Classification Task of Radar Returns in the Ionosphere Using Genetic Algorithm in the Neural Network

Novia Arum Sari

Research School of Computer Science
Australian National University, Canberra ACT, 0200 Australia
u6027508@anu.edu.au

Abstract. This paper presents the implementation of the simple feed-forward neural network, random noise feed-forward neural network, and the genetic algorithm implementation in the neural network for classification of radar returns in the ionosphere dataset. The simple feed-forward neural network works by using backpropagation algorithm that implements cross-entropy as the activation function and stochastic gradient descent (SGD) as the gradient descent optimiser in one single hidden layer. Moreover, the random noise feed-forward neural network works in the same way as the simple feed-forward but is done by adding random noise to the neural network input data during the training process. The genetic algorithm is implemented in the neural network to make a weight selection of the network and as an effort to improve the performance of the neural network. Experimental results in this paper show that the result of adding a random noise can outperform the result of using the simple feed-forward, especially for the testing accuracy. Moreover, implementing the genetic algorithm can improve the training accuracy and the testing accuracy and decrease the loss from the result of adding the random noise.

Keywords: Ionosphere · Classification · Feed-Forward Neural Network · Random Noise · Genetic Algorithm.

1 Introduction

There are many potential applications in the neural network that have been studied by researchers. One of those potential applications is signal processing. Boone, Sigillito, and Shaber [1] implemented the neural network for detecting specific nodules in radiological data. Moreover, the neural network can be used to distinguish between sonar signals which are from a similarly shaped rock and a mine [2]. Here, the neural network is experimented for solving a binary classification problem that is to discriminate whether radar returns from the ionosphere is good or bad.

The dataset that is used is ionosphere dataset which can be found in the University of California, Irvine website, particularly in the machine learning repository [3]. This dataset that was collected by a system in Goose Bay, Labrador consists of a phrased array of 16 high-frequency antennas. A total transmitted power on order is 6.4 kW, and an antenna obtains about 30 dBm at frequency ranges of 8 to 20 MHz. Good radar returns can show evidence of some structure type in the ionosphere while bad radar returns cannot. Also, the radar returns are used to analyze the ionosphere physics at the E- and F- layers (100 to 500-km altitude).

The ionosphere dataset has 351 instances, 34 features and two kinds of classes that are good and bad. All 34 features are represented as continuous numeric value and for good class is denoted with value 1 while bad class is denoted with value 0 [3]. Additionally, there are no missing values in this ionosphere dataset that to be the most substantial reason why this dataset is chosen. Since experiments that will be conducted here do not concentrate on how to handle missing values in the neural network, so avoiding to use missing values is necessarily essential.

Here, the neural network method that is implemented is a simple feed-forward neural network that uses backpropagation algorithm. Then, for trying to improve the average accuracy, the feed-forward neural network is modified by applying a random noise to its training data [4, 9, 10]. In this paper, the improvement attempt method is called as random noise feed-forward neural network. Besides that, the performance improvement effort is also done by applying a genetic algorithm to do a weight selection in neural network [5]. Experiments will be carried out using those methods, and the results will be compared based on their average accuracy.

2 Method

This section explains about dataset information that shows how to define the training data and the testing data. Furthermore, there are two kinds of the neural network methods that will be applied to experiments here that are a simple feed-forward neural network and random noise feed-forward neural network. There is an explanation about the implementation of genetic algorithm for selecting the weight of neural network. Also, there is an explanation about the evaluation method that will be implemented for assessing those the neural network methods performance.

2.1 Dataset

Since the used dataset which is ionosphere has uniform range [-1,1] for most features and binary range for few features and the targets, so that, no specific data pre-processing that is performed. From available 351 instances in the dataset, those instances are divided into two kind of sets that are training set in the amount of 75% and testing set in the amount of 25%. The distribution of the data is conducted by using random distribution.

2.2 Feed-Forward Neural Network

The feed-forward neural network is a network that involves an input layer of processing units (neurons), a hidden layer or intermediate layer and an output layer. For an illustration, a visual representation of the network is provided in the Figure 1.

