Evaluating and Improving the Network Reduction Technology on the Artificial Neural Networks Optimised by the Genetic Algorithm

Yongchao Lyu

Research School of Computer Science, The Australian National University,
Building 108 (CSIT), Canberra, Australian Capital Territory 0200, Australia.
E-mail: u6874539@anu.edu.au

Abstract. This article continues a piece of research on eye-tracking analysis by adopting data from their experiments. Instead of statistically analysed the relationships between different variables, we performed classification on it. The genetic algorithm was employed for optimising hyperparameters of the neural networks, and its results were analysed. After the neural network was well-tuned, the network reduction technology was applied and evaluated as well. This research took place on both the neural networks with one hidden layer and two hidden layers for comparison. At the end of this article, an adjustment for the neuron pruning process was proposed to enhance the pruning effect.

Keywords: artificial neural network, genetic algorithm, neural network reduction, distinctiveness

1 Introduction

Colossal amounts of information already exist and is continuously created every day on the internet. In order to get the appropriate information needed, web users usually perform searches on some kind of search engines, such as Google. Lots of researchers indicated that web searching has already become the most important activity on the internet [1]. Meanwhile, search engines kept evolving to better present their search results on different size of screens [2]. Since there are close connections between user search behaviours and performance [3], it's important to understand user search behaviours before developing an effective search engine. Eye-tracking technology was booming at the beginning of this century, which can be considered as a feasible method to trace user behaviours. Many researchers utilised this technology for measuring how a user interacts with the search results on a web page [4], which can be used as the basis for improving the result presentation of search engines, thereby enhancing user search performance and experience.

Kim et al. conducted research using eye-tracking technology for investigating how screen sizes and task types impact on search behaviours and performances [5]. With the help of effective statistical tools, they derived several meaningful conclusions. This article continued Kim's research by adopting the data from their previous experiments. Instead of statistically analysing the relationships between different attributes, we performed classification and prediction on it. In this research, we first built an artificial neural network with the attributes of user behaviours and performance as inputs and the attribute *screen size* as the prediction target. Then several technologies are utilised to enhance the performance of the neural network. To find the best hyperparameters of the neural network, such as the number of epochs and the learning rate, we employed an optimisation method called *genetic algorithm*. Basing on the optimised hyperparameters, our neural network achieved acceptable performance. After that, we applied the *network reduction technology* to the trained modules for evaluating the effects of redundant neurons pruning. Both the neural network with one hidden layer and two hidden layers were involved in this research. By analysing the relationship between pruning effects and the distributions of pattern vector angles, we provided a feasible way to enhance the effects of the network reduction.

2 Method and Module Design

There are three major stages took place in this research. Firstly, we built an artificial neural network as the infrastructure for the classification and prediction task. The neural network was implemented in a flexible code structure, which can switch the number of hidden layers between one and two by simply changing a global variable. Secondly, we performed the *genetic algorithm* on the neural network for hyperparameters optimisation. By using the optimised and selected hyperparameters, our neural network achieved a reasonably high prediction accuracy on the testing dataset. Thirdly, we employed the *network reduction technology* for pruning redundant hidden neurons, using the *distinctiveness* property as the determinant of removable neurons. The performance of the neural network before and after pruning were compared.

2.1 Establish an Artificial Neural Network

The dataset employed in this research was adopted from the research of '*Eye-Tracking Analysis of User Behavior and Performance in Web Search on Large and Small Screens'* conducted by Kim et al. [5]. The dataset contains 4 attributes of user search performance, 22 attributes of user search behaviours, and 3 attributes of others, namely, *subject id*, *screen size* and *task type*. From the perspective of consistency with the original research, using user search behaviours and performance attributes to predict *screen size* should be a good choice. So this research set the *screen size* as the target for classification. Meanwhile, since the column *subject id* is only used for identifying participants and was useless in our research, we eliminated it from the dataset during the data pre-processing.

