Improving the accuracy and efficiency of the Neural Network Classification with Auto-encoder And Genetic Algorithm

Linzhao Wu

Research School of Computer Science, Australian National University, Australia
u6191643@anu.edu.au

Abstract. The common issues people struggle with when they are doing the work of classification are improving the efficiency and accuracy of training. Some of the reasons for this are there are too many unnecessary features and the limitation of original training dataset. This paper addresses the first issue by proposing an approach of the Genetic Algorithm (GA) to select essential features from original dataset to improve training efficiency. The second issue is dealt with in this paper by extending the original dataset with the blurred output data of an auto-encoder-decoder which can be considered as a drop-out measure to improve its classification accuracy. Therefore, my experiments can be divided into four parts as following, the realization of classification model, the efficiency improvement with the method of GA, the accuracy enhancement of auto-encoder, and the combination of the above two measures to achieve better accuracy with similar amount of training time. My results of these experiments confirm my assumption of the availability of these methods. The feature selection method can reduce the half number of features by achieving similar accuracy, while dataset extending measure can enhance the accuracy by almost 10%. Furthermore, the combination method of these two successfully achieve better accuracy and efficiency, and the improvement is remarkable.

Keywords: Neural Network, Machine-Learning, Auto-Encoder, Genetic Algorithm, Feature Selection, Classification, Drop-out.

1 Introduction

There is a considerable number of applications for neural network, the most essential one of them is classification. However, the training of neural network is sometimes problematic. To be specific, the training process of some tasks is time consuming due to redundant features, which makes the neural network require more neurons and its training process last longer. Therefore, extracting optimal features amongst the given features without hurting the accuracy is of vital importance in the process of data preprocessing [11] [12]. In this paper, a feature subset selection measure based on a genetic algorithm [13] is introduced, and experiments related to the classification is later evaluated. The motivation behind these experiments is to testify the effectiveness of genetic algorithm on feature selection.

Another focus of this paper is on enhancing the accuracy of classification by extending the existing dataset with the processed dataset. The original dataset is put into an auto-encoder-decoder [1] and the output of the auto-encoder-decoder is the processed data. Then, the combination of the processed data and the original one is used as the input of the neural network. The objective of this measure is trying to use the blurring effect of data generated by the decoder as a substitution of the drop-out [2] method to extend the existing dataset and eliminate the over-fitting in the training process to achieve better accuracy.

After the above two experiments, these two approaches are combined for the sake of achieving higher accuracy with the same time consumed. This is because the feature selection deducts redundant features while auto-encoder generates more data out of the left features. The ultimate goal of these experiments is to combine these two approaches to achieve better performance of the classification neural network.

The training and testing dataset are the “mushroom” dataset from UCI collection [3], which is provided for classification, with 23 features and 2 labels. In comparison, the classification in the work of Hall and Simith [4] is also used to evaluate the classification results.

2 Methods

The whole processes of my experiments can be divided into four parts. The first one is the basic classification network which is a simple full-connection neural network; the second one uses genetic algorithm to select subset of the features of original dataset to minimize the training time and enhance training efficiency; the third one is using auto-encoder-decoder to generate blurring dataset from the original dataset to increase the training accuracy; the last one combines the above measures, the purpose of this is increasing the amount of training data with truncated features to enhance training accuracy but not increase the training time of the classification neural network.

2.1 Classification Network

The implementation of the classification neural network is simple, with three layers of neurons included. Each neuron from last layer connects every unit in the next layer of this simple neural network with weights, also known as a full-connection network, which can be illustrated as following Fig.1.