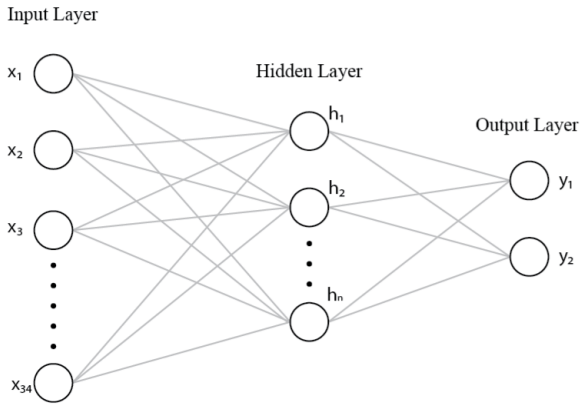

Fig. 1. Feed-forward neural network for ionosphere dataset

Based on the Figure 1, there are 34 neurons of input layer denoted as $\{x_1, x_2, x_3, ..., x_{34}\}$ which represent the input data or features to network and there are two neurons in the output layer denoted as $\{y_1, y_2\}$. The output layer will give out the predictions and use an activation function as the decision maker. Since the target of the network is binary which is either 1 or 0, sigmoid activation is used to map the input to values between 0 and 1. Regarding hidden layer, there is one hidden layer denoted as $\{h_1, h_2, ..., h_n\}$ in the network and experiments will be conducted using various number of neurons (n) in the hidden layer. Generally, the number of hidden layers is referred to the depth of the neural network and the deeper of the networks, the more complex functions that can be learned [7]. Since, the problem has a low complication, so that the number of layer of the used hidden layer is only one.

The training set is passed through the network, and the output generated from the network is compared with the real output. Using a function called cross-entropy whose value between 0 and 1, error or loss from the comparison result with the actual output is calculated. In the following step, that loss is used to modify the weights of the neurons so that the loss can be reduced gradually. Updating the weights is done by using an algorithm called backpropagation that applies gradient descent. Given the network and the loss function, back-propagation method calculates the gradient of the loss function associated with the neural network's weights. Gradient descent optimizer called stochastic gradient descent (SGD) is implemented for optimizing the finding process of the minimum point related to a certain weight. Additionally, a parameter called learning rate will determine the amount of the weights will be changed.

Therefore, in the implementation, the neural network process requires some parameter inputs. Those parameters that are fixed are the number of neurons in the input layer (34 neurons) and the number of neurons in the output layer (two neurons). While parameters that can be modified are the number of neurons in the hidden layer, the learning rate, and the number of epochs.

2.3 Random Noise Feed-Forward Neural Network

Random noise feed-forward neural network works in the same way with the simple feed-forward neural network in this paper. The only difference is that the method adds a random noise artificially to the neural network input data during the training process. Implementing this method causes the neural network model will be difficult to find a solution that fits precisely to the original training data and can decrease the neural network overfitting. Furthermore, there are many experimental results in previous works [4] which show that the random noise adding method can improve the neural network ability in generalization. Generalization indicates that a

trained the neural network can classify data from the same class as the training data that it has never learned before.

Using the random noise method, random noise is added to each training case in between training epochs. The amount of the random noise value is determined by a parameter called randomness. Moreover, the adding process will happen in every which iteration is defined by a parameter called epochForRandom. Based on the randomness and epochForRandom value, random noise is generated in the range of $\{-randomness, randomness\}$ and will be added to each input value in every epochForRandom iteration.

2.4 Genetic Algorithm Implementation in Neural Network

Genetic algorithm is an algorithm of search heuristic that implements the theory of natural biological evolution. This algorithm shows the process of natural selection to select the fittest individuals for the next generation process [6]. The general process of the genetic algorithm are initial population, fitness calculation, selection, crossover, and mutation. In neural network, the purpose of the implementation of the genetic algorithm is to attempt to optimize the performance of the neural network by selecting the weight of the network. In its implementation, the process starts by defining the population which comes from the neural network model. The chromosome in the genetic algorithm process represents the weight of the neural network. By determining a number of parents, the process of neural network forming is done as many as the number of parents. Here, the number of parents should be even so that the pair of parent selection that will be conducted is easier. From those neural network models, the fitness evaluation which is the loss function of each neural network in population is calculated. The fitness function determines how fit an individual is. Based on the fitness value, the population is sorted descendingly. Then, the next process is to pair two chromosomes based on their order in the population to be the first parent and the second parent. So that, the first parent is the best individual with the second-best individual based on their fitness function.