The dataset was split into two subsets, one was for training, which contains 80% of the original data, the other one was for testing, which contains 20% of the original data. Instead of randomly split the dataset, we used the fixed division method to ensure the stability and repeatability of the classification process. There was no validation dataset since the number of the examples was small and the computing time for training was short, we could evaluate the trained module directly through the testing dataset without waiting a long time. So, we don't have to waste parts of the data to evaluate the module during the training process for preventing overfitting. Furthermore, normalisation was also applied to the training dataset and the testing dataset, respectively, to get all data on the same scale. Thus avoid the classification bias to the attributes with bigger values. We implemented two normalisation algorithms in the codes, i.e. the *z normalisation* (the standardisation) and the *max-min normalisation*. One of them was chosen according to actual effectiveness.

The neural network was established with a flexible number of hidden layers, which means the number of hidden layers can be switched between one and two. By simply changing the value of a global variable, the hidden layers of the neural network would be adjusted automatically. We utilised the *cross-entropy* for computing the loss and introduced two optimisers, i.e. *SGD* and *Adam*. Similar to the normalisation methods, one of the optimisers was chosen according to the actual effectiveness in experiments. For getting the balance between robustness and efficiency of the training process, *min-batch gradient descent algorithm* is employed in our neural network as well. It split the training dataset into small batches for calculating the model errors, which can avoid local minima and converge faster than the *stochastic gradient descent algorithm*. All the hyperparameters were optimised by the *genetic algorithm* before using.

We mainly utilised the *confusion matrix* and the *overall accuracy* for evaluating the performance of the neural network, with the help of another two factors, namely, *sensitivity* and *specificity*. A *confusion matrix* is a useful tool for assessing how well a classifier can recognise tuples of different classes. For a binary classification problem, a confusion matrix is a 2x2 matrix with each entry indicates the number of tuples categorised to the actual class or the predicted one. Deriving from the confusion matrix, the *sensitivity* reflects the true positive rate, and the *specificity* reflects the true negative rate. *Overall Accuracy* is defined as the mean average of the *sensitivity* and *specificity,* which is considered more effective than regular *accuracy*, especially when the dataset is biased [6]. When applying the *genetic algorithm* for optimisation, we utilised the *overall accuracy* as the factor to evaluate how fitness a chromosome was.

2.2 Optimisation by the Genetic Algorithm

The famous naturalist Charles Darwin discovered rules of the *Natural Selection or Survival of the Fittest*, inspired by which computing scientists invented the *genetic algorithm*. Starting with some initial values for the variables used in an experiment, the algorithm continues to search for an optimal solution stochastically until the best ones are found or a finite number of generations have been produced [7]. In this research, we employed the *genetic algorithm* for searching the best hyperparameters of our neural network. The hyperparameters to be optimised include the number of hidden neurons, the number of epochs, the learning rate, and the size of mini-batch, each of which can be called as a *gene*. A set of these genes comprise a *chromosome*, and the goal of the *genetic algorithm* is to find the best chromosomes.

Table 1. The Constrains of Valid Value Ranges for Each Gene

Gene	Min Value	Max Value
The Number of Epochs	10	1000
The Learning Rate	0.001	0.500
The Size of Mini-Batch	5	640
The Number of Neurons (H1)	4	50
The Number of Neurons (H2)	2	25

For evaluating chromosomes, we utilised the *overall accuracy,* which has been defined at the end of section 2.1, as the *fitness value* to reflect how good a chromosome is. The chromosome had the highest *fitness value* was considered as the best. Since the process of initialisation and mutation are stochastic, we defined constraints of the valid value ranges for each gene, which are shown in Table 1. Our dataset contains a total of 640 tuples of data, so we set the maximum size of the mini-batch as 640. For this magnitude of data size, 1000 number of epochs should be enough in practice to well train the neural network. Meanwhile, a neural network with two hidden layers usually performs better if the number of neurons in the first hidden layer was about twice as large as the second hidden layer. That's why we set the maximum number of neurons in the second hidden layer as the half as the first hidden layer. Moreover, The size of the population of each generation was set as 10 and the process of evolution would stop after producing 100 generations. For better comparing the performances of the neural networks with different chromosomes, we set the seed of PyTorch as 8420 in the codes to ensure the initial weights are identical whenever the neural network was evaluated.