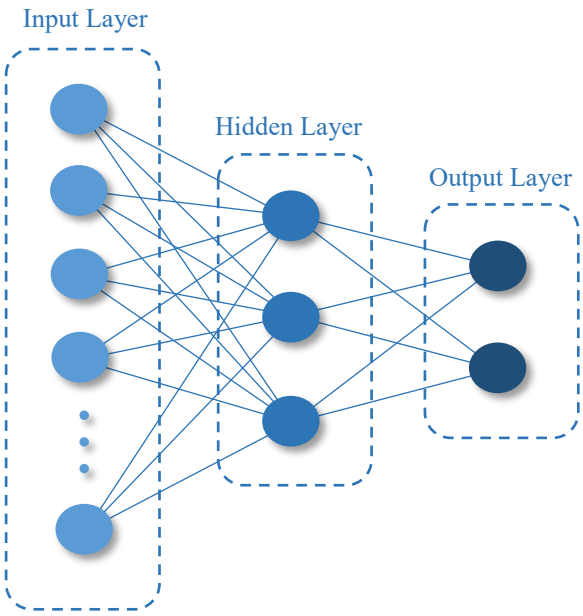

Fig.1. Classification neural network.

The neuron number of the input layers is the same as the number of features. In the usage of genetic algorithm, the number of input layer is the same as the number of the selected features, which means the number is changing with the genetic difference of individuals. On the other hand, in the instance of evaluation of the blurring effect of auto-encoder, the number of neurons of input layer is the same as the output layer of the auto-decoder for the reason that this classification network uses the result of the auto-encode-decode network. The motivation behind this is trying to realise the similar function of drop-out, which eliminating the over-fitting effect in the training session. This is achieved by adjusting the number of neurons in the hidden-layer of the auto-encode-decode network to generate the blurring effect on the decoded pictures, which hopefully having the same effects as the drop-out [2] process. The number of neurons in the hidden-layer is around half of that in input layer to capture the information of the features of input data. There are only two neurons in the output layer which represent the two labels of the "mushroom" dataset.

This simple classification network uses simple training measure as well. The optimizer used is Adam, which is an optimizer for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments [17]. The loss function in this classification network is the cross-entropy loss function which is motivated by an adaptive algorithm for estimating probabilities of rare event in complex stochastic networks [18]. The preprocessing of the "mushroom" includes normalization which convert the features that represented by single letters in the original databases to a floating number in the range of 0 and 1. The reason for this is keeping each feature having similar weights in the network. In addition, the mini-batch [6] is also used to avoid the local minimum effect in the training process, which is essential to keep out of local minimum value. The original dataset is divided into two parts, seven-eighths of them is training dataset, while the left one-eighth is testing dataset. However, the training dataset is shuffled in each epoch to avoid reliance of the sequence of data training.

The evaluation method in my experiments is accuracy which is a percentage number made by the number of correct predictions divided by the total number of predictions.

2.2 Feature selection by genetic algorithm

The original idea of genetic algorithms is on the basis of the Darwinian's theory of evolution, such as the survival of the fittest [14]. This method is commonly used to address optimization problems where the evolution of a specific biosphere is simulated with the concepts of mutation, crossover and reproduction and so on. The simulation of evolution is happened in the simulated environment of fitness function, which evaluate the survivability of individuals. After many generations with so-called reproduction and selection, new generations with better gene performance in fitness may be picked up.

Similarly, the objective of the method of feature selection in my experiments is to extract the best features from the ones in the original mushroom dataset, which can simplify the calculation of later classification network [15]. The flow chart of my implementation is illustrated as Fig.2.

