysis, conducting experiments and writing of the manuscript.

Yaya Heryadi: Advise data-analysis and design the research methodology.

Lukas: Contributed to conducting experiments.

Wayan Suparta: Contributed to conducting experiments, data-analysis and writing of the manuscript.

Antoni Wibowo: Advise research methodology.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Aghdam, M. A., Sharifi, A., & Pedram, M. M. (2018). Combination of rs-fMRI and sMRI data to discriminate autism spectrum disorders in young children using deep belief network. Journal of digital imaging, 31(6), 895-903. doi.org/10.1007/s10278-018-0093-8

Aghdam, M. A., Sharifi, A., & Pedram, M. M. (2019). Diagnosis of autism spectrum disorders in young children based on resting-state functional magnetic resonance imaging data using convolutional neural networks. Journal of digital imaging, 32(6), 899-918. doi.org/10.1007/s10278-019-00196-1

Beck, A. T., Ward, C. H., Mendelson, M., Mock, J., & Erbaugh, J. (1961). An inventory for measuring depression. Archives of general psychiatry, 4(6), 561-571. doi.org/10.1001/archpsyc.1961.01710120031004

Ding, S., Su, C., & Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. Artificial intelligence review, 36(2), 153-162. doi.org/10.1007/s10462-011-9208-z

Fausett, L. V. (1993). Fundamentals of Neural Networks: Architectures, Algorithms and Applications (1st Edition). Pearson. ISBN: 978-0133341867

Lam, G., Dongyan, H., & Lin, W. (2019, May). Context-aware deep learning for multi-modal depression detection. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 3946-3950). IEEE. doi.org/10.1109/ICASSP.2019.8683027

Pinaya, W. H., Mechelli, A., & Sato, J. R. (2019). Using deep autoencoders to identify abnormal brain structural patterns in neuropsychiatric disorders: A large-scale multi-sample study. Human brain mapping, 40(3), 944-954. 40, 944-954. doi.org/10.1002/hbm.24423

Russell, S., & Norvig, P. (2002). Artificial intelligence: A modern approach. ISBN: 978-0-13-604259-4

Sen, B., Borle, N. C., Greiner, R., & Brown, M. R. (2018). A general prediction model for the detection of ADHD and Autism using structural and functional MRI. PloS one, 13(4), e0194856. doi.org/10.1371/journal.pone.0194856

Wang, C., Xiao, Z., Wang, B., & Wu, J. (2019). Identification of autism based on SVM-RFE and stacked sparse auto-encoder. Ieee Access, 7, 118030-118036. doi.org/10.1109/ACCESS.2019.2936639

Wang, Y. P., & Gorenstein, C. (2021). The Beck depression inventory: Uses and applications. In The Neuroscience of Depression (pp. 165-174). Academic Press. doi.org/10.1016/B978-0-12-817933-8.00020-7

WHO. (2021). Depression. Depression Fact Sheet of World Health Organization. https://www.who.int/news-room/fact-sheets/detail/depression

Zhao, Z., Bao, Z., Zhang, Z., Deng, J., Cummins, N., Wang, H., ... & Schuller, B. (2019). Automatic assessment of depression from speech via a hierarchical attention transfer network and attention autoencoders. IEEE Journal of Selected Topics in Signal Processing, 14(2), 423-434. doi.org/10.1109/JSTSP.2019.2955012s