In this research, the selection method *Hall of Fame* was employed. For each generation, the best chromosome with the highest fitness values was selected and put into a parent pool. The parent pool was twice as large as the population size of one generation, which contains an archive of the best individuals found from each generation. To keep the diversity of genes in the parent pool, the chromosomes with identical genes would not be added twice. When producing the next generation, we set the *crossover rate* to 0.8 as a constant, which means about 80% of the population mate with another one in the same generation. At the same time, the *mutation rate* was set to 0.5 as a variable, which decreases along with generations and the *decay rate* is the reciprocal of the generation number. For example, if the current population was 100th generation, then the *decay rate* would be 0.01, and the *mutation rate* in this generation would be 0.5*0.01=0.005. The mutation could add new features to offspring and offer the possibility to get a better fitness value. By introducing a dynamic mutation rate, the parent pool could gather more diverse genes in the early stages of the evolution, but, in the meantime, would not lose too many good genes in the late stages.

2.3 Apply the Network Reduction Technology

The architecture of an artificial neural network can significantly influence its performance. Unlike input neurons and output neurons, it's hard to determine the number of hidden layers and the number of hidden neurons in advance. Some researchers claimed that the actual number of hidden neurons in an artificial neural network was usually higher than expected [10]. On the one hand, the excess number of hidden neurons may increase the risk of overfitting. On the other hand, it would also consume more computational resources. Gedeon and Harris believed that the extra hidden neurons just performed as supportive roles to significant ones or even not be used at all. Removing these trivial or unactivated hidden neurons is necessary and would not bring obvious side effects to a well-trained neural network [11].

There are several determinants could be used to identify which hidden neuron is removable, such as the *relevance*, *contributions*, and *sensitivity* [8-10]. This research employed the *distinctiveness* [11] for measuring how similar or opposite the functionalities of two hidden neurons are, and then determine which hidden neurons to remove. Gedeon and Harris indicated that if two hidden neurons are similar or opposite enough, then at least one can be removed [11]. More specifically, if the angle between the pattern vectors of two hidden neurons is less than 15 degrees, then one of them can be removed, and the weights of the removed neuron should be added to the remaining one. If the angle between the two pattern vectors is greater than 165 degrees, then both of them can be removed directly. To ensure the angle between pattern vectors can be range from 0 degrees to 180 degrees, we should normalise activations of hidden neurons to 0.5. In our actual experiment, the activations have already been normalised to 1 through the sigmoid function, so we only required to subtract 0.5 from them.

Before applying the *network reduction technology*, we must first determine the hyperparameters of the target artificial neural network. Although the hyperparameters optimised by the *genetic algorithm* described in the previous section could make our neural network achieve acceptable accuracy, however, these hyperparameters could not be used directly in the experiments for evaluating the effectiveness of the *network reduction technology*. That's because, through the optimising processes, the number of hidden neurons had already been reduced to a small number, there won't be many redundant neurons for pruning. So, in this research, we doubled the number of hidden neurons basing on the optimised hyperparameters. In other words, we employed 20 neurons for the first hidden layer and 10 neurons for the second hidden layer, with the number of epochs as 100 and the learning rate as 0.002.

Each time to perform the *network reduction technology* on the well-trained neural network, we recorded the module's current *accuracy* first and normalised the activations to 0.5 before computing *distinctiveness*. Cosine similarity of two pattern vectors was calculated and then converted to an angle in the unit of degrees. Since the angle calculation was performed on two pattern vectors, to avoid calculating the same vector pair twice, a duplication preventing mechanism was introduced into the program. Moreover, the indexes of the similar neuron pairs were stored separately in the codes from the opposite ones, so different pruning processes can be applied to them respectively. The program only printed out the neuron pairs whose vector angles were less than 15 degrees or greater than 165 degrees, and others are ignored. After everything was ready, the eligible hidden neurons would be pruned.