After the parent selection is done, the next process is crossover. Here, the crossover that is applied is uniform crossover by determining a crossover rate (P_c) as the chromosome ratio of the next generation from the parent. After the crossover process in each parent or chromosome pair, there will be two generated offsprings. Therefore, the number of the generated offsprings will be the same with the number of the parents. Afterwards, the next process is mutation for the offsprings. The mutation is carried out by replacing some genes of the offspring chromosome with a random number that ranges from -1 to 1. A mutation rate (P_m) should be also determined to define the portion of chromosome that will be replaced. After the mutation process is done, the fitness evaluation is performed again to find individuals from the population that are the best which the amount of the individuals is the same with the initial number of parents. Moreover, by defining a number of epochs, the genetic algorithm process will be iterated as many as the defined number of epochs.

After the iteration process is finished, the best offspring which represent the neural network model whose the best weight initialization is selected based on its fitness value. That neural network model then is used in the feed-forward neural network method implementation by using random noise.

2.5 Evaluation Method

Evaluation methods that are used for experiments in this the neural network process are accuracy and loss function. For the training process, the results are found by applying the accuracy method and the loss function while for the testing process, the results are generated by implementing the accuracy method. The accuracy and the loss are calculated averagely over a certain amount of epoch in ten times program running. The formula for the accuracy is provided below [11].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \tag{1}$$

where TP is true positive which is the outcome where the model predicts the positive class correctly, and TN is true negative which is the outcome where the model predicts the negative class correctly. Moreover, FP which is false positive is outcome where the model mispredicts the positive class, and FN which is false negative is the outcome where the model mispredicts the negative class. Regarding the ionosphere dataset, positive class means good while negative class means bad. Conclusively, accuracy is the division of the number of correct predictions and the number of total predictions multiplied by 100.

In addition, the loss function that is used to asses the training process is cross entropy loss since the target of the neural network is binary. The formula of the cross-entropy loss is provided below [12].

$$Cross\ entropy\ loss = -(t \log(y) + (1 - t) \log(1 - t) \tag{2}$$

where log is the natural log, y is the produced output of the neural network, and t is the target output.

Furthermore, since the accuracy that will be reported in this paper is the average result from ten times

running program, so a standard deviation has to be calculated. The standard deviation is used to know how much the amount of variation or dispersion of the values set. The formula of the standard deviation is provided below [13].

$$Standard\ deviation = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}} \tag{3}$$

where N is the length of the values set, x is the value and $\overline{x}$ is the average of the values set.

3 Results and Discussions

For doing the experiments, available pytorch library package is used to build the neural network and conduct the process [8]. In this section, there will be discussions about the result of feed-forward neural network, random noise feed-forward neural network, and the result of the implementation genetic algorithm in neural network. In addition, there is a comparison result between those methods and the result of a research that used the same dataset which is ionosphere [14].

3.1 Result of Feed-Forward Neural Network

Experiments of the feed-forward neural network are conduction using random distribution of the dataset. Moreover, using different parameters such as the number of hidden neurons, the number of the epoch and the learning rate is done as well for showing the effect to the evaluation results that are training accuracy, testing accuracy and the amount of the loss. There are three different values of parameters that are used. For the number of hidden neurons, there are 20, 40, and 80 hidden neurons. For the number of the epoch, there are 1000, 5000, and 10000. While the learning rates are 0.001, 0.01, and 0.1. All experiments are carried out as many as ten times program running. So that, all evaluation outcomes are the average of those program running outputs.

The accuracy average result of using random training and testing data is provided in Figure 2. For additional information, the standard deviation results is also provided in Table 1.

