ISSN 2278-3091
Volume 8, No.1.6, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse3681.62019.pdf
https://doi.org/10.30534/ijatcse/2019/3681.62019

GA-Deep Neural Network Optimization for Image Classification

Andrew Ngi Ing Hui[1], Aqilah Baseri Huddin[*1,2], Mohd Faisal Ibrahim[1,2], Fazida Hanim Hashim[1,2], Salina Abdul Samad[1,2]
[1]Department of Electrical, Electronic and Systems Engineering,
[2]Centre for Integrated Systems Engineering and Advanced Technologies (Integra),
Universiti Kebangsaan Malaysia,
UKM Bangi, 43600, Selangor, Malaysia.
*Corresponding author: aqilah@ukm.edu.my

ABSTRACT

Deep neural network (DNN) has been used to solve many pattern recognition tasks specifically for classification of images, sounds and texts. This is due to the ability of DNN model to extract high level representation of features. However, a deep neural network is built up from a set of hyperparameters that need to be tuned in order to obtain the highest performance and at lower computational time. In this paper, the hyperparameters that are tuned is the combination of number of hidden layers and the number of neurons in each layer. Nevertheless, the challenge arises when choosing the suitable tuning method for this model. The aim of this study is to evaluate and compare the tuning methods of a conventional grid search (GS) method with a population-based searching method, known as genetic algorithm (GA). The comparison is made based on the performance of DNN model for classifying MNIST handwritten digits, in terms of the classification accuracy and the time taken to complete the task. The MNIST handwritten dataset is divided into 3 sets, 54000 images in training set, 6000 images in validation set and 10000 images in testing set. The results show that GA and GS methods achieved a comparable classification accuracy of 98.23% and 98.27%, respectively. However, GA method took only half of the time to search for the optimized combination, when compared to GS method, which is only 4.19 hours compared to 8.59 hours, for the same search space area.

Key words: Deep Neural Network, Genetic Algorithm, Grid Search, Hyperparameters optimization

1. INTRODUCTION

Nowadays, artificial intelligence technology has been widely adopted and used in various areas. Machine learning is one of an important part in artificial intelligence technology as it able to perform any specific task by learning the pattern and inferences automatically. Neural network is one of the most popular algorithms for machine learning used in various pattern recognition tasks such as in medical application [1], agriculture [2], power distribution management [3], earth disaster prediction [4], and face recognition [5].

Research supported by The Ministry of Education Malaysia under the grant number FRGS/1/2017/TK04/UKM/02/4.

Early neural network architecture was introduced as early as in 1940s [6]. The idea of mimicking the neurons activity in the neural network architecture was ever since been explored and expanded. In the 1980s, a learning supervised algorithm of back-propagation in a neural network has sparked excitement among researches [7]. A back-propagation algorithm allows learning nonlinear features from multiple hidden layers in a neural network. However, the conventional neural network failed when there are more than 3 hidden layers in the network [8], [9].

A recent algorithm for machine learning, deep learning is an extension of conventional NN, where the models are more in-depth and has become a center of attraction [10], [11]. Deep learning network is ought to be powerful because of its ability to extract high representation features contributed by vast number of hidden layers in the architecture [12]. It is important to a build a deep learning neural network model that can achieve its maximum effectiveness. However, it is not easy to build one as the model contains a set hyperparameters that need to be carefully chosen and tuned [13]. Hyperparameters are the parameters' values set prior of the learning proses. For an example, for a deep neural network model, hyperparameters that need be tuned includes the activation function, loss function, number of neurons for each hidden layer, number of hidden layers, learning rate and batch size. The more hyperparameters to be tuned, the more time is needed to search for an optimized hyperparameters. Although it was said that tuning the model is more of an art than a science, but carefully tuned hyperparameters can increase its accuracy and reduce the computational cost. Hence it is important to use an efficient method in finding the optimized model.
Various methods have been performed in tuning the hyperparameters for an optimized model. The currently found techniques are exhaustive methods and model-specific methods [14]. Exhaustive methods search the hyperparameter space exhaustively. Hence, this kind of method is computationally expensive. This simplest and most widely used method is grid search [13]. Another method is using random search where the hyperparameters use the probability

distribution instead of sampling on a grid. However, exhaustive method only works well in cases of low dimensionality [15][16].

Another method is model-specific methods that optimize hyperparameters for a specific model choice. The work is more suitable to be applied to a small sample size. The downside of this method is that only certain model class is suitable and therefore it cannot be applied in general. It has to adapt to the best configuration hyperparameters which has been found today [17]. Besides, this technique does not produce the desired output [16].

