Stress detection by compact neuron network: Genetic Algorithm applied for feature selection

Yiwei Li

Research School of Computer Science, The Australian National University

Canberra, Australia

∗E-mail: u7078773@anu.edu.au

Abstract. Artificial Neural Network is susceptible to inappropriate input feature selection. It is widely observed that irrelevant input features will cause Neural Network deterioration, since of this, input feature optimization is critical especially for a shallow Neural Network. This study will explore feature selection through a novel genetic algorithm, it will implement several techniques to diagnose the efficacy of predictor variables and choose an optimized feature combination for Artificial Neural Network. This study is implemented under PyTorch 1.7.1, our feature selection scheme remarkably improves the Neural Network's classification result with a k-fold accuracy rate of 99.65% which is higher than the original study. We will conceptualize the workflow from genetic algorithm feature selection toward artificial neural network training, it will also try to generalize the methodology which may be widely employed in high-dimensional neural network training.

Keywords: Feature selection, Genetic Algorithm, Artificial Neural network, K-fold validation

1 Introduction

Input data quality is critical for a successful artificial neural network training and validation, this challenge is especially outstanding when a shallow neural network, say two layers two-class classification, is to be trained from high dimensional input data. In this context, input data quality not only means the integrity of raw data but also the right combination of input features for neural network training. This is because irrelevant input features may degrade the neural network's performance and robustness. This phenomenon is emphasized by Gedeon et al. (2020) and Jerritta et al. (2011). In other words, training a shallow neural network through excessive input features may reduce the model's accuracy rate in the testing set.

This is a follow-on study based on our previous paper *Stress detection by compact neuron network: Hidden neuron similarly analysis and pruning*. In the previous study, we feed all 98 (after feature engineering) into a two-layer neural network which underperformed the EEG dataset collector (Gedeon et al., 2020) proposed model. In this study, we will try to apply a genetic algorithm to input feature selection. We will illustrate why we do so, how it works and analyze the possible reasons that lead to the outperformance compared to the original study.

Unlike our previous work, in this study, we do not implement neural network pruning through activation vector angle analysis, instead, we will tackle the problem from a different point: optimization of input feature selection to boost artificial neural network's performance measured by K-fold validation.

2 Starting point and Assumption

The original dataset is contributed by Gedeon et al. (2020), which consists of 144 cases with 210 different variables, we will continue with this dataset we used in the previous study. The original dataset is collected under a controlled lab environment, with thirteen males and eleven females, 12 different music clips were delivered to them during which time, a wide range of physiological signals were captured. The motivation of this dataset collection is to predict individual's mood from physiological records.

According to our previous analysis, the two-classes label, which represents calm and stress, is balanced with almost the same frequency. However, the challenge of this neural network data processing task is due to its large number of features provided. Given 144 records with 210 different predictor variables, it is inevitable that a properly design feature selection method should be applied before the neural network training, testing procedure. Here we provide several important observations about the

dataset, based on which we will raise a set of assumptions. This assumption will guide our algorithm development, workflow design and model training/validation.

2.1 Observation and Assumption

1. Two-layer shallow Neural Network can produce a reasonable accuracy rate in the testing set with partial feature input, this is observed through arbitrary feature selection during our previous study. However, we did not explore a systematic method to automate this process previously.
2. Previously, we applied several statistical analysis methods to normalize and regularize input features, which reduce them from 210 to 98. Once we feed these 98 different variables into our two-layer neural network, the model tends to overfit during training, with close to 100% accuracy rate, while it does not generalize well with about 88% accuracy rate in testing set.
3. ReLU tends to be a better activation function while we still go with the traditional Logistic Sigmoid function in our previous study. Adam is the preferred optimization algorithm that notably improved both convergency speed and accuracy rate.
4. Dataset contributor Gedeon et al. (2020) mentioned the genetic algorithm for feature selection in their original study, however, there are no technical details revealed by them.

Given these observations and clues we proposed the following assumptions before we bring forward our study which is developed from these assumptions:

1. For this specific EEG dataset, a two-layer artificial neural network is sufficient for high accuracy rate in validation, this trend is observed by our previous study and supported by the original paper which proposes two-layer 15 hidden neural networks.
2. Given a set of input features, the size of neural network only influences its convergence rate. In short, convergency is monotonical no matter the size of neural network.
3. Given a set of input features, number of epochs for training only influences its final performance. In other words, convergency is monotonical no matter the number of training loop.