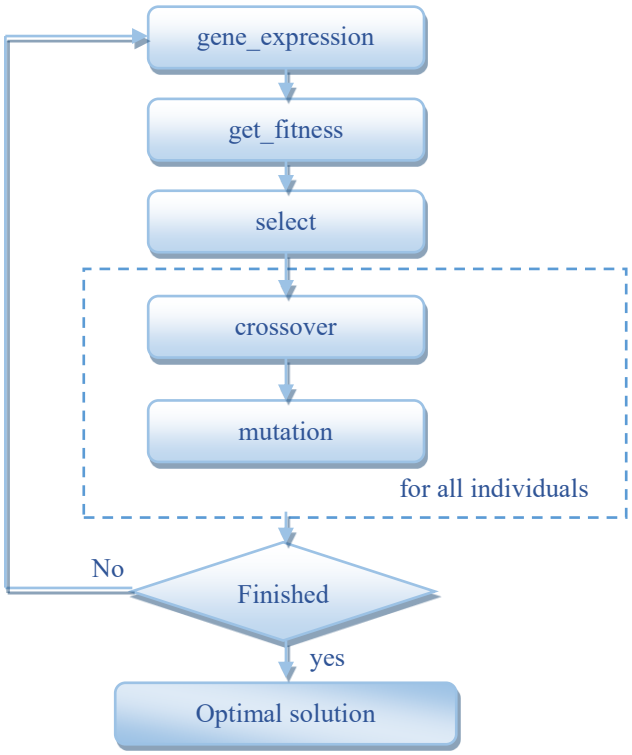

Fig.2. Genetic algorithms flowchart.

In the Fig.2, The specific steps of genetic algorithm are as follows:

- **gene expression:** A binary string is used to represent the chromosome which express the selection of features with the length of feature numbers. Genes are generated randomly by number 0 and 1, where 0 means this feature is excluded while 1 represent the usage of its corresponding feature. After this stage, all the individuals in the population are composed by one chromosome.
- **Get fitness:** This function denotes the fitness of one chromosome (one subset of features), which is calculated by the accuracy of the training result of module. The higher number of fitness value, the better one individual fit the environment which means the selection of subset of features are desirable.
- **Selection:** The selection method used in this paper is so-called Roulette wheel selection [16], which is a frequently used method in genetic algorithm. It follows the simple rules: the higher fitness value one individual has, the larger possibility of its surviving, which means the chances of being selected is proportional to individual fitness value. It can be denoted as the formula below:

$$p_i = \frac{\omega_i}{\sum_{i=1}^{N} \omega_i} \quad (i = 1,2,\ldots,N) \tag{1}$$

  Where p_i is the probability of the i-th individual is selected, while N is the number of individuals, and each of them is characterized by its fitness $\omega_i > 0$ (i = 1,2,….,N).
- **Crossover:** For each individual A, a crossover process is performed with another randomly picked one B in the selected population. The gene of this individual A is also randomly replaced by the gene of the other individual B. After the exchange of chromosome, a new child C is given birth, and this child C replace its parent A in the population. This crossover is implemented in a pre-defined cross rate (0.8) which means crossover happens in 80% times.
- **Mutation:** Mutation also happens in a pre-defined mutation rate for each child individual. For each gene in the chromosome, a random value is generated. If this number is smaller than the mutation rate, its corresponding gene value reverse.
- **Iteration:** The above process happens in a specific number of generation, which is also predefined. If the population number is large, the number of generation can be less. Otherwise, if the population number is small, the generation number should be larger.

2.3 Auto-Encoder and Auto-Decoder neural network

The realization of auto-encoder and auto-decoder is inspired by the work of Gedeon and Harris [1] using shallow feed-forward neural network of three-layer neurons. Each neuron from last layer connects every unit in the next layer of his simple neural network with weights, also being known as a full-connection network. The numbers of input neurons (the first layer) and output neurons (the last layer) are the same, so as to recover the original image. However, the hidden layer (the second layer) has fewer number of neurons for the sake of compressing images, and the quality of the decoded image is positively correlated to the number of hidden neurons. More hidden units can create better decoded images that more closely resemble original ones. The network can be illustrated as Fig.3.

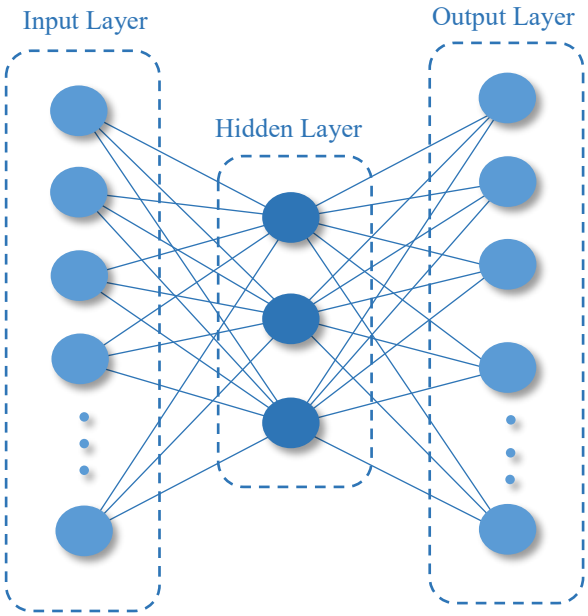

Fig.3. The auto-encoder neural network.