A method has to be implemented in order to reduce the computational cost of finding the optimized hyperparameters with larger search space and searching in a more intelligent manner. Genetic algorithm can be used to search for the optimized hyperparameters in such manner. Genetic algorithm is a method for solving optimization problems based on natural selection, which is the process that drives biological evolution [18]–[20]. The algorithm repeatedly modifies population of individual solutions over generations.

This study focuses on the performance of genetic algorithm in finding the optimized combination of hyperparameters in image classification. The performance is compared with the conventional optimizing method, that is grid search method. Evaluation on the classification's accuracy and time taken for the search of an optimized model using GA and grid search methods are analyzed.

This paper presents the best optimized tuning method to build a deep neural network, specifically for MNIST handwritten classification by comparing a simple conventional method, i.e. grid search with a population based-method, i.e. genetic algorithm method. The accuracy of the classifier and the time taken are the measures used for comparison.

In next section, the methodology of the proposed MNIST optimized-classifier approach are discussed. Section 3 discusses the results and discussions and the findings are concluded in Section 4.

2. RESEARCH METHODOLOGY

In this study, the model is used to classify the images of handwritten digits. These images are obtained from the Modified National Institute of Standard and Technology (MNIST) database, that is available online, where it contains 60000 training set and 10000 testing set of data [21]. Each sample image is in grayscale and has the size of 28 x 28 pixels. Images from MNIST are separated to two sets of data: training set and testing set. The images data from MNIST database are in 8-bit unsigned integer format and the grayscale of each image falls in range of 0 to 255. Value 0 indicates the background and 255 indicated the foreground.

The model that is being used in this study is a multilayer perceptron or a feedforward neural network. A feedforward neural network contains minimum of 3 layers; one input layer, one hidden layer and one output layer as shown in Figure 1.

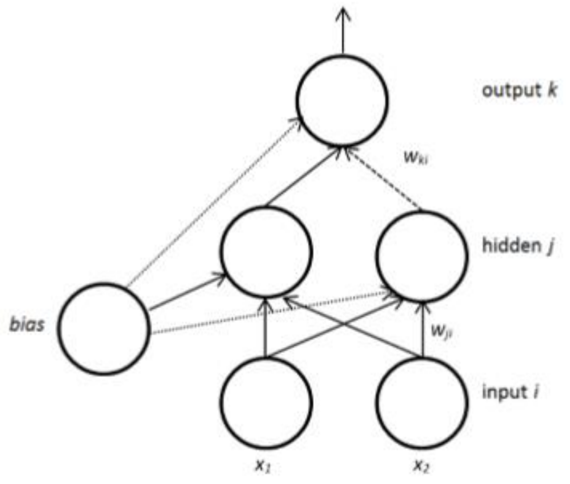

Figure 1: Multilayer perceptron neural network consists of input layer, hidden layer and output layer.

However, the hidden layer may be expanded to have more than one layer. The neurons in the input layer is directly connected to the input images. Whereas, the neurons in the output layer represents the classes, k of a layer, l is defined as

$$x_k^l = \sigma^{l-1}(b_{l,k}^{l-1} + \sum_{i=1}^{n} w_{l,k}^{l-1}\, x_l^{l-1}) \qquad (1)$$

where σ^{l-1} is the activation function which in this case is rectified linear unit (ReLu), $b_{l,k}^{l-1}$ is bias applied, $w_{l,k}^{l-1}x_l^{l-1}$ is the product of weight and input of previous neuron. For the output layer, Softmax function will be used in multi-classification task [22]. Adam optimization will be used to update the weights of the network in training data.

The model of neural network is built with Python version 3.5 with Keras library that will be run using Tensorflow as backend. The graphic card used is NVDIA Geforce 840M. ReLU is used due to its higher efficiency [23]. It is able to reduce the likelihood of vanishing gradient problem. The loss function will be using the categorical cross-entropy which it returns the cross-entropy between an approximating distribution and a true distribution. This function is normally used in "one-hot" encoding data.

In this study, the hyperparameters in the deep neural network to be tuned are the number of hidden layers, N_l and number of neurons, N_n. N_l will be set from 1 layer to 10 layers; whereas N_n will be set from 50 neurons to 1000 neurons with increment of 50 neurons. The combination of number of hidden layers and hidden nodes is searched using GA and GS methods.

Genetic algorithm or GA is an optimization technique inspired by the evolutionary process in biological cells. The principal of GA is based on Darwinian's theory, where the selected population is the survival of the fittest [24], [25]. The processes in GA are illustrated as in Figure 2.

The process starts with generating random population of chromosomes. Then, the fitness of each chromosome in the population is evaluated using fitness function, which in this work is the accuracy of the classification.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = true positive; TN = true negative; FP = false positive and FN = false negative.