Although there are some ready-made packages in the PyTorch for pruning, such as *torch.nn.utils.prune*, none of them can be used directly in our experiment. PyTorch offers a basic class *prune.BasePruningMethod* for customising the pruning logic, but it is hard to define a dynamic strategy in its method *compute_mask*. That's because the neurons to be removed cannot be determined in advance before the neural network is well-trained. Gedeon and Harris indicated that if the weight linking two neurons became zero, the activation using this weight would be ignored [11], which is analogous to remove the neuron architecturally. Basing on this premise, we could perform the pruning process on a removable neuron by simply and directly set the weight from it to the higher layer as zero.

Gedeon and Harris also claimed that the neural network pruned via the method they mentioned should not require any further training [11]. So, in this research, we firstly verified the effectiveness of the pruning method by comparing the module's *accuracy* before and after applying the technology, without any further training after pruning. Next, we expanded the observations to the relationships between the pruning effectiveness and some potential factors, such as the number of hidden neurons, the number of hidden layers, and the distribution of the pattern vector angles. Through comparing and analysing the differences between the process for pruning similar neurons with the process for pruning opposite neurons, a feasible approach was proposed to enhance the performance of network reduction.

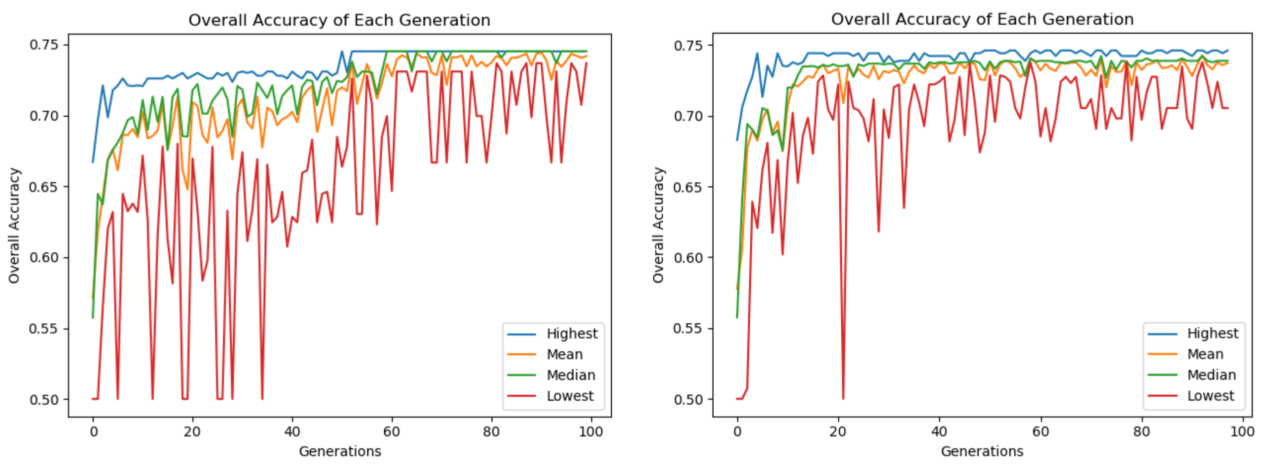
3 Results and Discussion

Through preliminarily tasting the artificial neural network built for this research, we found that *Adam* optimiser, which increased the prediction accuracy significantly, was better than the *SGD* one. Meanwhile, the normalisation performed on the training dataset and testing dataset respectively also enhanced the module's performance slightly. There were no significant differences between performances of the *z normalisation* and the *min-max normalisation*. That's because most of the attributes in our dataset have already valued to zero or one, perform further normalisation would not get obvious effects. Following sections illustrate the optimisation results of the *genetic algorithm* for hyperparameters and then the effectiveness of the *network reduction technology* was evaluated and discussed.

3.1 Optimisation Results of the Genetic Algorithm

The optimisation processes of the *genetic algorithm* for both the neural networks with one and two hidden layers are shown in Image 1 as below. The abscissa axis indicates the generation number and the ordinate axis represents the overall accuracy on the testing dataset. For the populations of each generation, we displayed both the best and the worst accuracies on the graph, as well as the mean and median values of the generation.

Image 2. The Optimisation Processes of the Genetic Algorithm for the Neural Networks with One Hidden Layer (Left) and Two Hidden Layers (Right).