Under assumption 1 and observation 1, we will design our evolutionary framework and final neural network training/testing under a two-layer shallow neural network structure. There will be some twists and modifications shall we point out later.
Taken assumptions 2, 3 and observation 2 into consideration, we will introduce parsimonious feature selection through a genetic algorithm. This is to say, the original dataset provided excessive features if our task is only to produce a two classes classification. We believe parsimonious input features could not only hugely speed up genetic algorithms but also contribute to a well-generalized artificial neural network under this dataset.
Observation 4 along with our previous study will work as a baseline, we will try to reach or even overpass previous studies' accuracy rate in k-fold validation. We will also incorporate observation 3 into our model building and activation function selection for both fitness calculation and neural network training.
To keep our analysis concise and straightforward, we will not repeat the data exploration analysis part which is covered in our previous study. It is also worth pointing out, in our previous study, to carry out hidden neuron distinctiveness analysis, we embedded several diagnostic codes in PyTorch's neural network specification. This practice is a heavy drag for model training, given our intention in this study, we removed the distinctiveness analysis function and focus on genetic algorithms for feature selection. Original code is available upon request or through the website. New code for this paper is also provided through the same source.

3 Method and Work Flow

In this study, we proposed a highly integrated workflow that consists of three parts: Genetic algorithm for feature selection, artificial neural network training, modified K-fold validation. These three parts work together in sequence with output from previous parts act as input in the following parts. To clarify our methodology, we will conceptualize several techniques which are stem from mainstream methods with critical and fit-for usage modifications.

3.1 Genetic Algorithm for optimized feature selection

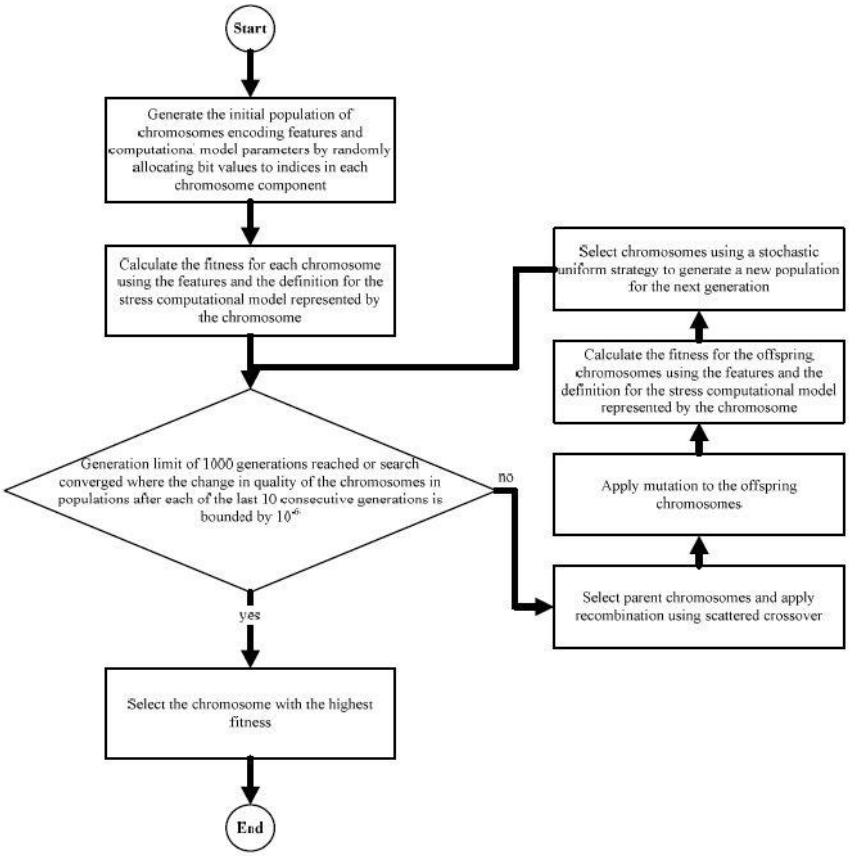
According to our previous study and other empirical research (Gedeon et al., 2020), we assume a two-layer neural network is adequate for the EEG dataset classification task. Our focus is to apply a Genetic Algorithm as an overlay on the top of an artificial neural network. A genetic algorithm by its nature is a heuristic selection method that must rest on a solid foundation.