The training measure used is the traditional back-propagation. The input data are firstly fed forward in the neuron layers, then, are compared with the difference with the labels. In the scenario of encoding and decoding images, the labels are actually the input data for the purpose of recovering original images. Then, update the weights of neurons according to the calculated errors with the Adam optimizer [5]. The activation function of the hidden layer used in this paper is the sigmoid function, since the input data have been normalized in the range of 0 to 1 so do the output data. The loss function used in this encoder-decoder neural network is mean-squared-error (MSE) function. This is because that this function is able to compare two arguments of the same dimension, which is essential in this application as the labels are actually input data. Whereas other loss functions that are for classification cannot do this work because their hyper-parameters are labels that are different with input data. Similarly, the usage of encoding and decoding mushroom data is using the same auto-encoding network to generate the blurring effect and extend the original dataset.

2.4 The combination of genetic algorithm and auto-encode for classification

The last experiment is the combination of genetic algorithm and auto-encode for preprocessing method of the classification neural network of mushroom to achieve a higher accuracy with the similar time consumed. This is because the feature selection deducts redundant features while auto-encoder generates more data out of the left features. The objective is to combine these two combined approaches to achieve better performance of the classification of neural network. The original dataset with full features are firstly put into the process of genetic algorithm to select an essential subset of features. Then, the dataset with the left features are put into an auto-encoder to generate the blurred dataset. After that, this blurred dataset is combined with before blurring the dataset to create an extended dataset with more data of fewer features. At last, these data are put into classification neural network. The training process of classification network is the same as before, while the beginning and end of the training time is recorded to be compared with the training process of the original dataset to testify the improvement of training efficiency and accuracy.

3 Result and Discussion

3.1 feature selection with genetic algorithm

The accuracy is the result of testing of classification neural network, so the training process and testing process of the classification network run once for each individual in the population of each generation. For example, in a population of 100 individuals, if the genetic algorithm runs 100 generations, the total run times of training and testing process will be 10,000 times. Due to the prohibitive time, it is critical to reduce the training time of classification network. Two methods have been done to accelerate the training process, the first one is this network trained using GPU. I transfer the training process of Pytorch from CPU to GPU, which makes the training process almost 10 times faster. The second one is making the learning rate as large as possible, so that the epoch number can be as few as possible. At last the network can be trained at the accuracy of more than 95% in just 15 epochs with the learning rate of 0.03. The loss function and

optimizer is MSE and Adam respectively, and the number of neurons in the hidden layer is set to 2 to minimise the training time. After these improvements the training process can be finished within 2 seconds.
To get better and more thoroughly results, the population of my experiment is set to 100, and the generation is also set to 100, while the cross rate of individual crossover is 0.8 and the mutation rate of each gene is set to 0.002. After the almost 6 hours evolution the results of the accuracy and the number of features through the whole period is shown in box diagram of Fig.4, and the number of individuals corresponding to different number of features in the whole 100 generation is shown in the Fig.5.

