Genetic Algorithm-optimized Bidirectional Neural Network for Classification Task

Zhengtong Tan and Yang Yan

School of Computing in the College of Engineering and Computer Science
The Australian National University
Acton ACT 2601 Australia
{zhengtong.tan,yan.yang}@anu.edu.au

Abstract. This paper has proposed a new methodology that hybridizes Bidirectional Neural Network (BDNN) with Genetic Algorithm (GA) to develop a more robust classifier. We use a genetic algorithm-based feature selection method to optimize the initial network parameters and then fine-tune these parameters via Error back-propagation. This methodology takes advantage of global search ability from GA and local search capability from BP to obtain an optimal result. Our result shows that GA-optimized BDNN could achieve promising classification performance on two different data sets [3,4] with an accuracy of 92.50% and 100% respectively. We find this model has better generalization ability and faster convergence compared with a traditional BDNN. We believe our work provides a unique bio-inspired perspective and direction for deep learning research. Therefore, it may worth investigating the performance of this methodology on other bidirectional neural architectures.

Keywords: Bidirectional Neural Network · Genetic Algorithm · Backpropagation · Classification.

1 Introduction

Many modern neural networks have been proven to have excellent performance on classification problems. However, these neural network models could not consistently produce reasonable input unless a particular network is trained for this task. Neural networks that could produce plausible input values for given outputs have many applications. As a result, we are particularly curious if the model that produces fairly valuable input could extract more accurate rules. Therefore, Bidirectional Neural Network (BDNN) has been introduced to associate and learn the relationship between the input and output patterns for this task [9].

Moreover, parameter initialization is a vital design choice when developing a neural network model. Neural network models commonly use an optimization algorithm that fine-tunes the network weights. The initial weights and bias are the starting points in the search space of possible network weights. Typically, an ideal set of weights and biases could result in respectively better model performance

than that with random parameters. However, the problem is this set of parameters could not be learned until model training. To solve these problems, we apply Genetic algorithms to discover the best parameters for our BDNN model. In this paper, the Genetic algorithm will act as a global searcher to derive optimal initial weights and bias. It is performed where the BDNN parameters assigned as a genome in GA are learned and updated to fit the target fitness function. These retrained parameters will act as the initialized parameters in BDNN to enhance the convergence speed and generalization ability of our BDNN model.

This paper proposes a novel technique hybridizing the GA and BDNN that enable more superior starting points to improve model performance. This paper test this algorithm on two data sets for validity verification.

2 Related Work

2.1 Weight Initialization

Parameter initialization defines the initial values for the parameters in neural network models prior to training the models on a data set. Neural network models commonly use optimization algorithms such as stochastic gradient descent that fine-tune the network weights incrementally to reach the minimal loss value, hopefully producing a set of weights for patterns that make good predictions. This optimization algorithm needs to find a starting point in its search space of possible network weight values to begin its process. The weights initialization is the stage that assigns the network weights to specific values that define the starting points for the optimization algorithm. In general, the model initialized with a diverse set of weights would result in different starting points for the optimization process and finally lead to different performance characteristics [17].

However, finding a set of optimal initial weights is a difficult task. Randomization is commonly used, but it could still lead to poor network optimization once the random distribution is chosen improperly. Besides, though normalized Xavier weights initialization standardizes random numbers with a uniform probability distribution between a specific range, this initialization method relies on the assumption of the linearity feature of the activation function, and this method does not apply to all the activation functions. In conclusion, the weight initialization is essentially important for model performance, but weight initialization techniques need to be chosen carefully.

2.2 Genetic Algorithm

The genetic algorithm is a stochastic global search and optimization method developed by simulating the biological evolution mechanism in nature [16]. GA uses global information and does not need local gradient information, enabling GA to find near globally optimal solutions. GA uses evolutionary operations such as selection, crossover, and mutation as random search approaches for exploring the state space and exploiting the potential areas [15]. The Selection operator

will select the most adaptive parents based on their fitness values. Besides, the Crossover operation will determine solutions to preserved or deleted and combine good traits. In addition, the Mutation operation maintains diversity and explores new solutions. These evolutionary processes also bring a flexible balance between searching efficiency and adaption to different environments [8].

Genetic algorithms have been used to solve complex problems with the target functions that possess issues such as continuity, differentiability, time-complexity, etc. GA implements a survival of the fitness strategy in the search space for better solutions [15]. It is found that GA is very effective at feature selection and optimization problems [10]. Moreover, it has been proved that GA could be an effective approach for improving the accuracy of Neural networks through optimizing the initial weights of Artificial Neural Network [2].

