Abstract.
In recent years, neural network has achieved some remarkable result in numerous fields as the computational power is continuously raising. However, the hyper-parameter of the neural network, namely the number of hidden neurons, learning rate and choices of optimization functions has always been confounding to tune for different networks. In this paper, I extend the previous work of bidirectional network and propose using a Hybrid of Random Immigrants Genetic Algorithm (HIGA) to adjust the hyper-parameters. The results showed the HIGA does not improve the performance of the neural network compare to the previous manually tuned network, but it still has its usefulness for the automation of the tuning task and there are still potential improvements on the algorithm that have yet to test.

Keywords: Genetic Algorithm, Neural Network, Bi-directional Neural Network, HIGA, Elitism-based Immigrants, Random Immigrants

1 Introduction

This paper is an extension from the previous work, the same Bidirectional Neural Network (BDNN) with the SFEW dataset that I have spited the PHOG and LPQ values of the dataset into ten features as inputs to the BDNN are still used from the previous work. Although in the previous result, the BDNN is performing better than the normal Multiplayer Perceptron Network, but the testing accuracy is still significantly lower than the accuracy claimed by the dataset paper.

My goal is to try and increase the testing accuracy with applying genetic algorithm (GA). GA can be applied in different aspects of a neural network, such as in the whole weight matrix of the BDNN, the architecture or the hyper-parameters etcetera. Calculating weight using GA theoretically will result in a slower convergence than back-propagation using gradient descent since GA is a stochastic search, applying GA on the architecture of BDNN is very interesting, but hyper-parameters are selected for this paper to be tuned by GA to improve upon the previous testing accuracy, because the structure of a neural network can be designed through architectural analysis based on mathematical foundations by human, hyper-parameters on the other hand, the typical methodology for tuning is usually trial and error which is more suitable for automation.

Due to the nature of neural network, the weights of the BDNN or the weights of any neural network are continuously changing and updating to find the optimum solution, thus applying GA in this dynamic environment where change may occur over time can be challenging. Traditional GAs often focused more on selecting fitter individuals based on the fitness scores, then recombining them using genetical operations, and often quickly locate the optimum solution. These traditional approaches often lose

most of their diversities after certain number of generations and does not track or adapt to the dynamic problems well. **[HIGA Reference]** To cope with the dynamic characteristics of the neural network, HIGA technique proposed by Shenxiang Yang is used to address this problem by keep certain level of diversity to maintain the adaptability.

The rest of the paper is going to first outline HIGA and explain the approaches used in implementing HIGA in Section 2. Section 3 presents the implementation results and discussions on the results. Section 4 concludes the paper and discusses future work.

Note in order to accommodate the genetic algorithm, the whole code of BDNN from previous work is refactored but the fundamental concept and network structure are remained the same

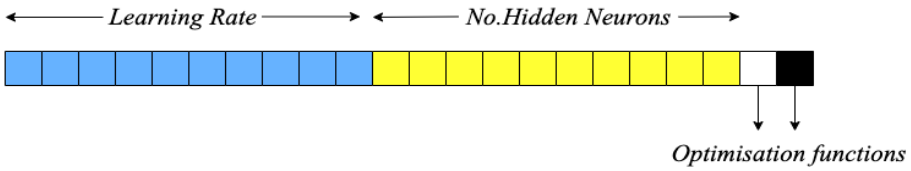
2 Method

2.1 HIGA

HIGA is a hybrid algorithm of random immigrants and Elite-based immigrants, the core idea is to memorise $rei \times n$ elites from the previous generation *G(t-1)*, where *rei* is the ratio of elites and n is the size of the population. After selecting and applying genetic operators similar to the normal GA, HIGA replaces $(rei + rri) \times n$ population with lowest fitness score with: $rei \times n$ mutated elites from *G(t-1)* with separate mutation rate m_e, and $rri \times n$ randomly generated population, where *rri* is the ratio of random immigrants. In this experiment, both *rei* and *rri* is set to 0.1, and m_e is set to 0.001.

2.2 Encoding

For this experiment, there are four hyper-parameters need to be optimised (4 genes): *Learning Rate*, *No.Hidden Neurons* and two *Optimisation functions* (one for forward pass and one for backward pass of BDNN). These hyper-parameters are combined into a single chromosome using binary and representation as the theory of genetic algorithm is mainly focused on binary representation and barely on non-binary representations **[float reference]**. The chromosome is of length 22, where the first 10 digits are the encoding for *Learning rate*, followed by 10 digits of *No.Hidden Neurons* encoding. The last two digits are the optimisation functions encoding, where $optimisation\ function \in \{Adam,\ Stochastic\ Gradient\ Descent\}$, hence one digit is used for each optimisation function. The same encoding scheme is applied for all the four genes, that is using fixed-point integer encoding.