Then, the chromosomes to be selected as parent to the next population is made based on Roulette wheel selection. Two operators are used in GA, crossover and mutation.

In this work, the number of generation set is to 5. On each generation, the population size is set with 15 chromosomes. All the population in the particular generation are trained and scored. At least 40% of the highest score population will be retained; the lower score population has 10% chance to be chosen. If there is lacking population number, crossover process will occur where the new generated population will inherit the characteristic of chosen population at the previous step. All the generated population will then undergo 20% mutation rate. This is to prevent the population from overfitting. The whole process is repeated until the last generation and the best hyperparameters can be determined. Further study is done by increasing the number of hidden layers to 50 and 1000 neurons for each hidden layer over 100 generations with 30 populations each.

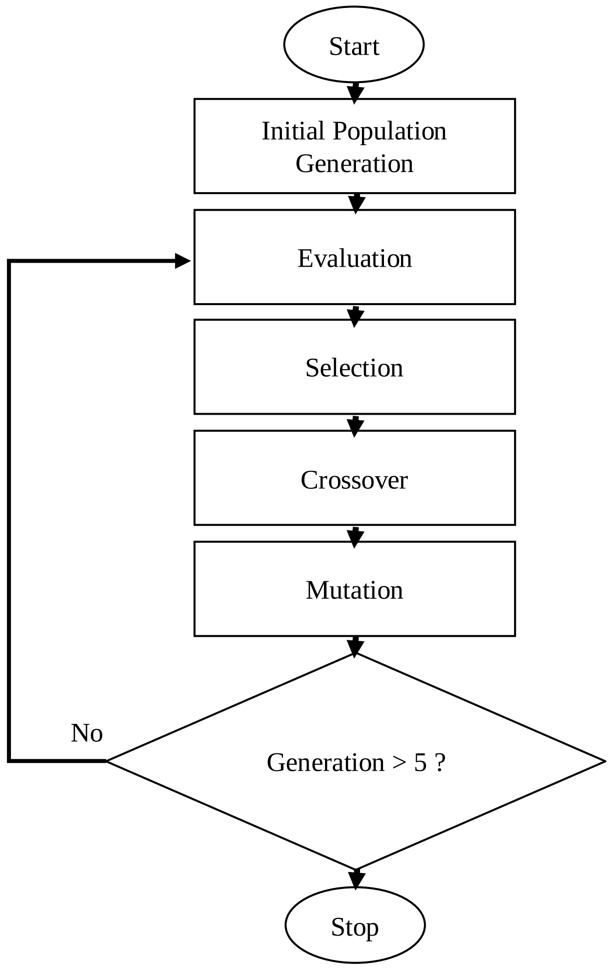

Figure 2: Flowchart of genetic algorithm processes applied in hyper-parameters optimization

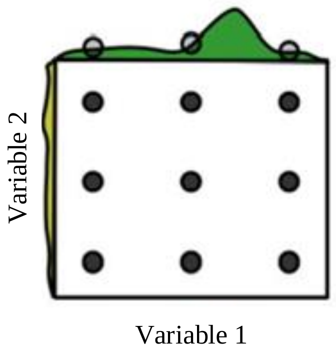

Figure 3: Grid Search Algorithm

The model is also trained based on the concept of GS to find the best hyperparameter's performance. The advantage of using GS is that, apart from it is simple to execute, it evaluates the classification performance for all possible combinations of hyperparameters [16]. The space search is illustrated in Figure 3. However, the downside is that as the number of hyperparameters increases, the numbers of evaluation function also increases exponentially. Therefore, it is not feasible for larger hyperparameters' space.

For both tuning methods, every combination of hyperparameters is trained for 10 epochs with batch size of 128. There will be 10% of the training data that is set as validation data by using the holdout validation method.

3. RESULTS AND DISCUSSION

The experimental work in this paper is divided into two experiment settings. The first experiment is to use GA method to find the best combination of number of hidden layers and number of neurons in each layer. The number of layers to be searched is range between 1 and 10 layers, and number of neurons is range between 50 and 1000 neurons, with increment of 50 neurons. The number of neurons is set to be the same in each of hidden layer. In addition, the parameter settings in GA method is further explored. The number of populations in GA is increased to expand the searching space.

In the second experimental work, GS method is used to find the best combination for network optimization. The searching space for GS method is similar with GA, where the number of layers range between 1 and 10 with number of neurons range between 50 and 1000, with increment of 50 neurons. For both experimental work, the comparison between performance of the GS and GA is analyzed in terms of accuracy and time taken to search for best hyperparameters.