This means we need a fitness measure to drive the Genetic process, in this study, this fitness measure is a specified small neural network. Furthermore, the genetic algorithm can work as a stand-alone tool, for example in the classic travel salesman NP-difficult problem the output of the genetic algorithm is the optimum solution, however, this is often not the case. As we will point out later, the purpose of genetic algorithm for feature selection is to pass the selected predictor variables to a big neural network for in-depth training and validation.

Some may argue that an artificial neural network can be optimized by another neural network, this idea is to apply a separated optimization neural network to improve the structure or feature selection of the target neural network. This idea may be simple for its implementation but will make the interpretation more difficult, the reason is, neural network by its nature is relatively opaque, overlay a neural network upon another may reduce the interpretability and make the diagnostic process even more complex. Considering this, in this study we will apply a genetic algorithm for input feature selection.

Our genetic algorithm is based on the framework proposed by N. Sharma and T. Gedeon (2013) as following.

Figure 1 EA overlay on Artificial Neural Network structure (N. Sharma & T. Gedeon, 2013)



Our Genetic Algorithm works as an overlay on the top of the Artificial Neural Network which is largely inspired by the left original structure raised by N. Sharma and T. Gedeon (2013). Based on their initialization, we made several modifications and introduced some techniques that will notably reduce the computational cost, our model also made some necessary trad-off between specialization and generalization which may promote its application in other areas.

- Unlike the original paper which calculate fitness score based on a stress model, we calculate fitness score by feed randomly encoded features to a small but good enough neural network, this small NN's accurate rate in the testing phase will work as our fitness value.
- Original study proposes a 1000-generation iteration which is computationally inefficient. We believe by replacing the stress calculation model with our small Neural Network, we can generate sufficient convergence within 100 generations.
- We will implement two different crossover methods, which are one-point crossover and uniform crossover, we will demonstrate the merits of each.
- We refined the selection phase, we will argue that by integrating proportional selection, top-k selection and tournament selection, the output feature set will be more reliable it will also enhance model stability.

Our selection phase composed of three steps:

Figure 2 Three steps selection under Genetic Algorithm overlay Strategy

Proportional selection: train a small neural network like usual, calculate accuracy rate in testing set which is translated into a non-negative proportional selection score. Parent selection is based on this proportional score.

Top-K seed player: keep the population size the same during the evolutionary process, choose the top-k individual in the final population with the highest fitness score as seed player for final selection.

Tournament: Top-K seed player will enter into a multiple turn, environment varying but controlled tournament, say 20 turns, individual with the highest average accuracy rate in the testing set during this 20-run tournament is our champion, this champion's features will be passed on for following big neural network training and testing.

Through this three-step selection, our study finds out the output features can markedly improve Artificial Neural Network's performance during both training and testing. NN's prediction will also become more stable which adds to its credibility.

There are four concepts that are important for our study and critical for its success which worth a few more words:

- Environment control: In this study, we propose the new concept which functions very much like a traditional k-fold validation. The idea is, during each loop in both genetic algorithm and following big neural network training/testing,

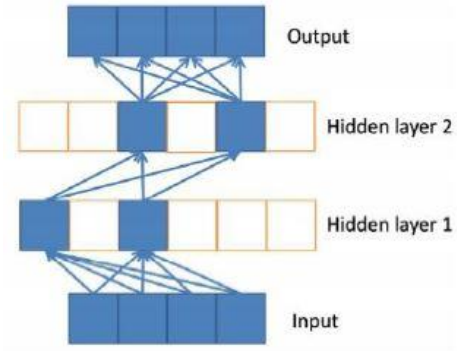
we will shuffle the prepossessed data and randomly apply an 80/20 split on it for training or fitness calculation. This will expand our ability to squeeze the most out from the small dataset, it provides us with enormous flexibility to treat the input data because each time the environment is new and the model is trained from zero, we can go through this process arbitrary time without concern of over-fitting and avoid under-fitting, it also rules out pure luck due to randomness especially for a small dataset like this one.