2.3 Bidirectional Neural Network

Bidirectional associative memories (BAM) enable the model to store hetero-associative pattern pairs. In addition, counter-propagation networks have been developed to learn bidirectional mapping [7]. However, these methods have some limitations, such as limited storage capacity and unstable convergence [6]. In contrast, Bidirectional Neural Network is designed to associate and learn the relationship between the input and output. Moreover, BDNN can avoid these problems and considerably improve the model generalization ability even without using any other generalization techniques.

As for the classification problem, the original relation between input and output is not invertible. That is, different patterns may correspond to the same result. BDNN with associative memories requires a one-to-one relationship between input and output to achieve the training in the reverse direction. To investigate the interaction between input and output, especially the impact of output on input, a one-to-one mapping is introduced to mitigate this problem.

In general, a function is invertible if and only if each input has a corresponding unique out and each output is matched with exactly one input. This way, reverse mapping is still a function. For those output patterns that could not map to exclusive input patterns, the extra attributes should be aligned with them to make the function invertible, as shown in Table 1.

Table 1. Modified output pattern with extra node (simplified version)

Label	0	0	...	0	1	1	...	1
Extra Node	0.000000	0.005025	...	1.000000	0.000000	0.005025	...	1.00000

3 Method

3.1 Network Architecture

Inspired by the potential search capability of GA, the network architecture used in this study is composed of Genetic Algorithm (GA) and Bidirectional Neural Network (BDNN), which applies the effect and ability of GA for evolving the initial weights and biases of BDNN. There are two main stages in this architecture. The first stage is to harness GA to find an optimal set of initial weights and biases of BDNN. The second stage is to use the error back-propagation algorithm in both the forward and reverse directions to fine-tune the weights and biases [1] efficiently .

GA is a population-based search algorithm that requires initialization through a population of chromosomes. Consequently, a set of randomly generated weights is regarded as genes of chromosomes. We define the fitness function as the reciprocal of the weighted sum of Binary Cross-Entropy Loss (BCE) of classes, Mean Squared Error (MSE) of extra nodes, and an extremely small value c as shown in Equation (3) , where the purpose of c is to prevent the denominator of a fraction to be zero. BCE evaluates the error between the actual labels and the predicted classes in the binary classification task, and MSE assesses the gap between predicted values and actual values of extra nodes. We will investigate the effects of different ratios in the experiment part.

$$BCE(\hat{y}, y) = \sum_{i=1}^{n} -y_i log\hat{y}_i - (1 - y_i)log(1 - \hat{y}_i) \tag{1}$$

$$MSE = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \tag{2}$$

$$Fitness = \frac{1}{\alpha BCE + \beta MSE + \mathbf{c}} \tag{3}$$

During each iteration of the GA process, the initial populations of chromosomes will experience three evolutionary processes: selection, crossover, and mutation. We first evaluate the fitness of the individual corresponding to each chromosome. The top 10% of individuals will be selected from the population to form a mating pool for crossover operation following the principle that the higher the fitness, the greater the selection probability. The crossover process will choose two random chromosomes from the mating pool and swap genes from parents' chromosomes to produce offspring with better chromosomes. Every time a crossover occurs, there is a tiny probability of mutation, i.e., a random value from neither of its parents. This GA process will be continuously repeated until the maximum number of generations is reached, or optimal fitness is achieved.

3.2 Bidirectional Network Architecture

As for the second stage, we apply BDNN to optimize network weights and biases to bring the model close to a global optimum with the local search ability of the

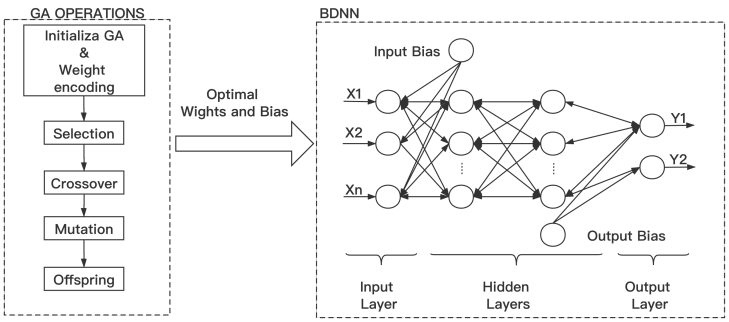

Fig. 1. Flowchart of GA-optimized training of BDNN [1]

BP algorithm. This study used the BP algorithm in both the forward and reverse directions to efficiently fine-tune the weight matrix. We use two hidden layers with n hidden units. In addition, we design independent input bias and output bias for forward and reverse direction. In contrast, the rest of the weights and bias on the connections are shared in these two directions as shown in Fig. 1 [11].