3.1 Genetic algorithm search method

The initial settings for GA method are 5 generations, with the total of 75 populations. Figure 4 shows the population distribution of the setting. However, the plots on the diagram show lesser because there are repetitions of population across the generations. The classification accuracy using deep neural network for each hyperparameter combination is recorded in each generation. Then, the top 40% combination are remained and been forwarded to populate in the next generation. The rest combination has 10% chances to be chosen to the next generation.

The average accuracy for each generation is as shown in

Table 1

Table 1. The average accuracy increases dramatically from generation 1 to generation 4 then decrease in generation 5. Generation 1 has the lowest accuracy due to random picked hyperparameters. The generations onwards generate population base on the top 6 populations of previous generation. The decrease of average accuracy in fifth generation may be due to the occurrence frequency of mutation. In Table 1, “L” represents the hidden layer numbers and “N” represents the neurons number. The bolded data is the hyperparameters that carried forward to the next generation and generated the rest population base on them. At the fifth generation, the optimized hyperparameters scored 98.23% with 3 hidden layers with 900 neurons each.

Further experiment using GA method is performed by increasing the number of hyperparameters, with number of hidden layers is increased to range between 1 and 50 and number of neurons is range between 50 and 1000 neurons. The searching space for GA is expanded by increasing the number of generation and population. The number of generations is increased to 100 generations, whereas the number of populations is increased to 30 populations. Figure 5 shows the population distribution after increasing the searching space. Figure 6 shows the average accuracy after 100 generations, where the accuracy has achieved 98.5%.