- Parsimonious feature selection: As we mention before, it seems like in this two-class classification task, only a small bunch of input features materially contribution meaningful prediction. Irrelevant features may damage neural network's performance. In order to realize this concept, during our population initialization, we act very conservatively on genotype generation, every single gene has only 10% change to be encoded as 1 which stands for presence. Our experiment proved this is a critical step for its success. Parsimonious also means a relatively low mutation rate which has a default value of 0.001 in our study.
- Elitism Protection: Recent studies have widely proposed that elitism can accelerate genetic algorithm's convergency, in our study we implement vanilla evolution which means no elitism protection, we also test the evolution with elitism protection in which, best performing k individual's DNA is passed to next population without crossover and mutation. The heuristic is that: excellent parent some time produce bad child, given them are good parents in this time, they desire a second chance.
- Population's behavior is valuable: We believe evolution is very much a collective phenomenon which means evolution appears in the whole population. Based on this concept, in our genetic algorithm, we will monitor not only the best individual's performance but also the population's average performance. The logic is: a champion is more likely to be a real champion if it is out of a population that is already pretty good.

3.2 Big Artificial Neural Network based on EA's feature selection

After the three-step selection (Proportional selection, Top-k, Tournament), the champion produced by our Genetic Algorithm will guide our big model building. This is the reason we call it a Genetic Algorithm overlay on Artificial Neural Networks for feature selection. We need to point out, although we choose a small neural network as our fitness calculation function during the EA phase, this small NN does not directly modify our final big NN's structure, instead, we will build the big NN from the champion produced after the tournament from zero. This process will avoid overfitting and guarantee a robust generalization. The most significant improvement in this study is that we replaced the Logistic Sigmoid function with ReLU in both the Genetic algorithm and big NN training/testing phase. ReLU improves the accuracy, provide stabilization, and accelerate convergency.

Figure 3 ReLU provide obvious advantage in both EA and NN training Source: Nathan Elazar's Lecture materials



These advantages provided by ReLU probability stems from its sparse feed-forward behavior and constant deviation characteristic. ReLU only passes on weighted input values when it greater than zero, which means the network is sparse, these features contribute to a much faster calculation. Furthermore, ReLU have a constant first order derivate which means learning is more efficient without knowledge loss.

In our study, both the small NN used in EA and big NN employed in final model building are two-layer feed-forward fully connected Neural Network, with one hidden layer populated with 5 neurons driven by ReLU. Empirical study shows this structure works well for both the Genetic Algorithm's fitness function and the big NN model. One may state that given 210 raw input, 5 hidden neuron seems too small, our argument is, given we already find out many inputs does not contribute positively toward high accuracy rate, there is no need to use all of them. After statistical normalization, 210 raw inputs have been reduced into 98 preprocessed features. Furthermore, in our genotype initialization step, each gene (single feature) has a 10% chance to be encoded as 1, this means for these 98 different features, individual will on average possesses 10 features. This means 5 hidden neurons is a reasonable choice.

3.3 Environment controls replace K-fold validation

As we mentioned previously, in this study, we introduce a new model validation concept as environment control rather than traditional one-shot 80/20 split, K-Fold validation, or Leave One Out. This modification is to accommodate the small nature of the input data.

This EEG dataset consists of only 144 patterns, one-shot 80/20 split is not strong enough to produce a reasonable test result, since on average only 29 cases will be chosen as a testing set.

K-Fold validation can provide a better model performance estimation, but the issues is if the K is too small, it does not help a lot given the dataset itself is very small, if K is large, say 10-fold validation, each partition of the 10-fold will only contain about 14 cases, which means randomness will play a big role in validation, a single partition of K-fold may condense in outlier and damage evaluation.

Leave One Out is computationally costly, especially for fitness function which is chosen as a Neural Network.

Given these considerations, we propose this Environment Controlment experiment concept, which is implemented in both Tournament and final big Neural Network building process. The basic idea is, when we want to compare performances among several neural networks, or when we want to validate the performance of our final model, we first shuffle the input data, we apply a traditional 80/20 split training/testing partition N times. During each time, we retrain our models from zero under the same hyperparameter setting, compare their average accuracy rate.
Advantage of Environment Controlment validation:

- Statistically reliable given a reasonably adequate iteration.
- More flexible than K-Fold validation, we can run arbitrary times as we like (in our study we run it 10 times for top-5 Tournament and 20 times for final model validation). 20-run Environment Controlment guaranteed the comparison among different NNs is fair, it also on average give every data point at least one chance to participate into both training and testing phase.
- Computational cost is less than leave one out.
- The logic behind our Environment Controlment is very similar to Bagging for Random Forest, instead of applying randomly generated dataset to decision tree, we apply it for Genetic Algorithm and Neural Network, our study show some merits provided by this process.