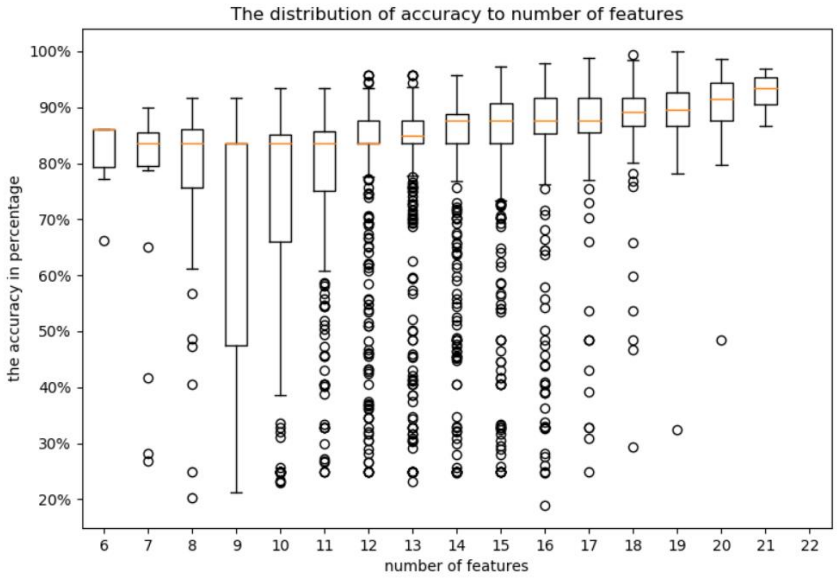

Fig.4. The distribution of accuracy to number of features.

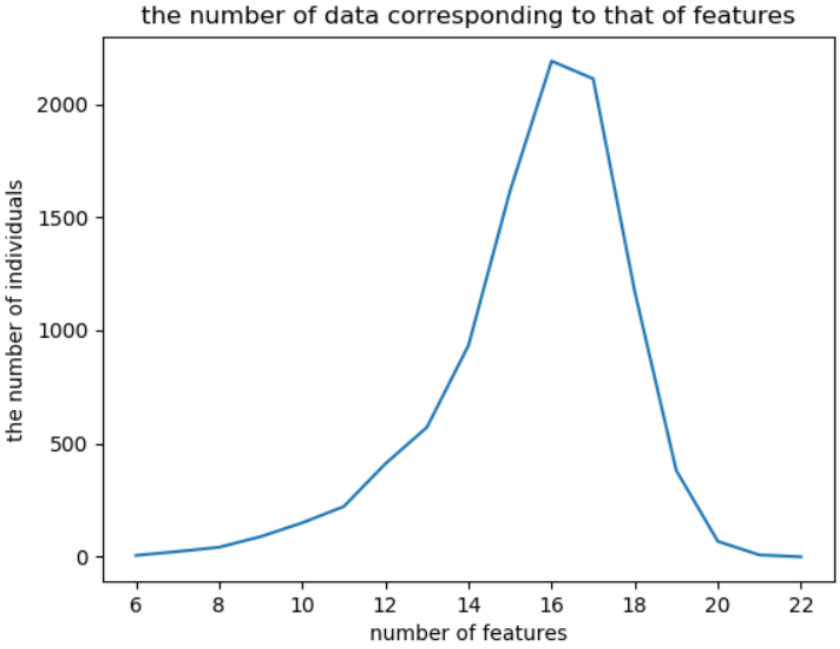

Fig.5. The number of data corresponding to that of features

From the median value of different number of features in the box diagram Figure 4, we can see that the accuracy increases with the number of features. However, for the purpose of minimizing the number of features, the first quartile and the third quartile of feature number 12 is obviously better than those of the data with feature number 11, while not significantly worse than those of 18 or 20. At the same time, the feature number 14 has the much better median value than 13 and not much worse than the ones with more features. Therefore, from this box diagram, decrease the feature number from 23 to 12 or 14 is desirable. From the Figure 5, we can see the most number of individuals have 12 to 19 features, so the individuals of 12 and 14 features have enough amount of data to support the results in Figure 4, which makes its data applaudable. Consequently, choosing an individual with 12 or 14 features is appropriate.