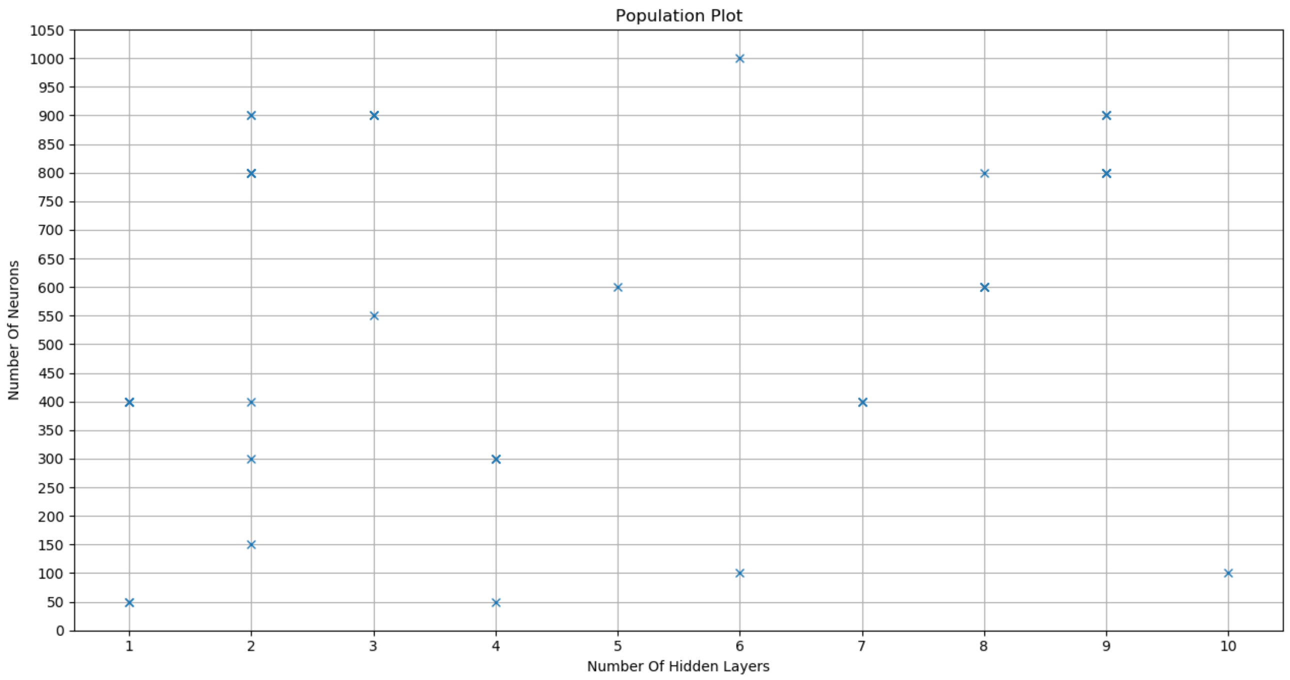

Figure 4: Population Distribution across 5 Generations

Table 1: Score Achieved by the Population for Every Generation

Generation 1			Generation 2			Generation 3			Generation 4			Generation 5		
L	**N**	**Score**	**L**	**N**	**Score**	**L**	**N**	**Score**	**L**	**N**	**Score**	**L**	**N**	**Score**
1	50	.9696	**1**	**400**	**.9809**	**3**	**900**	**.9791**	**1**	**400**	**.9812**	2	900	.9810
10	100	.9717	4	300	.9774	**1**	**400**	**.9824**	3	900	.9777	3	900	.9799
1	50	.9717	**7**	**400**	**.9786**	**2**	**800**	**.9796**	**3**	**900**	**.9820**	3	900	.9769
6	1000	.9772	**2**	**800**	**.9805**	**3**	**900**	**.9807**	3	900	.9739	3	900	.9746
4	**300**	**.9794**	**3**	**900**	**.9798**	9	800	.9737	2	800	.9811	3	900	.9817
2	150	.9786	**8**	**600**	**.9769**	7	400	.9761	3	900	.9739	1	400	.9804
3	**900**	**.9787**	9	900	.9757	8	600	.9781	9	800	.9753	3	900	.9821
9	**900**	**.9764**	2	400	.9780	1	400	.9776	3	900	.9795	3	900	.9763
2	**800**	**.9788**	**3**	**900**	**.9826**	3	900	.9765	**3**	**900**	**.9820**	**3**	**900**	**.9823**
8	**600**	**.9787**	4	300	.9773	8	800	.9788	**3**	**900**	**.9789**	3	900	.9750
1	**400**	**.9798**	9	900	.9694	**9**	**800**	**.9781**	**3**	**900**	**.9794**	5	900	.9758
6	100	.9739	**9**	**800**	**.9795**	**1**	**400**	**9790**	1	400	.9799	3	900	.9788
7	**400**	**.9789**	2	300	.9766	**3**	**900**	**.9805**	**3**	**900**	**.9815**	3	400	.9806
5	600	.9775	3	550	.9782	**3**	**900**	**.9812**	**2**	**900**	**.9829**	3	900	.9795
4	50	.9697	8	600	.9720	3	900	.9790	**3**	**900**	**.9826**	3	200	.9810