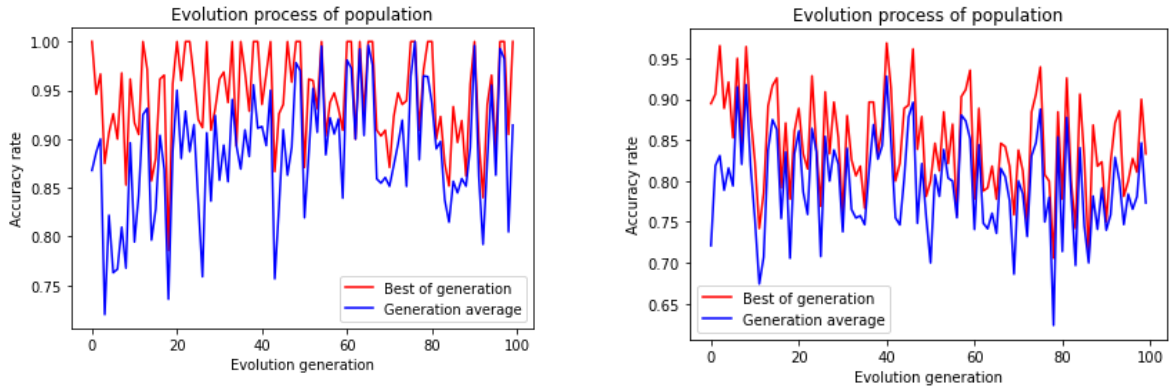
Disadvantage:

- This method is not widely used among Neural Network for the information processing community, its generalization still calls for further study.
- It reduces computational cost with I/O scan cost, our algorithm currently does not provide significant cost reduction in terms of running time.

4 Result and Discussion

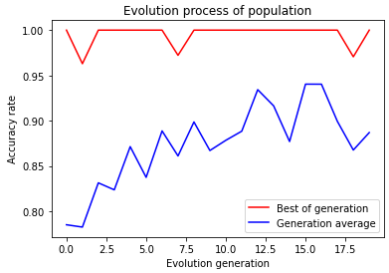
In this part, we will first provide several intermedial results which guide our hyperparameters setting for both the Genetic Algorithm and final big Neural Network training/testing. We will also carry out a comparison study to demonstrate the merits of our model building structure and its potential employment in relevant areas.
Through our study, we find out that, the one-point crossover does not perform well for this given dataset, population size is critical for the Genetic algorithm's performance. Without elitism protection mechanism evolutionary process is too wild to be useful.

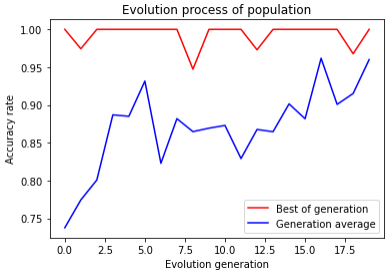
Figure 4 Evolution Process results: One-point crossover with population size of 10 without Elitism protection



The left graph in figure 4 is trained with ReLU, right graph is Trained with Sigmoid. Both are trained under one-point crossover, population size 10, 100 generations without elitism protection. As we can see, the performance of best individual and population is very wild, although accuracy rate generates rest on a high-level, around 85 %, the performance fluctuates widely through generations and can fall sharply which means evolution in individual-level and population-level is not stable enough to provide a meaningful result. ReLU may be marginally better under this setting but still not strong enough to produce a champion as our final feature selection.
We now try to increase population size from 10 to 100, by doing so genetic algorithm's stability improved a lot.