At the beginning of the entire optimisation process, the program randomly initialised population for the 0th generation based on the constrains of valid value ranges for each gene. So, the overall accuracies of the 0th generation were relatively low for both of the neural networks with one and two hidden layers, and the mean and median values of the overall accuracies were biased to the worst individuals. As more generations were produced through crossover and mutation, the mean and median accuracies were getting closer to the highest value. Their increasing speeds were faster at the early stages of the evolution but slower at the late stages. That's because of at the late stages of the evolution, better and better individuals were selected from previous generations and placed into the *Hall of Fame*, and each individual in the new generation was reproduced directly by the top individuals of the *Hall of Fame*. The diversity of genes would be weakened at the late stages of the evolution.

The overall accuracies of the worst individual in each generation kept fluctuating during evolution due to the bad results of crossover or mutation. For example, we might get an acceptable accuracy when the number of hidden neurons was

10 and the number of epochs was 500, as well as when the number of hidden neurons was 50 and the number of epochs was 100. However, if the algorithm produced an individual with the number of hidden neurons was 50 and the number of epochs was 500 through a crossover, the neural network would probably be overfitting and the overall accuracy on the testing dataset might decrease. However, the oscillating amplitude of the worst accuracies reduced with the progress of evolution. In other words, the fluctuation of overall accuracies became smaller at the late stages of the evolution. That's because, on the one hand, the results of the crossover were more stable at the late stages, as discussed above; on the other hand, the mutation rate was lower at the late stages. As mentioned in the method section, we employed a dynamic mutation rate, which decayed with the number of generations. This approach facilitated us in both obtaining more diverse genes at the early stages and keeping more selected genes at the late stages of the evolution.

By comparing the processes of optimisation on the neural networks with one and two hidden layers, we found that the neural network with two hidden layers converged faster than the neural network with one hidden layer. That's because the structure of the former was more complicated, which allows it to achieve higher accuracy faster. The philosophy is similar to a deeper neural network is easier to be overfitted than a shallow one.

Table 2. Top 3 Chromosomes in the *Hall of Fame* for the Neural Networks with One Hidden Layer and Two Hidden Layers.

Hidden Layers	H1 Neurons	H2 Neurons	Epochs	Learning Rate	Mini-Batches	Accuracy
1	4	0	48	0.002	81	74.51%
1	9	0	48	0.002	81	74.51%
1	9	0	48	0.002	80	74.51%
2	12	3	40	0.003	119	74.61%
2	10	7	40	0.003	119	74.41%
2	11	5	40	0.003	46	74.22%

The top 3 individuals in the *Hall of Fame* for both the neural network with one hidden layer and two hidden layers are shown in Table 2 as above. For both of these two kinds of neural networks, the highest overall accuracies all reached about 74.5% and the learning rates were about 0.002. The numbers of epochs were all below 50, however, the numbers of epochs for the one-hidden-layer neural networks were slightly higher than the two-hidden-layer ones. Meanwhile, the numbers of hidden neurons were significantly lower than our expectation.

3.2 Effectiveness of the Network Reduction Technology

The results for evaluating the effectiveness of *network reduction technology* are displayed in Table 3. We divided some small angle ranges to record the number of pattern vector angles falls into that range. This should be an effective way to observe the distribution of angles, and it also facilitated our attempts to find relationships between the pattern vector angle distribution and the pruning effectiveness. There are two major groups in the result table, which correspond to the different number of hidden layers. Since the pruning effectiveness was not ideal on the neural network with two hidden layers, and no obvious changes observed on its performance each time we tried, there is only one entry of result for it. For the neural network with only one hidden layer, we compared the pruning effects on the different number of hidden neurons. The number of result entries for the neural network with 20 hidden neurons is the largest. This group of results was used for evaluating the relationship between the pruning effects and the number of removable neurons, and the relationship between the pruning effects and the distribution of the pattern vector angles.