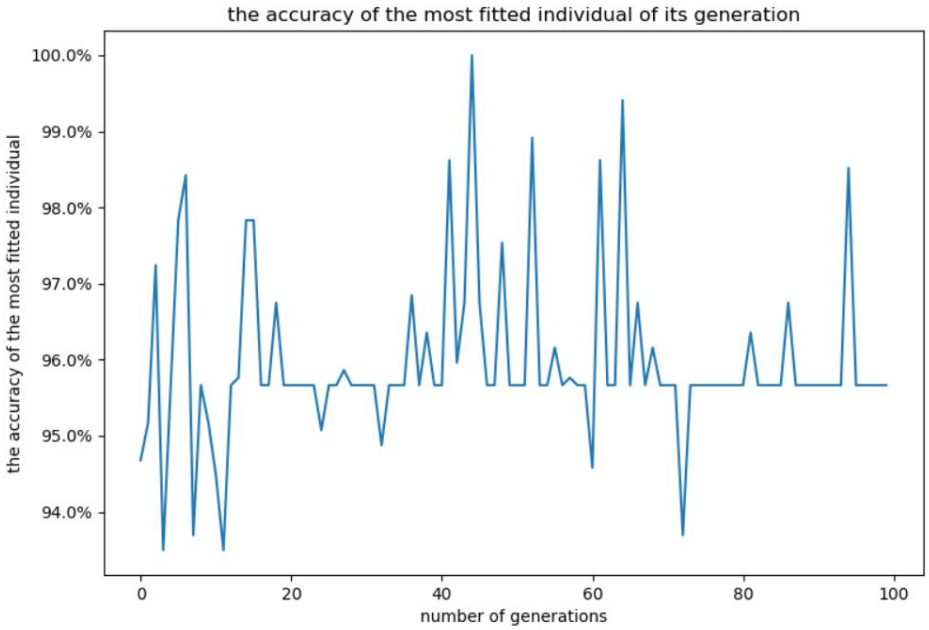

Fig.6. the accuracy of the most fitted individual in each generation

From figure 6, we can see that the best accuracy of each generation does not improve significantly after the 20th generation, and most of them have the best accuracy of 95.665024%. In other words, the result of this genetic algorithm converges before the generation 20. Therefore, in this scenario, the gene of individuals after 20 generation can be selected as the result of genetic algorithm. The reason of this maybe because that the number of population is large enough to cover most situations of the relative small size of gene 23 (which is the number of features as well), so more evolution makes no major improvements.

3.2 Classification using the output from the auto-decoder

The objective of this experiment is trying to use the blurring effect of data generated by the decoder as a substitution of the drop-out [2] method to extend the existing dataset and eliminate the over-fitting in the training process to achieve better accuracy. The dataset chosen is the "mushroom" dataset collected from UCI collection [3] which has 8124 instances that is specifically used for classification, with 22 features and 2 labels. For the purpose of comparison, another classification in the work of Hall and Simith [4] is also used to evaluate the classification results. In the preprocessing session, the dataset is first translated from the original character represented features to floating numbers ranging from 0 and 1. Then, they are clustered in mini-batch of size 30 to avoid sticking in local minimum value. After that, the training set are shuffled to avoid relying on a specific sequence. In this experiment, the decoded dataset is mixed with the original dataset, because if only decoded dataset exists, the over-fitting still can exist.

The number of training epoch is 20000, the learning rate is 0.001, the number of neurons in input layer, hidden layer, and output layer is 22, 18 and 22 respectively, the loss function and optimizer is MSE and Adam respectively. After the training process, the result (the blurred data with dimension of 8124 x 22) is combined with the original data (8124 x 22) as an input (with dimension of 16248 x 22) of the classification network. On the other hand, the classification network is trained 25 epochs using this combined data, other hyper-parameters are: the learning rate is 0.001, the number of neurons in input layer, hidden layer, and output layer is 22, 11 and 2 respectively, the loss function and optimizer is Cross Entropy and Adam respectively. After that, this classification network is trained another time with original data (8124 x 22) by 50 epochs for the sake of comparison. In both of these two experiments, 1/8 dataset is used as test dataset, the left 7/8 is used for training.