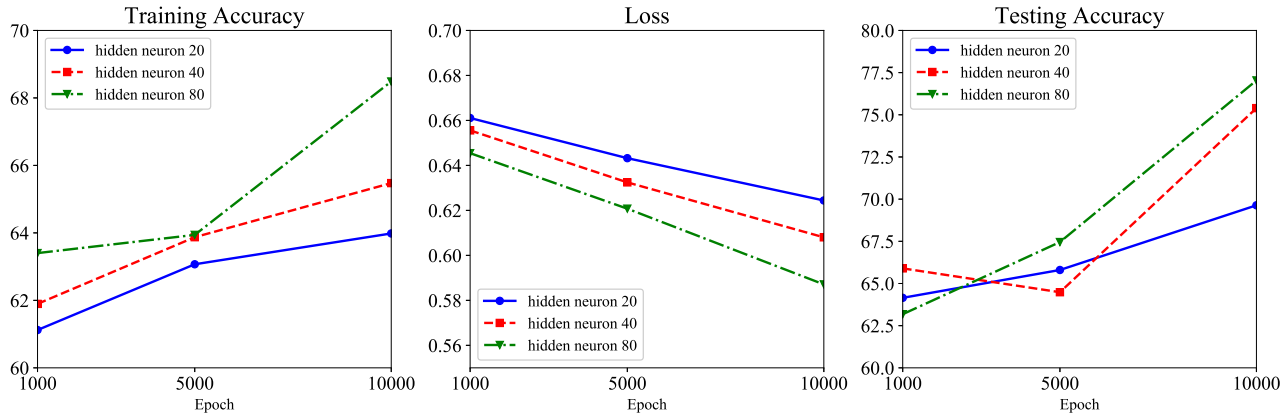

Fig. 2. Result of feed-forward NN using random training and testing data

Table 1. Standard deviation of using random training and testing data in feed-forward NN.

	Training Accuracy			*Testing Accuracy*		
Hidden neurons	Epoch 1000	Epoch 5000	Epoch 10000	Epoch 1000	Epoch 5000	Epoch 10000
20	3.76	2.05	1.10	4.53	4.66	4.01
40	1.34	1.57	1.17	2.67	5.23	5.71
80	1.32	1.21	1.18	2.59	5.42	4.61

Parameters that are applied differently are the number of epochs, the number of hidden neurons, and the learning rate. The result of using the diverse number of epochs and the different number of hidden neurons can be seen in Figure 2, that uses random training and testing dataset. Firstly, the first and the third graphs show that the higher of the number of the epoch, the higher of the training accuracy and the testing accuracy.

Moreover, the higher of the number of the epoch, the lower of the generated loss. Secondly, the higher of the number of hidden neurons, the higher of the training accuracy and the testing accuracy. Also, the higher of the number of hidden neurons, the generated loss will be lower. Therefore, using a higher number of the epoch and a higher number of hidden neurons, the training accuracy, and the testing accuracy will be higher as well leading the generated loss will be lower.

Furthermore, the result of using different learning rates with their standard deviation is provided in the Table 2. Since using a high number of the epoch and high number of hidden neurons show high accuracy, so that for using different learning rate experiments use 10000 epoch and 80 hidden neurons. The table shows that the higher of the learning rate that is used, the higher of the training accuracy and the testing accuracy while the generated loss will be lower. This means that using the higher of the learning rate value can reach the target faster and much precisely rather than using smaller learning rate in this the neural network model.

Table 2. Result of using different learning rate with 80 hidden neurons and 10000 epochs in feed-forward NN.

Evaluation type	Learning rate 0.001	Learning rate 0.01	Learning rate 0.1
Training accuracy	68.486 ± 1.18	86.744 ± 0.78	95.980 ± 0.64
Loss	0.587	0.328	0.115
Testing accuracy	77.036 ± 4.61	88.997 ± 2.91	91.043 ± 3.10

Conclusively, the best result of the feed-forward neural network implementation is the training accuracy 95.980 ± 0.64, the testing accuracy 91.043 ± 3.10 and the loss 0.115 by using 80 hidden neurons, 10000 epochs and learning rate 0.1

3.2 Result of Random Noise Feed-Forward Neural Network

Experiments of random noise feed-forward neural network are done using the same parameter values from the best result in using the simple feed-forward neural network. They are 80 hidden neurons, 10000 epochs, and the learning rate 0.1 by using the random distribution of the training data and testing data. Moreover, the experiments use different kind values of randomness and epochForRandom. The epochForRandom values are 50, 100, 200, and 400 while the randomness values are 0.01, 0.05, 0.1, 0.2, and 0.4. The evaluation results are training accuracy, loss, and testing accuracy that is produced in ten times program running respectively.