Figure 5 Evolution Process results: One-point crossover, population size of 100 without Elitism protection

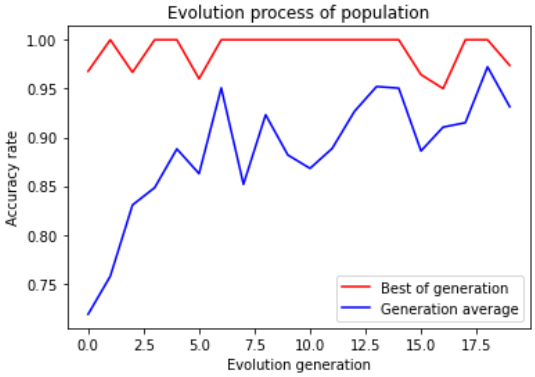




The left figure in figure 5 is the evolution record under ReLU as activation function and the right graph record performance of Sigmoid. As we can see, when we increase population size from 10 to 100, both the best individual within each generation and population across generations can steadily improve their fitness score which is represented by accuracy rate. Increasing population size is very costly in terms of the algorithm running time, this is the reason we choose 20 generations evolution loop to demonstrate the improvement in stability provided by a larger population size. We can tell from both graphs that, larger population size benefits both best individual and population's average performance. The best individual is represented by the red line, which is stable and stays close to a very high accuracy rate, fluctuation is vastly reduced which means a greater population size produces better individuals. It is also worth pointing out that, population's average performance is also steadily improved and swinged much less than before. This phenomenon proved our hypothesis raised before that: evolution is largely a collective behavior, and greater gene diversity provided by increased population size help to bring out a stronger individual who tends to perform much more stable in various environments.

We now turn our attention to exploring whether uniform crossover can provide better evolutionary results than the one-point crossover.

Figure 6 Evolution Process results: ReLU, population size 100, generation 10



Here we apply uniform crossover to a population of size 100, iterate 20 generations under ReLU as the activation function. We found that uniform crossover does not significantly improve the best individual's performance, this is because the best individual does not have much space to improve. However, we can see that the population as a group performs much better on average compared to one-point crossover. This phenomenon has two meaningful indication:

- Uniform crossover helps the population converge to higher lever accuracy much faster. As we can see from figure 6, the gap between the red line and blue shrinks very fast and closer than before. This means the gap between best individual and generation average is smaller. We would argue that a number one is more likely to be a true number one if its peers are also pretty good, this means the uniform crossover provides some necessary creditability to the best individual's ability when applying it into a new environment. From a statistical viewpoint, this means the standard deviation of the population is largely reduce which will increase the significant level of the individual's creditability.
- Given the right population size, the population's average accuracy rate can converge to above 90% level which already beat our previous study. This gives us confidence and guideline for other parameters setting since we can set generation number at a reasonable low but sufficient level that allows us to deploy more time and computational budget to other areas.

We can demonstrate that generation number and population size do not need to be too large by figure 7, which is generated under 500 population size, 100 generations. As we can see, the pattern of population and best within each generation do not change too much, population average converge to its upper bound just below 90% accuracy rate within 20 generations, after which, both individual and population stays where they are and do not fluctuate too much. We need to point out that given population size increased to 500 there is no double that the population average will rest on a relatively lower level compared with population size of 100. This is because, when population size goes up, the chance bad individuals within each population will increase which will

Figure 7 Evolution Process results: ReLU, population size 500, generation 100

that

pull down the population average. However, the population as a whole still stale relative stable in about 87% accuracy rate which is a promising signal. It is also worth pointing out, that the best of the generation's performance became even more stable with most of the time, their accuracy rate stays at almost 100% level.
Given these graphic diagnostic results, we will continue our genetic algorithm experiment will the following principles:

1. Population size and generation numbers do not need to be overly big. A reasonable setting is mathematically adequate and will improve computational efficiency.
2. ReLU activation function provides marginal benefit in computation efficiency, model stability and convergence rate.
3. Big population size in prohibitively cost, for example, in figure 7, 500 population size in 100 generations take around 45 minutes to compute which is carried under Windows 10, i7 CPU and NVIDIA GeForce GTX 1650.

Given these considerations we our fined tuned final input as following:
Population size: 20, generation: 50
Probability each gene encoded as true: 10%, Mutation rate: 0.001
Learning rate for Genetic algorithm's small Neural Network: 0.005
Number of epochs for Genetic algorithm's Neural Network: 100
Tournament running turn: 10 times, Top-K individual participate into Tournament: 5
Learning rate for final champion Neural Network training: 0.008
Number of epochs for champion Neural Network training: 500
Environment Controlment validation for Champion individual: 20 times
Activation function: ReLU, Optimization algorithm: Adam

