An Improved Weight Matrix Indicator of Hidden Neuron Functionality: Analyze Based on Genetic Algorithm Optimized Music Affect Brainwave Classification Neural Network

Yiran Wang

Research School of Computer Science, Australian National University u7152008@anu.edu.au

Abstract. Music is closely related to our life and can affect our moods and feeling. This paper uses one channel of the EEG data, generated while the participant is listening to music[9], to train a classification neural network. The feature and hyperparameter selection were made by using the genetic algorithm. The trained classifier performs poorly on the testing set, far from the original paper, indicating that one channel of the EEG data cannot reveal the type of music participants are listening to.
Pruning redundant neurons can help decide the number of neurons in the hidden layer of a neural network, which is very important for today's increasingly deep neural networks since it is challenging to decide the number of hidden neurons by human experimentation. Also, pruning the branches helps generalize the neural network. We want to show that the weight matrix in [4] can be used as an indicator of the function of hidden neurons. In the process of trying this, we found that changing the normalization method can make the weight matrix perform better. We perform experiments on the music effect dataset, pruning bit by bit and calculating the minimum angle between the weight matrix and neuron behavior separately. The experiments show that the improved weight matrix performs much better than the original weight matrix and can approximate the neuron behavior indicator.

Keywords: Neural network · Neuron pruning · Classification · Electroencephalogram · Genetic Algorithm

1 Introduction

Nowadays, the application of neural networks is becoming more and more widespread. For example, deep learning, the most cutting-edge neural network, is utilized in many fields such as speech recognition[2], language translation[1], and autonomous driving[6]. Although there are very powerful GPUs and other processors that can help us apply this technology to large data sets now, it is a highly complex task to manually set up and try to find the number of neurons for each layer[11]. Therefore, pruning is essential. Pruning not only helps to decide how many neurons are appropriate for the current network, which can reduce redundancy, save computational time and memory cost, but also increases the generalization of the neural network, helping it not overfit the training and to perform well on the testing set.

To better perform pruning, we aim to find a good indicator of hidden neuron functionality. According to [4], they built a weight matrix indicator, but it failed to perform well on their dataset. In this paper, we test this weight matrix indicator again on a different task and propose an improved weight matrix that approximates the neuron behavior indicator mentioned by [4] as a benchmark.

Music is an important part of our daily life. We always feel that listening to different kinds of music can elicit different kinds of emotions. It was proved that music could improve our mood and decrease our anxiety[8]. Relaxing classical music can help us sleep better[7]. In the paper [9], they show that we can use the EEG brainwave signal to identify which category of music people are listening to. The electroencephalogram (EEG) is a commonly used signal for people to understand brain activity[9]. EEG signal can be divided into Delta(δ), Theta(θ), Alpha(α), Beta(β), and Gamma(γ) five frequency bands. Each wave represents a different state of people. In this paper, we attempt to use only one channel of EEG data in [9] to build a neural network classifier, which classifies which kind of music the participants listen to.

Using high dimension features can bring adverse effects to the classification if some features are irrelevant or redundant.[13] The high dimensionality of the data also increases the computation time. Feature selection can solve those problems by removing unnecessary features and reduces the dimension of input data. Hyperparameter tuning is also vital for the functionality and accuracy of neural networks.[12] Selecting the best hyperparameters is one of the most major challenges for neural network design. Genetic algorithm is found effective in both feature and hyperparameter selection.[13][12] Therefore, in this paper, we combine these two types of things that need to be selected, using a genetic algorithm to accomplish selection to get the best input and model setting.

Our classifying experiment is a failure even with feature and hyperparameter selection. Only one channel of EEG data cannot train a good classifier. However, since it is a well-trained neural network, this classifier can still be used to test the hidden neuron function indicator. We use this network to perform pruning and analyze indicators of hidden neuron functionality.

The paper is organized as follows: Section 2 describes the dataset and the methods we used to complete our study. In section 3, we explain the results and discuss the cause of those results. Section 4 presents a conclusion and describes the possible future work to dig deeper.

2 Materials and Methods

2.1 Datasets

The dataset we used is the brain wave data captured by EEG, which was used to analyze the effects of different kinds of music[9]. There were 24 participants in their experiment. Each participant was asked to listen to two out of three categories of music: classical, instrumental, and pop[10]. The total music categories are balanced among all participants. They collected participants' brain waves through Emotiv EPOC headset. The Emotiv EPOC provided them 14 channels of data, and the sampling rate they used was 128 Hz. Our experiment only use one of those channels, the F7 channel, which was one of the most informative channels according to their description. They filtered the raw signal by median smoothing filter to deduct noise and then applied band pass filtering to keep alpha, beta, and gamma waves they were interested in. After that, each person's data was processed into 12 patterns per category; each pattern consists of 25 linear and non-linear features they extracted from the waves. Table 1 shows those features. In conclusion, the dataset we used is a subset of them. It only contains the F7 channel's 25 features from 24 participants, labeled with the category of music they were listening to.