From the results, we first noticed that pattern vector angles are not distributed evenly or uniformly. Almost no vector angle falls into the range that less than 5 degrees or greater than 175 degrees. It means that in our experiment, we did not find many hidden neurons which were very similar or very opposite to another. If too many neurons were removed even though they're not very similar, some negative effects might take place at the module. By comparing the results of the neural network with 20 hidden neurons to the neural network with 50 and 100 hidden neurons, we found that more hidden neurons were removed from the latter, and their post-pruning accuracy dropped down more significantly as well. It's easy to understand why there were more removable hidden neurons in a bigger neural network. The more hidden neurons a neural network had, the more redundant its structure might be, so the more removable neurons could be found. When the number of hidden neurons was only 20, sometimes we found there was no removable hidden neuron at all after the neural network was well-trained, which means the structure of the neural network was not redundant.

Overall, the performance of the *network reduction technology* based on methods provided by Gedeon and Harris [11] worked well, and it did not obviously affect the prediction accuracy of the well-trained neural networks. The accuracy of a neural network was almost the same before and after applying the redundant neurons pruning. However, for the 4th and 5th result entries in Table 3, their prediction accuracies were close before pruning, the numbers of removable hidden neurons were the same, and their pattern vector angles were both distributed in the range of 165-170 degrees. But, while the post-pruning accuracy of the 5th entry only changed a little, the accuracy of the 4th entry dropped dramatically. The same symptom could be also observed in the 6th and 7th result entries. The 6th and 7th entries had an identical prediction accuracy before pruning, the same number of hidden neurons to be removed, and their pattern vector angles were in the same distribution, however, their post-pruning accuracies were very different. It indicates that the *network reduction technology* was not always stable, especially when more than one neurons were removed. More evidence can be found throughout the table, such as the result entries from 10th to 14th.

Table 3. Module Accuracy Before and After Pruning Hidden Neurons.

ID	NHL	H1	H2	[0, 5]	(5, 10]	(10, 15]	(165, 170]	(170, 175]	(175, 180]	ABP (%)	AAP (%)
1	1	20	-	0	1	0	0	0	0	69.53	69.53
2			-	0	0	0	0	1	0	67.97	65.62
3			-	0	0	2	0	0	0	67.97	68.75
4			-	0	0	0	2	0	0	67.19	55.47
5			-	0	0	0	2	0	0	68.75	64.06
6			-	0	0	1	1	0	0	68.75	67.19
7			-	0	0	1	1	0	0	68.75	56.25
8			-	0	0	2	1	0	0	67.97	64.84
9			-	0	0	1	2	0	0	70.31	57.03
10			-	0	0	0	4	0	0	68.75	56.25
11			-	0	1	5	1	0	0	70.31	73.44
12			-	0	0	5	4	0	0	67.97	54.69
13		50	-	0	3	9	5	2	0	68.75	55.47
14			-	0	2	12	9	1	0	68.75	64.84
15		100	-	0	10	37	16	3	0	68.75	50.78
16	2	20	10	0	11	12	10	12	1	70.31	46.88

NHL - The **N**umber of **H**idden **L**ayers
H1, H2 - The number of neurons in the first, second **H**idden layer
(a1, a2] - The number of vector pairs whose angle falls in the range (a1, a2]
ABP, AAP – The prediction **A**ccuracy **B**efore and **A**fter the **P**runing

Furthermore, we picked out the entries whose post-pruning accuracies dropped below 60% from Table 1 and compared them with other result entries. We found that, in similar conditions, the result entries with lower post-pruning accuracies usually contain more opposite neuron pairs than others. One example is the 8th and 9th result entries. Both of them had three pairs of hidden neurons to be removed (they're in a similar condition). However, the 9th entry had more neuron pairs marked as opposite, meanwhile, it gets a lower post-pruning accuracy. Another "coincidence" took place as well on the 11th and 12th entries. The 11th and 12th result entries had a similar number of removable neurons, but a different distribution of the pattern vector angles. The 12th result entry had more opposite neuron pairs than the 11th entry, and, at the same time, it got a lower post-pruning accuracy. Were these phenomenons accidental or inevitable?