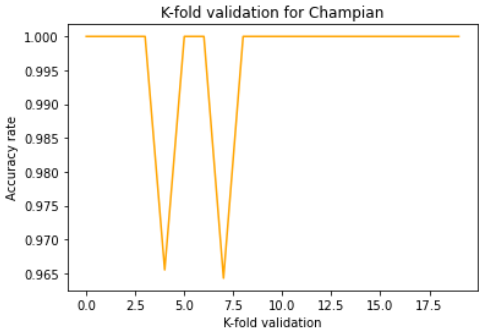
Experiment environment: Window 10 operation system, i7 CPU, NVIDIA GeForce GTX 1650
Under the given experiment environment and hyperparameters setting, our Evolution Algorithm overlay upon Artificial Neural Network reached the highest average accuracy as 99.65% in the 20-run environment-controlled validation. It turns out this Champion's features consist of 12 features. Computation is completed within 1.5 minutes.
Table 1 Summary of model performance

ANN(Previous study)	GA + ANN	SVM	GA + SVM	GA + ANN (Current)
88.62%	82%	67%	98%	99.65%

ANN (Previous study): preprocessed input all feed into two-layer Neural Network with 15 hidden units
GA + ANN, SVM and GA + SVM are reported by T. Gedeon et al. (2020)
GA + ANN (Current): 20-run Environment controlment average under our proposed GA feature selection overlay upon ANN.
It should be pointed out, due to randomness in population initialization, the final output does swing within a narrow range, 99.65% is the highest possible outcome under our experiment, overall performance almost all fall above 95% level. Considering our report is an average value under a 20-run environment-controlled validation, we believe this result is statistically reliable.
There are sever interesting points:

- Under most experiments, 'mean_first_diff' and 'sencod_second_diff' are present in champion's features, this is supported by T. Gedeon et al. (2020), according to their study, change in an individual's physiology signal is more useful in the prediction of a specific individual's mood.
- Besides the two 'diff' measures, other engineered features like fuzzy measurement and hjorth parameter, which is a measurement of individual movement, also tend to be components of the final champion's features. This observation may suggest that part of our success is based on properly engineered features and necessary data preprocessing.
- Lage features input does not necessarily increase the final model's accuracy, our champion individual consists of only 12 features, in some cases, 9 features as input can also achieve above 98% accuracy rate.

Figure 8 GA + ANN evolutionary process monitoring and Champion individual's performance



As we can see this specific population that breeds our champion individual follows a healthy evolutionary pattern, the population average accurate rate steadily increases from below 60% level toward 90% with limited fluctuation. Champion individual's performance is excellent and stays close to 100% during the 20-run validation testing phase, its accuracy rate remains in the range of [96.5%, 100%] which demonstrates its ability to precisely predict unseen data.

We believe there still have some space for our GA + ANN structure to grow, given our computational constraints and limited time budgets, we specified our structure in a parsimonious way with a low gene encoding rate, low mutation rate, reasonably small population size and generation numbers. Our current setting can complete within 1-2 minutes for an up-to-date individual commercial computer. If we have more computation budgets, we can specify a bigger population and generation number which may further boost the champion's performance.

During our monitoring of computer resource during Genetic Algorithm evolution and Model training, we find out that our GA + ANN structure does not consume unreasonable CPU computation ability, its requirement on GPU and Main memory were also very low, the main reason it will take quite a long time to go through all procedures is that we introduce many data structures in our program. In the code accompanied with this paper, we applied Python build-in dictionary, list, numpy.ndarray, pandas dataframe PyTorch tensor. Many of our functions require frequent interaction between different data types and data structure, this is a heavy drag in terms of computation time.

This issue is a general challenge for Genetic Algorithms and relevant evolutionary coding, they are flexible and simple in theory but can be cost in computation. The lack of a proven upper bound on computation also means in practice, early stop criterion often never hit, instead, a properly chosen iteration level is usually used as a replacement.

6 Conclusion and future work

This study is an expansion from our previous research, the genetic algorithm for feature selection is inspired by EEG dataset contributor's original work. Our study is based on secondary data provided by the origin study, an important part of our success should be attributed to high-quality raw data and properly engineered features. Unlike the original study which is greedy on Neural Network design with 15 hidden units and input much more predictor variables for classification. Our GA + ANN follows a different philosophy, it is implemented through a genetic algorithm under parsimonious gene encoding, conservative mutation rate, three-step selection and modified k-fold validation. By bring these concepts together, our model advanced to a higher accuracy rate on average, this is realized under a much simpler final Artificial Neural Network with around 10 different features, two-layer ReLU activated structure with 5 hidden neurons.