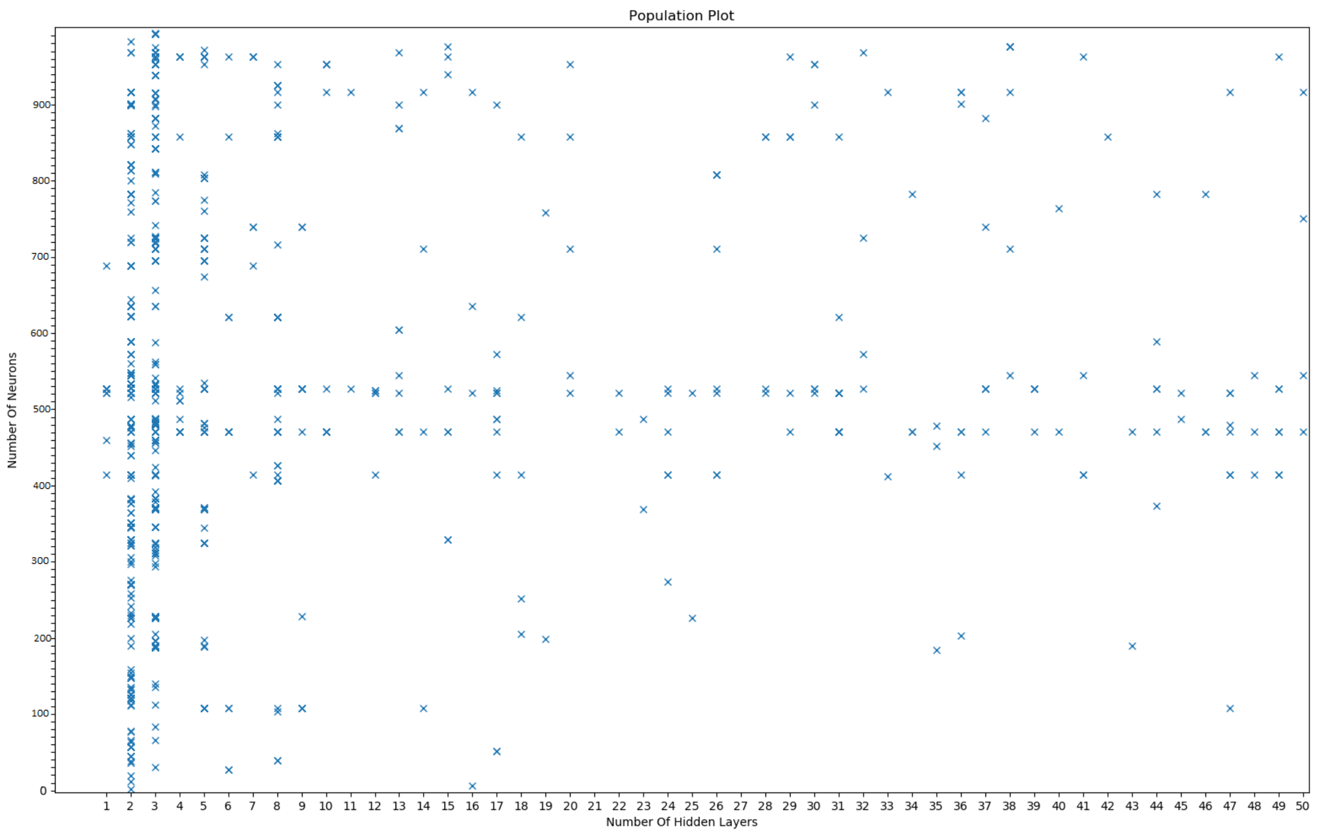

Figure 5: Population Distribution after Increasing the Search Space

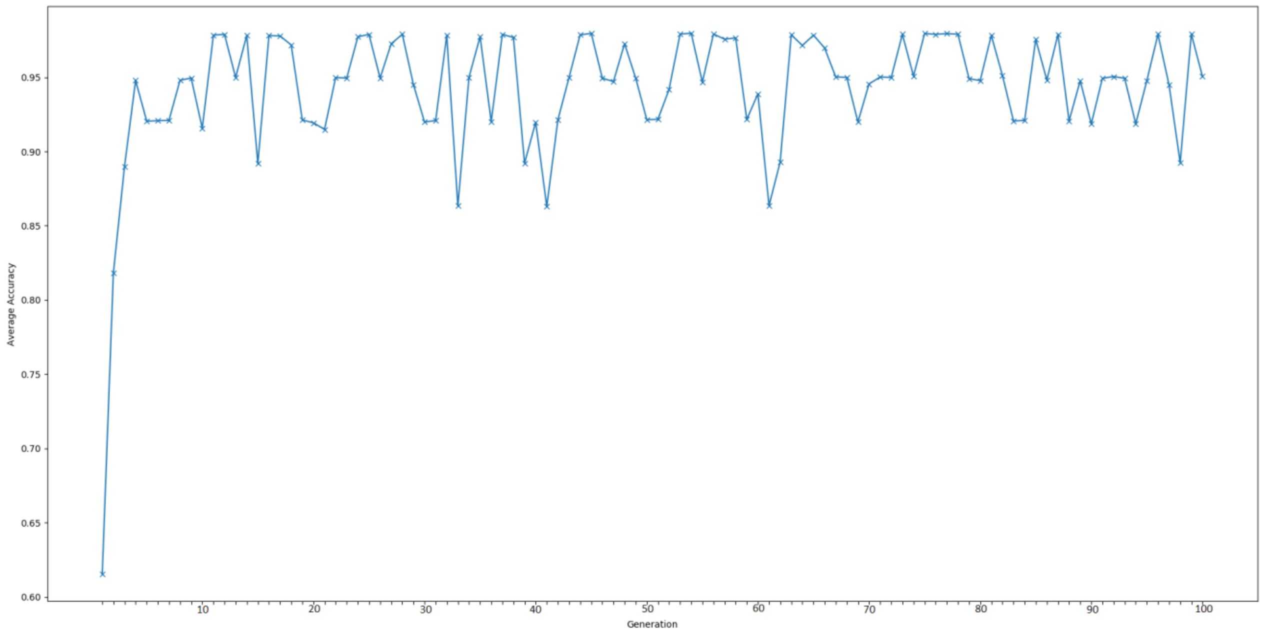

Figure 6: Average Classification Accuracy after 100 Generations

3.2 Grid search method

The simplest method to search for the best hyperparameters combination is by using GS method. Basically, gird search method finds the parameters combinations deterministically, by laying down a grid of all possible combinations of hyperparameters. Thus, the advantage of GS is that every hyperparameters on the grid are trained and scored in order to determine the best hyperparameters. However, the disadvantage is its computational cost, especially when the number of parameters to configure is increased.

In this work, the parameters to be searched is only two, which are the number of hidden layer and number of hidden nodes. Thus, it is still feasible to use GS method to find the best combinations of those two parameters. The number of hidden layers and number of neurons is set to be the same with the first setting of GA's method, which are the number of layers to be searched is range between 1 and 10 layers, and number of neurons is range between 50 and 1000 neurons, with increment of 50 neurons.