4 Experiment

4.1 Dataset and Data-preprocessing

This paper uses two different datasets. First, we use the Anger Dataset [3]. This dataset contains 400 samples generated by an experiment that collects participants' pupil responses to certain types of anger stimuli for detecting the authenticity of anger. There are 6 features include the mean of pupil size, the standard deviance of pupil diameter, degree of pupil dilation, 1 st and 2 nd components of Principal Component Analysis on images. In addition, each sample is labeled with "Genuine" or "Posed". We will attempt to classify the actual anger through these features.

Second, we use banknote authentication Data Set [4]. The data is extracted from authentic banknotes and fake banknote-like specimens images. Wavelet Transform is used to extract features from images. This data set contains 1372 items, including 4 features. They are the variance, skewness, kurtosis, entropy of the image. Each of them is labelled as 0 (authentic) or 1 (forgery). This study tries to detect the veracity of banknotes.

We applied unity-based normalization to the input feature and added extra nodes for a one-to-one mapping between input and output. The data set is split into a training set (80%), validation set (10%) and test set (10%).

4.2 Experiment with Genetic Algorithm Hyper-parameters

The fitness function of GA is calculated by the reciprocal of the sum of BCE and MSE. In addition, we attempt different proportions of losses in the fitness

function to get the best measurement for weighing the merits of the solution since the importance of BCE and MSE may differentiate.

The GA hyper-parameters, such as the number of generations, percentage of mutation, selection methods, crossover probability, etc., are selected by grid search. The grid search algorithm is commonly used for hyper-parameter tuning in machine learning models, which tries some combinations of hyper-parameters, evaluates the model results and scoring metric. Therefore, the combination of hyper-parameters that gives the best model performance will be considered as the hyper-parameter setting.

According to the result of the grid search algorithm, this paper studies a Genetic Algorithm with an initial population of BDNN parameters, steady-state selection, single-point crossover with a probability of crossover 0.8, random mutation with 8 solutions selected in a mating pool, and the maximum number of 300 generations.

4.3 Experiment with Network Backbone

The models in this paper include BDNN and GA-optimized BDNN. The learning rate is set to be 2e-2, and an Adam optimizer is used to reduce oscillation during the training procedure. The number of epoch is set to be 3000 and 300 respectively for two different datasets. We adopt L2 regularization with a minor weight decay of 1e-5 to prevent over-fitting. We use MSE for extra nodes and BCE for labels for the forward direction while applying MSE for input for the reverse direction. We use the Sigmoid activation function in the output layer to calculate each output class's normalized probability.

4.4 Experiments with Weight Initialization Methods Comparison

We compare among weight initialization methods, including zero weight initialization, random weight initialization, Xavier weight initialization [13], GA-optimized weight initialization with recommended hyper-parameter setting employed by Gad et al [5], and GA-optimized weight initialization with grid search discussed in this paper. We evaluate different initialization methods on convergence speed and generalization capability.

5 Result and Discussion

5.1 Loss Ratio and Weight Initialization

Table 2 shows the results from different sets of loss ratios. The result illustrates that the various loss ratios will result in diverse model performance. The model achieves the highest accuracy and the most robust generalization ability when α is 0.1 and β is 0.9.

As for the experiment results of different network parameters initialization methods, the first evaluation, model generalization, is measured by the accuracy

Table 2. Comparison among different losses ratios

BCE α	MSE β	Training Accuracy	Validation Accuracy	Testing Accuracy
0.5	0.5	94.38	91.67	85.00
0.1	**0.9**	**99.69**	**94.44**	**92.50**
0.3	0.7	95.94	88.48	91.25
0.7	0.3	96.88	83.17	87.50
0.9	0.1	97.71	94.40	87.50

of the test set. The results are in Table 3. ZEROWI means the zero weights initialization, while GAOPRHS means GA-optimized weight initialization with recommended hyperparameter setting [5]. In addition, GAOPGS means GA-optimized weight initialization with optimal hyperparameters via grid search.