The result is shown in the Table 1:

Loss result of encoder-decoder	0.00033521
The accuracy of combined dataset	95.1747%
The accuracy of original dataset	85.0246%
The accuracy in the comparing pater	94.75%

Table 1. The comparison of accuracies of different groups

From this experiment, it is clear that the accuracy is increased more than 10%, which testifies the effectiveness of this method. Apart from this final result, other experiments were also done, the conclusion is that when the result of it is not large, the accuracy of combined dataset is better than the original one. However, when the loss value is too high, the accuracy of combined dataset declines instead. Therefore, the key point in this experiment is to find appropriate hyper-

parameter of encoder-decoder to generate a desirable improvement of classification. In contrast, in the work of Hall and Smith [4], the accuracy of the dataset mushroom is 94.75%, which is 0.4% lower than the combined dataset. However, other feature selection methods they use have higher accuracy. One method in their work is called Correlation-based Feature Selection (CFS) having the accuracy of 98.53%, and the other method called wrapper [10] achieves the even higher accuracy of 98.86%.

3.3 Classification using the results from the auto-decoder and feature selection

The objective of this experiment is to combine the results of two above approaches (3.1 and 3.2) to achieve better performance of the classification of neural network. The first step is the data preprocessing to get the results of the blurred data and selected features, which are the same with the two above experiments: the original dataset with full features are put into an auto-encoder to generate the blurred dataset. After that, this blurred dataset is combined with the original not blurred the dataset to create an extended dataset. Then, the features of combined data are selected by two selected subsets of features with 12 and 14 features (from the result of 3.1), which finishes the data preprocessing stage. Secondly, these data are put into classification neural network. The training process of classification network is the same as before, while the beginning and end of the training time is recorded to be compared with the training process of the original dataset to examine the improvement of training efficiency and accuracy.

The selections of features are from the result of genetic algorithm of section 3.1. The selection array of 12 features is [0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0], where each element represents the selection of its corresponding feature. “0” means deselect this feature, while “1” means select. The selection array of 14 features is [0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1].

For the sake of comparison, the classification network is trained four times as the ones shown in Table 2. The second, third and fourth networks (full features, 12 features and 14 features respectively) are trained 25 epochs while the first one is control group which is trained 50 times. This is because the first one uses the original dataset which is half the scale of others, whereas the second, third and fourth use the combination of original and blurred dataset which is twice the size of the first one, so the first one doubles its epoch number. Other hyper-parameters are: the learning rate is 0.001, the number of neurons in input layer, hidden layer, and output layer is 22, 2 and 2 respectively, the loss function and optimizer is Cross Entropy and Adam respectively. In these experiments, 1/8 dataset is used as test dataset, the left 7/8 is used for training.

Something noteworthy is that the number of neurons in the hidden layer is 2 which is fewer than that of above experiments, so the accuracy is lower than classification network of above experiments. The reason for this is because the focus of this experiment is on both accuracy and efficiency, so fewer hidden layer makes the training process faster.

The server I run this result has a memory of 64116 megabytes, with 12 CPUs (Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz), and one GPU (NVIDIA Corporation Device 1b06).

The result is shown in the Table 2:

	The accuracy	The time consumed
Control group	76.157635%	14.151923s
Full features of combined dataset	90.989660%	55.087041s
12 features of combined dataset	90.004923%	11.769714s
14 features of combined dataset	90.004923%	13.263888s

Table 2. The comparison of accuracies and time consumed of different groups

From the result, we can see that the features pruned groups (12 and 14 features) have the similar accuracy (less in 1%) with the one having the full features (22 features), while the time saved is significant, they just use 20% or so time of the one with the full features dataset. This means the feature selection of genetic algorithm has successfully chosen essential features and excludes the unnecessary ones. While the dataset of 12 features has the same accuracy with the one of 14 features, it saves almost 2 seconds, so the one with 12 features is a better choice. Something else noteworthy is that the results of 12 and 14 features groups are better than the control group which match the assumption before experiments. This means the improvement of efficiency and accuracy with auto-encoder and genetic algorithm is remarkable.

4 Future work

Due to the limitation of time in this assignment, not enough amount of experiments is made to fully testified the results. Therefore, more extensions can be made to thoroughly testify the conclusion of the whole experiment. For instance, in the encoder-decoder part, more optimizer can be tried to see if there are better choices and to find their merits and drawbacks.