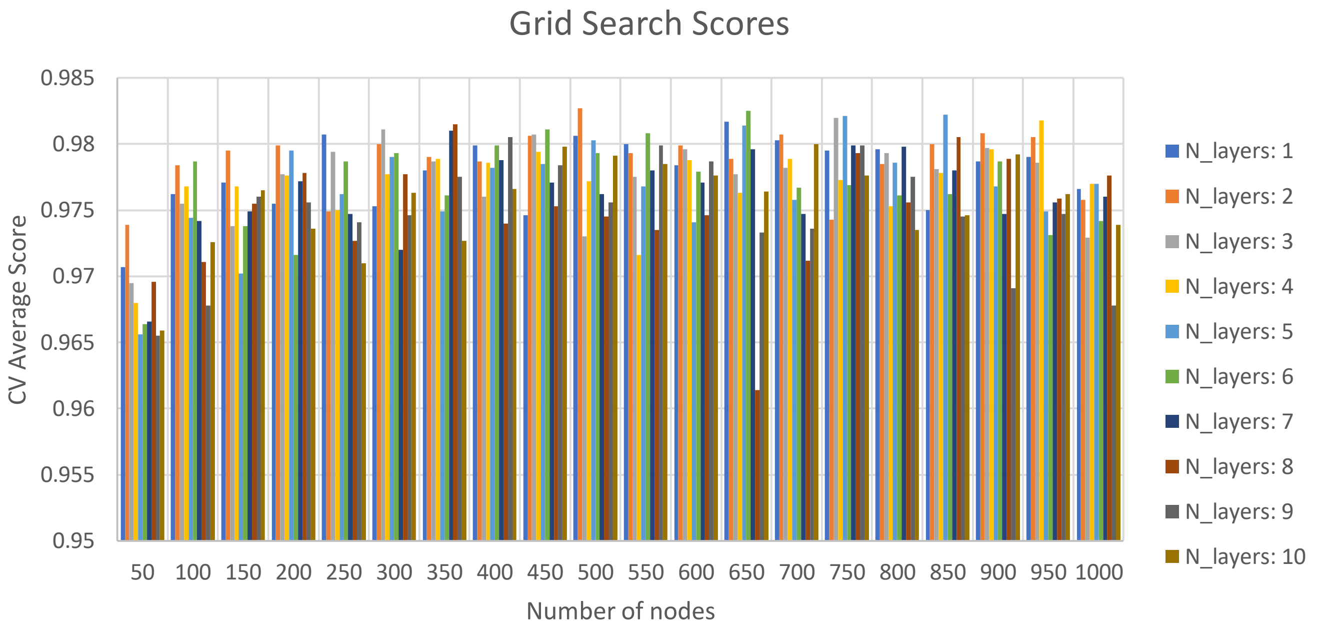

Figure 7: Accuracy for GS Method

The accuracy of each combination is plotted as shown in Figure 7. The model with 500 neurons with 2 hidden layers achieved the highest score 98.27%.; the second highest score is the model with 650 neurons with 6 hidden layers that scored 98.25%; the next highest score goes to the model with 850 neurons with 5 hidden layers that scored 98.22%. All the top three scored around the same.

Indeed, the number of neurons and hidden layer numbers affect the execution time in searching for the most optimized hyperparameters as shown in Table 2. Increasing the hidden layers and neurons increases the training time of data. The figure shows that 2 hidden layers with 500 neurons at every hidden layer take the least time (<100s) and achieving the highest accuracy.

Table 2: Comparison of accuracy and estimated time taken for top 6 best performance

No. of layers	No. of neurons	Accuracy	Estimated time taken (s)
2	500	98.27%	70
6	650	98.25%	191
5	850	98.22%	251
5	750	98.21%	201
3	750	98.20%	142
4	950	98.18%	251

3.3 Overall Performance Comparison of GA vs GS method

The best performance of MNIST classification using deep neural network tuned by GA and GS methods are compared. The best accuracy achieved by GA tuning method is 98.25%, whereas the best accuracy achieved by GS is 98.27%. The accuracy achieved by both methods are comparable. However, the time taken for GA to complete the classification task is half of the total time taken by GS, with 15098.4s or 4.19 and 30911.85s or 8.59 hours, respectively. The results are shown as in Table 3.

Table 3:Time taken for GS and GA to obtain optimized model

Method	No. of layers	No. of neurons	Accuracy	Estimated time taken (s)
Grid search	2	500	98.27%	30911.85
Genetic algorithm	6	650	98.25%	15098.4

4. CONCLUSION

The optimization of hyperparameters is important in building up the neural network model. GA is preferable as it uses lesser time and computation power in searching for the optimized hyperparameters. Both GA and GS achieve comparable accuracy where there is only a slight difference between them. GA technique is smarter when it involves a large search space of hyperparameters where GS is only suitable when the hyperparameters search space is small.