Table 3. Result of accuracy on test set

Data set	Weight initialization	Test accuracy
Anger Dataset	ZEROWI	82.50%
	RANDOMWI	76.25%
	XAVIERWI [13]	77.85%
	GAOPRHS [5]	86.35%
	GAOPGS	91.25%
Banknote authentication Dataset	ZEROWI	94.50%
	RANDOMWI	98.01%
	XAVIERWI [13]	93.38%
	GAOPRHS [5]	100.00%
	GAOPGS	100.00%

The results of convergence speed are shown in Fig. 2. The GAOPGS experiences a smooth and stable increase during the training procedure with less oscillation than the rest of the initialization methods. Note that model with zero weight initialization learns slowly in the early stage, and model with random initialization experiences noticeable vibration at the early phase. The obtained results indicate that GAOPGS has better convergence performance over other weight initialization methods.

5.2 Comparison with Benchmark model

To evaluate the performance of our model, we take traditional bidirectional neural network as the baseline model, which has the same network architecture as GA-optimized BDNN and is trained by a similar training methodology. The results of training performance are shown in Fig. 3. As is demonstrated, the convergence speed of GA-optimized BDNN offer a significant advantage over the BDNN.

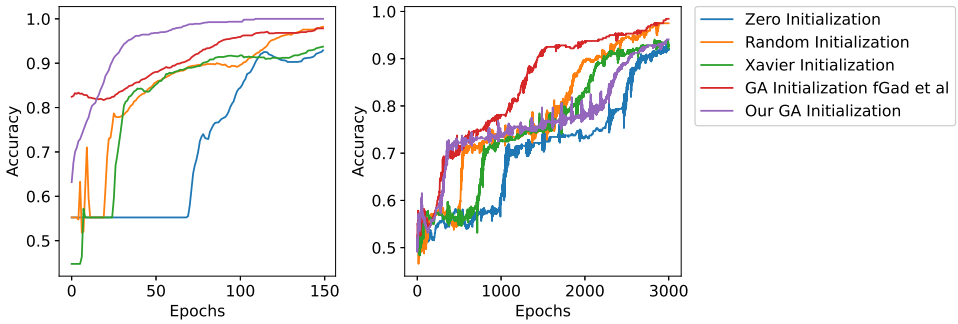

Fig. 2. Model performance comparison among weight initialization methods

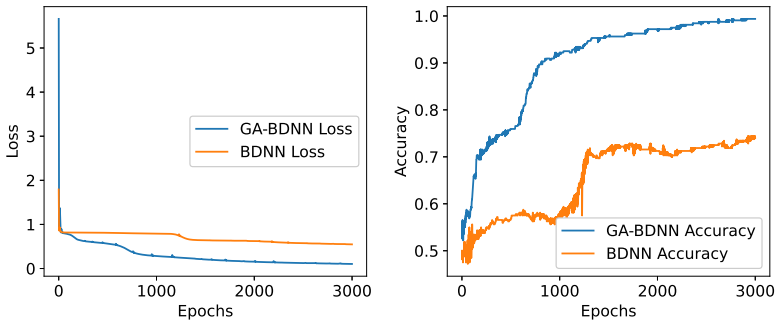

Fig. 3. Model performance comparison of BDNN and GA-optimized BDNN

5.3 Performance of the Classification Models

The accuracy $= \frac{TP+TN}{TP+FP+TN+FN}$ is an important criterion when evaluating the classification task.Besides, we also use recall $= \frac{TP}{TP+FN}$, precision $= \frac{TP}{TP+FP}$, and F1 score$= \frac{2*Precision*Recall}{Precision+Recall}$as additional measurement criterion for these classification tasks. We have implemented a 10-folds cross-validation technique for the model generalization test. The advantages of this train-test-split approach are measuring the variance of performance across the different data sets and detecting model instability [12]. The results of classification performance on anger dataset which are the average values of 10 runs are showed in Table 4, the GA-optimized BDNN classification model performs well on genuine Anger detection task, with test accuracy of 92.50%. In addition, the model achieve 92.50% recall and 92.50% precision, which means a balanced result. On the other hand, GA-optimized BDNN also has outperformance of the rest models on the banknote authentication task. The testing result indicates that the GA-optimized BDNN model could overcome the over-fitting problem and have a satisfying generalization ability for adapting to new data.

Table 4. Accuracy, precision, recall and F1 score of classification models

Metric	BDNN	Gad, el al [5]	GA-optimized BDNN
Testing accuracy	81.25	91.25	92.50
Precision	86.11	92.30	92.50
Recall	77.50	90	92.50
F1 score	81.58	91.14	92.50

5.4 Discussion

In conclusion, the genetic algorithm could help improve the model performance of BDNN by systematically tuning the parameter initialization. On the one hand, hybridizing BDNN with GA brings power to derive an optimal set of initial weights that enhances the efficiency of our BDNN model. On the other hand, it infers that the inherent defects of the BP algorithm, such as local minimum and slow convergence speed, could be mitigated by the global search capability of genetic algorithms.