Appendix 1

training.m

```
% Predicting Depression Level Using BPNN
% Eddy Muntina Dharma
% eddy.dharma@binus.ac.id
% ----------------------------------------

clc;clear;close all;warning off all;
tic;
% Read from excel file
filename = 'dataset_bdi.xlsx';
sheet = 2;
xlRange = 'D2:AB182';

Data = xlsread(filename, sheet, xlRange);

% V : weight between input layer and hidden layer
V=importdata('param_V.txt').data;
% W : weight between hidden layer and output layer
W=importdata('param_W.txt').data;
% b_hid : bias on hidden layer
b_hid=importdata('param_b_hid.txt').data;
% b_out : bias on hidden layer
b_out=importdata('param_b_out.txt').data;
% lr : learning rate;
lr = 0.1;

% Sample Data = 227, Training 181 and Testing 46
jml_data   = 181;
jml_epoch  = 0;
target_MSE = 0.001;
MSE = 1;
fileMSE = fopen('repMSE.txt','a');
```

```
while (MSE>target_MSE)
    jml_epoch = jml_epoch + 1;
    SSE = 0;
    for i=1:jml_data
        % Forward
            % X : Input Layer
            X = Data(i,1:21)';
            % T : Target
            T  = Data(i,22:25)';

            % Z_in : hidden layer; Z : sigmoid of Z_in
            Z_in = V * X + b_hid;
            Z = sigmoid (dlarray(Z_in));

            % Y_in : output layer; Y : sigmoid of Y_in
            Y_in = W * Z + b_out;
            Y = sigmoid (dlarray(Y_in));

            % E : Error
            E   = T - Y;
            SSE = SSE + dot(E,E);

        % Backward
            % Update weight W & bias on output layer
            delta_out = E.*Y.*(1-Y);
            dW = lr.*delta_out*Z';
            db_out = lr.*delta_out;
            W = W + dW; % --> Update weight W
            b_out = b_out + db_out; % --> Update bias on Output Layer

            % Update weight V & bias on hidden layer
            delta_hid = (delta_out'*W)'.*Z.*(1-Z);
            dV = lr.*delta_hid*X';
            db_hid = lr.*delta_hid;
            V = V + dV; % --> Update weight V
            b_hid = b_hid + db_hid; % --> Update bias on Hidden Layer
    end

    % Each epoch saves weights and biases to a file
    writetable(table(extractdata(V)), 'param_V.txt');
    writetable(table(extractdata(W)), 'param_W.txt');
    writetable(table(extractdata(b_hid)), 'param_b_hid.txt');
    writetable(table(extractdata(b_out)), 'param_b_out.txt');

    % Save MSE
    MSE = SSE/jml_data;
    fprintf(fileMSE,'%g\n',MSE);

end
fclose(fileMSE);
toc;

dataMSE=importdata('repMSE.txt');
figure
plot(dataMSE)
title('MSE vs Epoch')
xlabel('Epoch')
ylabel('MSE')
```

Appendix 2

testing.m

```
% Predicting Depression Level Using BPNN
% Eddy Muntina Dharma
% eddy.dharma@binus.ac.id
% ----------------------------------------

clc;clear;close all;warning off all;

% Read from excel file
filename = 'dataset_bdi.xlsx';
sheet = 3;
xlRange = 'D2:AB47';

Data = xlsread(filename, sheet, xlRange);

% V : weight between input layer and hidden layer
V=importdata('param_V.txt').data;
% W : weight between hidden layer and output layer
W=importdata('param_W.txt').data;
% b_hid : bias on hidden layer
b_hid=importdata('param_b_hid.txt').data;
% b_out : bias on hidden layer
b_out=importdata('param_b_out.txt').data;

benar=0;
jml_data=46;
for i=1:jml_data
        % Forward
            % X : Input Layer
            X = Data(i,1:21)';
            % T : Target
            T  = Data(i,22:25)';

            % Z_in : hidden layer; Z : sigmoid of Z_in
            Z_in = V * X + b_hid;
            Z = sigmoid (dlarray(Z_in));

            % Y_in : output layer; Y : sigmoid of Y_in
            Y_in = W * Z + b_out;
            Y = sigmoid (dlarray(Y_in));

            % E : Error
            E   = abs(T - round(Y));
            if (sum(E)==0)
                benar=benar+1;
            end
end
akurasi=benar/jml_data*100

"Amount of Data Testing : " + jml_data
"Classified correctly   : " + benar
"Accurate               : " + akurasi + "%"
```