Firstly, using different values of epochForRandom results with the standard deviation is shown in Table 3. The results are done by using the same randomness value which is 0.1. The result shows that the higher of the epochForRandom, the higher of the testing accuracy. While for the loss and the training accuracy, the value pattern is not significantly changed as the higher of the epochForRandom. It indicates that the value of epochForRandom which represents on every which iteration the adding noise process happen can influence the result of the testing accuracy but not for the training accuracy and the loss.

Table 3. Result of using different epochForRandom with randomness 0.1 in random noise feed-forward NN.

Evaluation type	epochForRandom 50	epochForRandom 100	epochForRandom 200	epochForRandom 400
Training accuracy	95.430 ± 0.60	95.943± 0.47	95.764 ± 0.56	95.934 ± 0.48
Loss	0.112	0.106	0.112	0.112
Testing accuracy	89.632 ± 3.37	90.088 ± 2.80	91.627 ± 2.07	91.84 ± 2.85

Secondly, using different values of randomness outcomes can be seen in Figure 3. The results are experimented by using two different kinds of epochForRandom which are 200 and 400. The graph shows that the higher of the randomness value, the lower of the training accuracy. While for the loss is in the consistent range even though the randomness value is changed. Moreover, for the testing accuracy, the pattern value looks not consistent as along the randomness is changed. However, for both epochForRandom 200 and 400 show that the highest point of the testing accuracy is on applying randomness 0.1. In summary, the randomness value only influences the result of the training accuracy but not for the loss and the testing accuracy.

In conclusion, the best result of implementing random noise feed-forward neural network is on the training accuracy 95.934 ± 0.48, the loss 0.112 and the testing accuracy 91.84 ± 2.85. This best result uses randomness 0.1 and epochForRandom 400. Compared with the result of implementing the simple feed-forward neural network, the random noise implementation is proven that can improve the training accuracy and the testing accuracy and decrease the loss even though the difference value is very minimal.

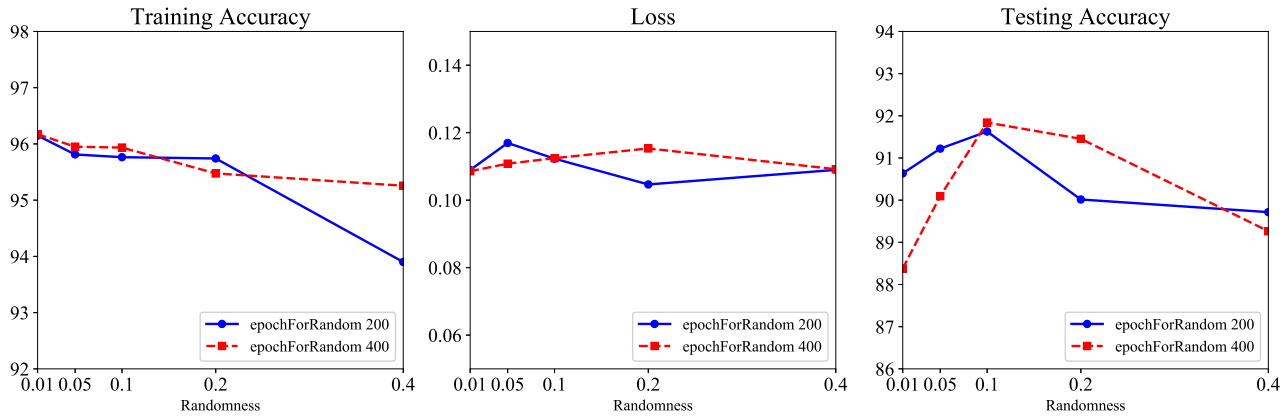

Fig. 3. Result of using different randomness values in random noise feed-forward NN

3.3 Result of Genetic Algorithm Implementation in Neural Network

Experiments of genetic algorithm implementation in neural network are done using the same parameter values from the best result in using the simple feed-forward neural network and in using random noise feed-forward neural network. The neural network parameters are 80 hidden neurons, 10000 epochs, and the learning rate 0.1 and the random noise parameters are randomness 0.1 and epochForRandom 400. Moreover, the experiments use different kind values of genetic algorithm parameters. The GAepoch values are 10 and 100, the number of parents are 6,10 and 14, the crossover rate (P_c) are 0.5, 0.7, and 0.9 and the mutation rate (P_m) are 0.01, 0.05, and 0.1. ISince using random noise can outperform the result of using simple feed-forward, these experiments are conducted by using the random noise. The evaluation results are training accuracy, loss, and testing accuracy that is produced in ten times program running respectively or in ten different random initialisations.