For the classification network, more hyper-parameters should be tried to get more results to confirm the conclusion, and a critical value of the Loss result from encoder-decoder network could be found that combined dataset and original dataset getting the same accuracy. Moreover, in the combined dataset, the current proportion is 50-50, other percentage

of decoded data to original data should also be tried to get more results. Furthermore, the feature selection methods in the paper of Hall and Smith [4] are also worthy of trying to get better accuracy.

For the genetic algorithm, more choices of number of generation and population can be tried to get better choices of features that generate higher accuracy and save more training time of classification network. Other hyper-parameters of the genetic algorithm are also needed to have a try to fully explore its potential.

5 Conclusion

These experiments have successfully testified the effectiveness of the usage of auto-encoder-decoder and genetic algorithm in improving accuracy and efficiency of neural classification network.

In the part of testifying auto-encoder-decoder, the classification networks show desirable results. The method of using blurred data indeed creates better accuracy than the original data. The key point to get better result of classification is choosing a set of hyper-parameters that generates loss value which is not too large. Otherwise, the accuracy can be worse.

As for the part of genetic algorithm, the process of getting desirable result is long due to the amount of population and generation, so I use several measures to shorten the training time of classification network so that the getting fitness step, the most time-consuming step, can run faster. From the result of 100 population and 100 generation, I choose two candidates with 12 features and 14 features, for further experiments.

For the last and most important part, the selected features from the result of genetic algorithm is used to select features from the combination dataset, which is combined of the decoded blurred dataset and the original dataset. The result is desirable. Comparing to the control group, this method achieved higher accuracy with less training time. That is to say, it achieved higher accuracy and efficiency.

References

1. Gedeon, T. D., & Harris, D. (1992, June). Progressive image compression. In *Neural Networks, 1992. IJCNN., International Joint Conference on* (Vol. 4, pp. 403-407). IEEE.
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.
3. Merz, C. J. and Murphy, P. M. 1996. UCI Repository of Machine Learning Data-bases [http://archive.ics.uci.edu/ml/datasets/Mushroom]. Irvine Calif.: Univ. of Calif., Dept. of Information Science.
4. Hall, M. A., & Smith, L. A. (1999, May). Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper. In FLAIRS conference (Vol. 1999, pp. 235-239).
5. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
6. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
7. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.
8. Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.
9. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
10. Kohavi, R. (1995). Wrappers for performance enhancement and oblivious decision graphs. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE.
11. Goswami, S., Saha, S., Chakravorty, S., Chakrabarti, A., & Chakraborty, B. (2015). A new evaluation measure for feature subset selection with genetic algorithm. International Journal of Intelligent Systems and Applications, 7(10), 28-36. doi:http://dx.doi.org.virtual.anu.edu.au/10.5815/ijisa.2015.10.04
12. Dash, M., & Liu, H. (1997). Feature selection for classification. Intelligent data analysis, 1(3), 131-156.
13. Kramer, O. (2017). Genetic algorithms.Studies in Computational Intelligence, 679, 11-19. doi:10.1007/978-3-319-52156-5_2
14. Lin, C. H., Chen, H. Y., & Wu, Y. S. (2014). Study of image retrieval and classification based on adaptive features using genetic algorithm feature selection. Expert Systems with Applications, 41(15), 6611-6621.
15. Goswami, S., Saha, S., Chakravorty, S., Chakrabarti, A., & Chakraborty, B. (2015). A new evaluation measure for feature subset selection with genetic algorithm. International Journal of Intelligent Systems and Applications, 7(10), 28-36. doi:http://dx.doi.org.virtual.anu.edu.au/10.5815/ijisa.2015.10.04
16. Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. Physica A: Statistical Mechanics and its Applications, 391(6), 2193-2196.
17. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
18. Rubinstein, R. Y. (1997). Optimization of computer simulation models with rare events. European Journal of Operational Research, 99(1), 89-112.