2.3 Genetic operators

Cross over and mutations are both applied as genetic operators for this experiment. Cross over is applied two times, the first crossover happens at cross rate of p_c using uniform crossover where the two crossing chromosomes are split to three sections: *Learning rate*, *No.Hidden Neurons* and *Optimisation functions* and only the selected crossing sections under p_b of uniform crossover is crossing for the second time using one-point crossing over. Bit flip mutation is applied for the mutation operator, and mutation happens at mutation rate of P_m for each bit of the 22 bits chromosomes. p_c is set to 0.8 and p_m is set to 0.001 for this experiment.

2.4 Fitness Function and Selection

Fitness of this experiment is determined by the testing accuracy of the BDNN using the selected hyper-parameters from HIGA, and the selection uses proportional selection technique. which the fitter individuals will have higher probability to be selected and survive to the next generation. This approach does keep certain level diversity of the population as the less fit individuals still have chance to be selected, but the diversity will start to fade out by the high pressure of selection with probability theory. However. The proportional selection technique is combined with HIGA, HIGA keeps generating *rri%* random population to keep the diversity, moreover, HIGA also brings *rei%* previous mutated elites to speed up the convergence.

3 Result and Discussion

Larger population size and a greater number of generations could possibly yield better results, but due to the lack of computational power, the experiment is tested on the population of 20 with 15 generations. On the 15 generation, the most fitted DNA was translated to 0.115 learning rate, 84 hidden neurons and use Adam as forward optimiser, SGD as backward optimiser, it reached 33.82% accuracy. I used the same BDNN to train the same parameters from the most fitted DNA, and the result is recorded in Table 1. The average accuracy of the 5 trials is 23.05% which is 0.03% less than the average accuracy from the previous work with hyper-parameters manually tuned.

Table 1.

Trial No.	Accuracy
1	21.26
2	22.56
3	25.36
4	20.87
5	25.20

The result does not improve the average accuracy from the previous work, however, the testing accuracies throughout the generations have no sharp changes, the testing accuracy at the beginning of generation 0, that is randomly generated parameters also has very similar average testing accuracy compare to generation 15. This can indicate the search space for hyper-parameters of the BDNN is very limited, the hyper parameters of a neural network determines how fast the neural network converges, there is a max and a min threshold for the network to be overfit or underfit, and the search space is to find the optimum value in between the two thresholds. Moreover, the effects of the hyper-parameters are limited, the optimum and sub optimum may not differ by a large scale.

Although HIGA does not improve the average performance in accuracy, but HIGA does help in exploring the optimum values under the dynamic environment of the neural network, it extended the range that the maximum accuracy of the neural network can achieve, and automated the process of manual tuning the hyper parameters. Calculating the hyper-parameters of neural network will require relatively large computational power in relation to the size of the network and the dataset, and can be very time consuming, which may not be very realistic to tune using the genetic algorithm, but using GA for few generations can provide a good starting point to tune.

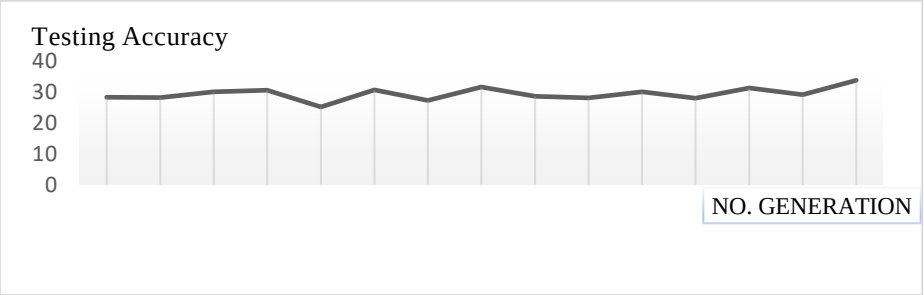


Fig. 1 Testing Accuracy of each Generation.

4 Conclusion and Future work

With HIGA approach, the result in the end does not find much improvements on the average accuracy but it does discover a higher upper bound a neural network can

achieve. While the improvements on the neural network but adjusting the hyper-parameters are limited, the automation for the tuning process provides beneficial value. In the future, the encoding of the chromosomes can be switched to gray coding to reduce the hamming distance that slows down the convergence [1], compare the effects and performance of different genetic algorithm on hyper-parameters tuning of BDNN, and compare the difference on applying HIGA on the hyper-parameters of MLP and BDNN to see whether HIGA will perform better in tuning the hyper-parameters for MLP.

5 Reference

Budin, L., Golub, M., & Budin,A 2010. Traditional techniques of genetic algorithms applied to floating -point chromosome representations [1]

Montana D.J., & Davis, L. 1989. Training Feedforward Neural Networks Using Genetic Algorithm [2]

Yang, S., 2007. Genetic Algorithm With Elitism-Based Immigrants For Changing Optimization Problems. [3]