Let's review some details of the pruning process again. For the hidden neurons whose pattern vector angles less than 15 degrees, one of them can be removed. The weight vector of the removed neuron must be added to the reserved one. In contrast, for the hidden neurons whose pattern vector angles greater than 165 degrees, both of them can be removed directly without keeping their weights. It is a reasonable solution because the opposite neurons probably correspond to opposite weights, and the sum of their weights would be very close to zero if they are opposite enough. Set the weights of two neurons to zero should not have obvious side effects when they were diametrically opposite. However, for the actual situation in our experiment, almost no pair of hidden neurons whose the pattern vector angle greater than 175 degrees, let alone diametrically. So, for the pairs of opposite hidden neurons, is it possible to improve the post-pruning accuracy by keeping the weights of them? In other words, if we remove only one of them, and add the weight vector of the removed neuron to the reserved one, maybe the neuron pruning performance could be enhanced.

Based on the above conjecture, we adjusted our codes by following a new strategy for pruning the opposite neuron pairs. For the hidden neurons whose pattern vector angles greater than 165 degrees, instead of removing them both, only one of them would be removed. The weight vector of the neuron removed must be added to the reserved one. We run the program again after the adjustment and wonder whether there were some differences against the previous evaluation.

The new results are shown below in Table 4. By comparing the two result tables, we found that there is less number of result entries whose post-pruning accuracies dropped below 60% in Table 4, and the accuracies look more stable and robust than before. The 6th and 7th result entries in Table 4 have a similar condition against the 8th and 9th result entries in Table 3, no matter the number of removable neurons or the distribution of pattern vector angles. However, the post-pruning accuracies of the result entry pair in Table 4 are closer to each other than the entry pair in Table 3. Another evidence is the 9th and 10th result entries in Table 4, both of which has a similar number of removable neurons but a different distribution of pattern vector angles. In comparison with the 11th and 12th result entries in Table 3, the post-pruning accuracies of the 9th and 10th result entries in Table 4 are not quite different to each other as the entries in Table

3 were. Furthermore, the post-pruning accuracy of the 10th result entry in Table 4 is more acceptable without dropped dramatically, even though it contains more opposite hidden neuron pairs.

Table 4. Module Accuracy After an Attempt at Pruning Algorithm Optimization.

ID	NHL	H1	H2	[0, 5]	(5, 10]	(10, 15]	(165, 170]	(170, 175]	(175, 180]	ABP (%)	AAP (%)
1	1	20	-	0	0	1	0	0	0	69.53	69.53
2			-	0	0	0	1	0	0	69.53	68.75
3			-	0	0	2	0	0	0	67.19	67.97
4			-	0	0	0	2	0	0	68.75	66.41
5			-	0	1	0	1	0	0	68.75	67.19
6			-	0	0	2	1	0	0	69.53	60.94
7			-	0	0	1	2	0	0	68.75	59.38
8			-	0	0	1	3	0	0	70.31	65.62
9			-	0	1	2	1	0	0	69.53	66.41
10			-	0	0	2	2	1	0	68.75	64.06
11			-	0	0	5	2	0	0	69.53	59.38
12			-	0	1	3	5	0	0	69.53	51.56
13		50	-	0	4	8	10	1	0	67.19	61.72
14			-	0	5	11	15	0	0	68.75	59.38
15		100	-	0	5	28	29	6	0	68.75	60.94
16	2	20	10	6	8	6	9	14	3	67.19	46.88

NHL - The **N**umber of **H**idden **L**ayers
H1, H2 - The number of neurons in the first, second **H**idden layer
(a1, a2] - The number of vector pairs whose angle falls in the range (a1, a2]
ABP, AAP – The prediction **A**ccuracy **B**efore and **A**fter the **P**runing

Although the performance of the *network reduction technology* was still not quite stable when there were many hidden neurons to be removed, our adjustment on the pruning strategy worked and slightly enhanced the pruning effect.

4 Conclusion

This article continues the research '*Eye-Tracking Analysis of User Behavior and Performance in Web Search on Large and Small Screens'* [5] by adopting the data from their experiments. Instead of statistically analysing the relationships between different variables, we performed classification and prediction on their dataset. To maintain consistency with the original research, we set the attribute *screen size* as the predicting target and used the data of user behaviours and performance as the inputs for training neural networks. Several technologies were employed for building the neural network and getting acceptable performance, such as the *normalisation* and *mini-batch gradient descent algorithm,* etc. Moreover, for more objectively and comprehensively evaluating the effects of different technologies, both the neural networks with one and two hidden layers were involved in this research.