Further improvement can be done to increase the accuracy of the model by using the k-fold cross-validation (k-fold CV) technique for the validation data instead of holdout method. K-fold CV method is to divide the training set into k equal subsets; enables each subset to be used as validation set just once during training in order to reduce the overfitting scenario. The drawback of the method is the time taken will increase by k times for each hyperparameters combination.

Besides, other hyperparameters like the batch size, learning rate, activation function, epochs and weight initialization can be tuned as well. In the case where lots of hyperparameters are involved, GA technique is a better choice.

ACKNOWLEDGEMENT

The author would like to thank The Ministry of Education Malaysia for funding this paper via research grant FRGS/1/2017/TK04/UKM/02/4.

REFERENCES

[1] L. Sun, J. Wang, Z. Hu, Y. Xu, and Zhongwei Cui, "Multi-View Convolutional Neural Networks for Mammographic Image Classification," *IEEE Access*, vol. 7, pp. 126273–126282, 2019. https://doi.org/10.1109/ACCESS.2019.2939167

[2] V. R. Bhanuprakash Dudi, "Medicinal Plant Recognition based on CNN and Machine Learning," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 4, pp. 999–1003, 2019. httpps://doi.org/10.30534/ijatcse/2019/03842019

[3] B. S. de Araujo, H. L. S. de Almeida, and F. L. de Mello, "Computational Intelligence Methods Applied to the Fraud Detection of Electric Energy Consumers," *IEEE Lat. Am. Trans.*, vol. 17, no. 1, pp. 71–77, 2019.

[4] F. H. Hassan and N. A. Azelan, "Comparing Performance of Machine Learning Algorithms in a Flood Prediction Model with Real Data Sets," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 1.4, pp. 152–157, 2019. https://doi.org/10.30534/ijatcse/2019/2381.42019

[5] J. R. B. Del Rosario, "Development of a Face Recognition System Using Deep Convolutional Neural Network in a Multi-view Vision Environment," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 8, no. 3, pp. 369–374, 2019. https://doi.org/10.30534/ijatcse/2019/06832019

[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[7] G. E. Hinton, "Learning to represent visual input," *Philos. Trans. R. Soc.*, vol. 365, pp. 177–184, 2010.

[8] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, no. 1, pp. 1–127, Jan. 2009.

[9] D. Yu and L. Deng, "Deep Learning and Its

Applications to Signal and Information Processing," *IEEE Signal Process. Mag.*, vol. 28, no. 1, p. 145–+, 2011.

[10] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep Machine Learning — A New Frontier in Artificial Intelligence Research," no. November, pp. 13–18, 2010.
https://doi.org/10.1109/MCI.2010.938364

[11] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*, First Edit. O'Reilly Media, 2017.

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Adv. Neural Inf. Process. Syst. 19 Proc. 2006 Conf.*, vol. 19, p. 153, 2007.

[13] Y. Shevchuk, "Hyperparameter optimization for Neural Networks," 2016. .

[14] A. Severyn and A. Moschitti, "Hyperparameter Optimization with Factorized Multilayer Perceptrons Nicolas," *Ecml Pkdd*, pp. 1–16, 2015.

[15] A. Johnson, "Evaluating Hyperparameter Optimization Strategies," 2016. .

[16] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[17] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, pp. 437–478, 2012.

[18] M. Konomi and G. M. Sacha, "Influence of the learning method in the performance of feedforward neural networks when the activity of neurons is modified Highlights :," pp. 1–11.

[19] F. . H.K. Lam, S.H. Ling and P. K. S. T. .F. Leung, "Tuning of the Structure and Parameter of Neural Networks using an Improved Genetic Algorithm," *2010 Int. Conf. Bus. Econ. Res.*, no. 1, pp. 110–114, 2010.

[20] H. Sariffuddin and M. F. Ibrahim, "A genetic algorithm based task scheduling system for logistics service robots," *Bulletion Electr. Eng. Informatics*, vol. 8, no. 1, pp. 206–213, 2019.
https://doi.org/10.11591/eei.v8i1.1437

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[22] A. S. Walia, "Activation Function and its Type," 2017. .

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[24] R. Rojas, "Genetic Algorithms," in *Neural Networks - A Systematic Introduction*, 1996, pp. 429–450.

[25] M. Rahul, S. Narinder, and S. Yaduvir, "Genetic Algorithms: Concepts, Design for Optimization of Process Controllers," *Comput. Inf. Sci.*, vol. 4, no. 2, 2011.
https://doi.org/10.5539/cis.v4n2p39