Firstly, using different values of GAepoch results with the standard deviation is shown in Table 4. For the initial experiment, the used number of parents is 6 parents and the P_c is 0.5 and the P_m is 0.01. The results show that the higher of the number of epochs used in the genetic algorithm process, the higher of the training accuracy and the testing accuracy. It indicates that using a high number of the epoch can improve the performance of the neural network.

Table 4. Result of using different GAepochs with 6 parents, P_c 0.5, and P_m 0.01.

Evaluation type	GAepoch 10	GAepoch 100
Training accuracy	97.596 ± 0.46	98.561 ± 0.30
Loss	0.073	0.049
Testing accuracy	89.182 ± 4.17	92.842 ± 3.70

Secondly, using different number of parents results with the standard deviation is shown in Table 5. Since the best result of using different GAepochs is using 100 epochs, so this experiment is done by using 100 epochs. The results show fluctuating pattern of the training accuracy and similar results of loss values. However, the best testing accuracy is showed by using ten parents.

Table 5. Result of using different number of parents with GAepoch 100, P_c 0.5, and P_m 0.01.

Evaluation type	Parents 6	Parents 10	Parents 14
Training accuracy	98.561 ± 0.30	98.493 ± 0.30	98.543 ± 0.25
Loss	0.049	0.051	0.049
Testing accuracy	92.842 ± 3.70	93.172 ± 2.05	90.154 ± 3.15

Thirdly, using different P_c values with the standard deviation is shown in Table 6. In this experiment, the used number of epochs is 100 and the number of parents is 10 as the best result of the previous experiments and the P_m value is 0.01. The results show that the higher of the P_c value, the lower of the testing accuracy. Moreover, the results of the training accuracy and the loss values show a fluctuating pattern as the P_c value increase. It indicates that using a small P_c value can improve the generalization of the neural network.

Table 6. Result of using different P_c values with GAepoch 100, 10 parents, and P_m 0.01.

Evaluation type	P_c 0.5	P_c 0.7	P_c 0.9
Training accuracy	98.493 ± 0.30	98.271 ± 0.18	98.486 ± 0.31
Loss	0.051	0.056	0.052
Testing accuracy	93.172 ± 2.05	92.729 ± 2.96	91.924 ± 2.86

Lastly, using different P_m values with the standard deviation is shown in Table 7. In this experiment, the used number of epochs is 100, the number of parents is 10, and the P_c value is 0.5 as the best result of the previous experiments. The results show that the higher of the P_m value, the lower of the training accuracy and the testing accuracy. It indicates that using small P_m value can improve the performance of the neural network based on their evaluation measurements.

Table 7. Result of using different P_m values with GAepoch 100, 10 parents, and P_c 0.5.

Evaluation type	P_m 0.01	P_m 0.05	P_m0.1
Training accuracy	98.493 ± 0.30	98.411 ± 0.19	98.394 ± 0.15
Loss	0.051	0.054	0.055
Testing accuracy	93.172 ± 2.05	92.333 ± 2.48	92.141 ± 3.10

In summary, the best result of implementing the genetic algorithm in the neural network is on the training accuracy 98.493 ± 0.30, the loss 0.051, and the testing accuracy 93.172 ± 2.05. This best result uses 100 epochs, 10 parents, P_c 0.5, and P_m 0.01. Compared with the result of implementing the random noise feed-forward neural network, the result of genetic algorithm implementation is proven that can improve the training accuracy and the testing accuracy and decrease the loss.

3.4 Comparison Results

This section discusses the comparison results between this paper and existing work that did experiments on the same dataset which is ionosphere but using different method [14]. The existing research worked by constructing a feedforward the neural network with a single hidden layer for pattern classification. The model starts with a small number of hidden neurons in the network and increases the number of the hidden neurons as required to improve the accuracy of the network. Moreover, for defining when to stop increasing the new hidden neurons, the method uses a subset of the given training data for doing cross-validation. So that, new hidden neurons are added to the neural network only when the accuracy of the training and the cross-validation are improved.