The *genetic algorithm* was utilised in this research for optimising the hyperparameters of the artificial neural network. Starting from relatively low values, the mean and median accuracies of each generation kept improving and approached to the highest accuracy. In the meantime, the highest accuracy increased as well and at the end of the evolution, it's improved by 6-7% for both the neural networks with one and two hidden layers. In contrast, the worst accuracies oscillated throughout the entire evolution due to crossover and mutation. However, the amplitude of the oscillations was getting smaller and smaller with generations because of the decaying of the mutation rate. We also found that the two-hidden-layer neural network converged faster to the highest accuracy than the one-hidden-layer neural network did.

The *network reduction technology* was applied to the neural networks with one and two hidden layers as well. We utilised the *distinctiveness* as the determinant for finding redundant neurons and pruned them by setting corresponding weight vectors to zero. In this research, we found that the effect of the network reduction was generally acceptable for the one-hidden-layer neural network, but not ideal for the two-hidden-layer ones. The effectiveness of neuron pruning was better when the number of removable hidden neurons was smaller. Moreover, the pruning effects were not always stable, especially when there were many opposite hidden neuron pairs. By analysing the pruning algorithm, a feasible adjustment on the network reduction process was proposed and slightly enhanced the effects of the pruning.

Next, we will implement more properties, such as *relevance*, *contributions*, and *sensitivity*, to determine which hidden neuron can be removed. By combining these factors and even give them different weight values when making decisions, maybe a more reasonable and precise approach can be found to determine whether a hidden neuron is removable.

References

1. Liu, F.: 'How Information-Seeking Behavior Has Changed in 22 Years', *NN/g Nielsen Norman Group*, viewed 1st May 2020, https://www.nngroup.com/articles/information-seeking-behavior-changes/ (2020)
2. Jones, M, Buchanan, G & Thimbleby, H, 'Improving web search on small screen devices', *Interacting with Computers*, vol.15, pp. 479-495. (2003)
3. Granka, LA., Joachims, T., & Gay, G.: 'Eye-tracking analysis of user behaviour in WWW search'. *In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR' 04)*, pp. 478-479. (2004)
4. Buscher, G., Cutrell, E., Morris, MR.: 'What do you see when you're surfing? Using eye-tracking to predict salient regions of web pages', *Conference on Human Factors in Computing Systems - Proceedings*, pp. 21-30. (2009)
5. Kim, J., Thomas, P., Sankaranarayana, R., Gedeon, T., & Yoon, HJ.: 'Eye-tracking analysis of user behaviour and performance in a web search on large and small screens', *Journal of the Association for Information Science and Technology*, 66(3), pp. 526-544. (2015)
6. Alberg, AJ., Park, JW., Hager, BW., Brock, MV., Diener-West, M.: 'The use of "overall accuracy" to evaluate the validity of screening or diagnostic tests'. *J Gen Intern Med.* pp. 460-465. (2004)
7. Carlos, A. Coello, C.: 'An Introduction to Evolutionary Algorithms and Their Applications', *Proceedings of the 5th international conference on Advanced Distributed Systems*, pp. 425-442. (2005)
8. Mozer, MC., Smolenski, P.: 'Using relevance to reduce network size automatically', *Connection Science,* vol. 1, pp. 3-16. (1989)
9. Sanger, D.: 'Contribution analysis: a technique for assigning responsibilities to hidden units in connectionist networks', *Connection Science*, vol. 1, pp. 115-138. (1989)
10. Karnin, ED.: 'A simple procedure for pruning back-propagation trained neural networks', *IEEE Transactions on Neural Networks*, vol 1, pp. 239-242. (1990)
11. Gedeon, TD., Harris, D.: 'Network Reduction Techniques', *Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE*, vol. 1, pp. 119-126. (1991)