6 Conclusion and Future Work

This paper has presented a methodology that applies a Bidirectional Neural Network with a Genetic Algorithm to develop a more powerful and robust classifier. This methodology takes advantage of global search ability from GA and local search capability from the error back-propagation algorithm to get an optimal result. Compared with the BDNN model, this approach has a faster convergence speed and more consistent predictions on unobserved data, which show the and validity and robustness of harnessing these two bio-inspired methodologies.

In our work, we use GA to find an optimal set of initial parameters of BDNN and fine-tune the weights and biases via an error back-propagation algorithm. One problem is that the GA is only applied in weight initialization and is not fully involved in optimizing the parameters. Though the optimal parameters have a significant improvement on the model effect, it is worth investigating the performance of the weight cross-optimization with both GA and BP that allows GA to participate in parameter initialization and neural network process training. In this way, the global search ability of GA is expected to help the model jump out of local minima.

As for the network architectures, this study implements a fully connected bidirectional neural network, which lacks flexibility and produces redundancy. As a result, we will also explore other neural network structures with directional features such as bidirectional recurrent neural network [14].

References

1. AGRAWAL, V., CHANDWANI, V., NAGAR, R., SINGH, S.: Hybrid artificial neural networks aided conceptual stage design of industrial roof trusses

2. Chang, Y.T., Lin, J., Shieh, J.S., Abbod, M.F.: Optimization the initial weights of artificial neural networks via genetic algorithm applied to hip bone fracture prediction. Advances in Fuzzy Systems **2012** (2012)
3. Chen, L., Gedeon, T., Hossain, M.Z., Caldwell, S.: Are you really angry? detecting emotion veracity as a proposed tool for interaction. In: Proceedings of the 29th Australian Conference on Computer-Human Interaction. pp. 412–416 (2017)
4. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
5. Gad, A.F.: Pygad: An intuitive genetic algorithm python library. arXiv preprint arXiv:2106.06158 (2021)
6. Hecht-Nielsen, R.: Counterpropagation networks. Applied optics **26**(23), 4979–4984 (1987)
7. Jabr, N.A.A., Kareem, E.A.: Modify bidirectional associative memory (mbam). International Journal of Modern Trends in Engineering and Research (IJMTER) **2**(08), 136–151 (2015)
8. Katoch, S., Chauhan, S.S., Kumar, V.: A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications pp. 1–36 (2020)
9. Kosko, B.: Bidirectional associative memories. IEEE Transactions on Systems, Man, and Cybernetics **18**(1), 49–60 (1988). https://doi.org/10.1109/21.87054
10. Li, Y., Chen, Y.: Genetic ant algorithm for continuous function optimization and its matlab implementation. In: 2010 International Conference on Intelligent System Design and Engineering Application. vol. 1, pp. 791–794. IEEE (2010)
11. Nejad, A.F., Gedeon, T.D.: Bidirectional neural networks and class prototypes. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 3, pp. 1322–1327. IEEE (1995)
12. Lopez-del Rio, A., Nonell-Canals, A., Vidal, D., Perera-Lluna, A.: Evaluation of cross-validation strategies in sequence-based binding prediction using deep learning. Journal of chemical information and modeling **59**(4), 1645–1657 (2019)
13. Sai, T.A., Lee, H.h.: Weight initialization on neural network for neuro pid controller-case study. In: 2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT). pp. 1–4. IEEE (2018)
14. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing **45**(11), 2673–2681 (1997)
15. Shen, Z.Q., Kong, F.S.: Optimizing weights by genetic algorithm for neural network ensemble. In: Yin, F.L., Wang, J., Guo, C. (eds.) Advances in Neural Networks – ISNN 2004. pp. 323–331. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
16. Yu, C.C., Liu, B.D.: A backpropagation algorithm with adaptive learning rate and momentum coefficient. vol. 2, pp. 1218 – 1223 (02 2002). https://doi.org/10.1109/IJCNN.2002.1007668
17. Zhou, Q., Ooka, R.: Performance of neural network for indoor airflow prediction: Sensitivity towards weight initialization. Energy and Buildings **246**, 111106 (2021). https://doi.org/https://doi.org/10.1016/j.enbuild.2021.111106, https://www.sciencedirect.com/science/article/pii/S037877882100390X