Based on the experiment result of [14], the training accuracy of using the ionosphere dataset is 99.20 ± 0.15 while for the testing accuracy is 89.52 ± 2.26. The result is produced by calculating the average of the accuracy in 10 tenfold cross-validation runs with the standard deviations as the range value of the data dispersion. Compared with the result of this paper, the best result of the testing accuracy outperforms the testing accuracy of the cross-validation method with 93.172 ± 2.05 and 89.52 ± 2.26 respectively. The difference of their training accuracy is also minimal which is around 0.71 even though the training accuracy in [14] produced a higher value. It indicates that implementing the genetic algorithm as a method for weight selection in the neural network can improve the performance of the neural network, especially for the generalization process.

4 Conclusion and Future Work

This paper has presented implementations of different methods which are a simple feed-forward neural network, random noise feed-forward neural network and genetic algorithm implementation in the neural network. In the implementation of using the simple feed-forward, using the higher number of hidden neurons, epochs and learning rate produce the better result rather than using lower values of them. Regarding the implementation of the random noise feed-forward neural network, the result shows that the testing accuracy is improved compared with using the simple feed-forward neural network. In addition, the usage of different parameters in the random noise model which are the randomness and epochForRandom is not too significant for the accuracy result. Furthermore, by using the genetic algorithm for weight selection in the neural network can improve the testing accuracy from using random noise feed-forward neural network.

There are many of future works that can be experimented for the next research. The first option is using bigger number data in the dataset with the higher number of the target class. It will be more challenging if the future research can do an experiment that is not binary classification. Moreover, using a dataset that has missing values will also be more interesting since the pre-processing data will be more complex to handle the

missing values. The second option is using a different kind of the neural network model, algorithm, activation function and weight optimizer will have different results. Last option is using a different kind of improvement methods to increase the accuracy. There will be much information that can be obtained by implementing a different type of methods for the neural network classification.

References

1. Boone, J. M., Sigillito, V. G., & Shaber, G. S. (1990). Signal detection capabilities of the neural networks: Radiology applications. Med. Phys, 17(2), 234-241.
2. Gorman, R. P., & Sejnowski, T. J. (1988). Analysis of hidden units in a layered network trained to classify sonar targets. the neural networks, 1(1), 75-89.
3. UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Ionosphere/. Last accessed 19 Apr 2018
4. Bustos, R. A., & Gedeon, T. D. (1995). Decrypting the neural network Data: A GIS Case Study. In Artificial Neural Nets and Genetic Algorithms (pp. 231-234). Springer, Vienna.
5. Montana, D. J., & Davis, L. (1989, August). Training Feedforward Neural Networks Using Genetic Algorithms. In IJCAI (Vol. 89, pp. 762-767).
6. Introduction to Genetic Algorithms, https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. Last accessed 26 May 2018
7. Davidian, D. (1995). U.S. Patent No. 5,438,646. Washington, DC: U.S. Patent and Trademark Office.
8. Pytorch Documentation, http://pytorch.org/docs/master/nn.html. Last accessed 19 Apr 2018
9. Zur, R. M., Jiang, Y., Pesce, L. L., & Drukker, K. (2009). Noise injection for training artificial the neural networks: A comparison with weight decay and early stopping. Medical Physics, 36(10), 4810?4818. http://doi.org/10.1118/1.3213517
10. Matsuoka, K. (1992). Noise injection into inputs in back-propagation learning. IEEE Transactions on Systems, Man, and Cybernetics, 22(3), 436-440.
11. Classification Accuracy and its Limitations, https://machinelearningmastery.com/confusion-matrix-machine-learning/. Last accessed 20 Apr 2018
12. Loss Functions, http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. Last accessed 20 Apr 2018
13. Calculating standard deviation step by step, https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/calculating-standard-deviation-step-by-step. Last accessed 27 Apr 2018
14. Setiono, R. (2001). Feedforward the neural network construction using cross validation. Neural Computation, 13(12), 2865-2877.