Through our study, we find that uniform crossover generally works better than the one-point crossover, which stabilizes the population's performance and accelerates convergence. Proportional selection, top-k selection and tournament are not exclusive to each other, they can work together and breed stronger individuals. In general, the proportional selection is suitable for parent selection based on their normalized fitness values, after reasonable generations, top-k selection can provide a seed player pool for the final tournament. We found out that a champion produced in this way can excel in various environments.

We also introduced a novel concept which is an extension of k-fold validation and share some common logic with bagging for random forest. Environment-controlled validation is implemented in both tournament and champion model training and testing, this data processing and model training method provides us with the necessary flexibility in both individual selection and model validation.

Our study does not close the book but draws forth several future explorations.

First and foremost, our study incorporates population behavior into the genetic algorithm's diagnosis and monitor, evolutionary process follows a very complex dynamic process that is very hard to formulate. Our study, by its empirical nature, assumes genetic algorithm is monotonic and will finally converge toward its optimal solution. However, given the high-dimensional nature of this EEG dataset, randomness in population initialization still has a notable influence on feature selection. Engineered features tend to be chosen but this does not guarantee the champion individual is the global optimization. How to verify genetic algorithm's performance is still a challenge.

Secondly, we believe there should be smarter ways to utilize the population's intermediate output. In our study, we record and plot out the population's average accuracy rate, applied top-k selection on the final population. Future work could come from introducing mechanisms that can trigger early stop for genetic algorithm, the reason is that it should be possible to tell whether an immature horse can be a champion before it does become a champion. Statistical analysis for the population may help us achieve this target in the future.

Finally, our experiment is implemented under Python and PyTorch, multiple data structures and data type is utilized which reduced the computation efficiency, by improving GA code through on-the-fly optimization, the genetic algorithm can be generalized for wider application.

Reference

1. Gedeon, T.D., 1995, November. Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. In *Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems* (pp. 26-29). IEEE.
2. Gedeon, T.D. and Harris, D., 1991. Network reduction techniques. In *Proceedings International Conference on Neural Networks Methodologies and Applications* (Vol. 1, pp. 119-126).
3. Rahman, J.S., Gedeon, T., Caldwell, S., Jones, R. and Jin, Z., 2021. Towards Effective Music Therapy for Mental Health Care Using Machine Learning Tools: Human Affective Reasoning and Music Genres. *Journal of Artificial Intelligence and Soft Computing Research*, *11*(1), pp.5-20.
4. Irani, R., Nasrollahi, K., Dhall, A., Moeslund, T.B. and Gedeon, T., 2016, December. Thermal super-pixels for bimodal stress recognition. In *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)* (pp. 1-6). IEEE.
5. Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
6. Xu, B., Wang, N., Chen, T. and Li, M., 2015. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.
7. Madhiarasan, M. and Deepa, S.N., 2016. A novel criterion to select hidden neuron numbers in improved back propagation networks for wind speed forecasting. *Applied intelligence*, *44*(4), pp.878-893.
8. Jerritta, S., Murugappan, M., Nagarajan, R. and Wan, K., 2011, March. Physiological signals based human emotion recognition: a review. In *2011 IEEE 7th International Colloquium on Signal Processing and its Applications* (pp. 410-415). IEEE.
9. Mozer, M.C. and Smolensky, P., 1989. Using relevance to reduce network size automatically. *Connection Science*, *1*(1), pp.3-16.
10. Rumelhart, DE, "Learning and Generalisation," *Proc. IEEE Int. Con$* on *Neural Networks*, San Diego, 1988.
11. Sanger, D., 1989. Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. *Connection Science*, *1*(2), pp.115-138.
12. Han, J., Kamber, M. and Pei, J., 2011. Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems*, *5*(4), pp.83-124.
13. Sharma, N. and Gedeon, T., 2013, April. Hybrid genetic algorithms for stress recognition in reading. In *European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics* (pp. 117-128). Springer, Berlin, Heidelberg.
14. Eiben, A.E. and Smith, J.E., 2003. *Introduction to evolutionary computing* (Vol. 53, p. 18). Berlin: springer.
15. Model's accuracy rate code and its' print-out format is modified from COMP8420 lab materials based on lab2, lab3 and lab4.
16. Mutation function in Python code is inspired by COMP8420 lab7 with significant modification.