Table 1. Feature types and names[9]

Feature type	Feature names
Linear	Mean, Maximum, Minimum, Standard Deviation, Interquartile Range, Sum, Variance, Skewness, Kurtosis, Root Mean Square, Average of the power of signals, Peaks in Periodic Signals, Integrated Signals, Simple Square Integral, Means of the absolute values of the first and second differences, Log Detector, Average Amplitude Change, Difference Absolute Standard Deviation Value
Non-Linear	Detrended Fluctuation Analysis, Approximate Entropy, Fuzzy Entropy, Shannon's Entropy, Permutation Entropy, Hjorth Parameters, Hurst Exponent

2.2 Prepocessing

Due to the large number of features, we only present the descriptive statistics of a few representative features in Table 2. The complete statistics for all features are shown in the Appendix. Through Table 2 we can observe that there are outliers in the fuzzy entropy data. The 75th percentile of all fuzzy entropy data is only 1.7471, but the maximum is 65535, which is equal to $2^{16} - 1$ a common error value. Therefore, we eliminate the pattern with fuzzy entropy of 65535 in preprocessing. We did not observe this problem in other features.

Table 2. Descriptive statistic of some features in the original data

	mean_F7	max_F7	min_F7	std_F7	sum_F7	**fuzzy_F7**	shannon_F7	pe_F7	hurst_F7
mean	1.8191	6.3541	0.2438	1.9005	24.8340	**457.7391**	0.9390	0.4759	0.6917
std	2.5700	9.7837	0.7636	2.9311	37.0508	**5,446.7663**	0.2530	0.2042	0.2865
min	0.1772	0.3311	0.0000	0.0581	1.3113	**-0.0909**	0.2338	0.0000	0.1354
25%	0.4897	1.2277	0.0057	0.3637	6.0505	**0.5054**	0.6886	0.2181	0.4910
50%	0.8148	2.0145	0.0153	0.5800	9.7277	**0.9074**	0.9503	0.5812	0.6031
75%	1.6324	6.1330	0.2309	1.8853	23.7085	**1.7471**	1.1034	0.6373	0.8874
max	21.3101	49.6533	14.5919	16.2437	230.5620	**65,535.0000**	1.3705	0.7245	1.5707

After excluding the outliers, we randomly divide about 80 percent of the data into the training set and the rest of them into the testing set. The data from the training set is first used in the genetic algorithm to obtain

the optimal feature selection and optimal neural network hyperparameters, then used in the final neural network training process. The test set data is only used to test the final neural network. During the genetic algorithm, for each evaluation of each individual, we further split the training set from the above step into 80% for the training set and 20% for the validation set. More details are included in section 2.5. In all dividing procedures, we artificially let data from one person be in the same set to make sure our results are as objective as possible.

The different feature contains different information; they may be all important for classifying. However, the difference of each feature's range has a great impact on how well the neural network classifies. As we can see in Table 2, the scopes of our features are varied. For example, the maximum of permutation entropy(pe) is less than the minimum of sum, but that should not mean the permutation entropy is less important. Therefore, after each data division, we use the Min-Max normalization method to bring all features in the training set within the range 0-1 to avoid some features dominating the classification process. We normalize all features respectively and then use those parameters, the min and max, to normalize the testing set (validation set).

2.3 Classifier

Same as the original paper of this music affects dataset[9], we use a simple neural network with one input layer, one hidden layer, and one output layer. A sigmoid activation function is used in the hidden layer since it is not a deep network. We choose cross-entropy loss as our loss function and use Adam optimizer to perform backpropagation. The hyperparameters and the feature we finally selected are gained using the genetic algorithm.

2.4 Evaluation Measures

We use accuracy, precision, recall, specificity, and F-measure to evaluate our classification result comprehensively. Accuracy describes the percentage of correctly classified data out of the total data. Recall represents how many positive samples are correctly classified. Specificity is the ratio that a negative sample is correctly classified. F-measure is calculated by the harmonic mean of recall and precision. It is another kind of accuracy to assess the goodness of the classifier.[14] Their equations are shown below:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \tag{1}$$

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{2}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3}$$

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive} \tag{4}$$

$$F - measure = \frac{2 \times TruePositive}{2 \times TruePositive + FalsePositive + FalseNegative} \tag{5}$$

2.5 Feature and Neural Network Hyperparameter Selection

We use the genetic algorithm to select the most valuable features and the hyperparameters of our neural network.

Our chromosomes have 28 genes. For the first 25 genes in the chromosome, we use binary encoding to encode feature selection. Each of these genes represents a different feature. The gene value equal to one means its corresponding feature was selected; otherwise, it was not. The 26th, the 27th, and the 28th gene were encoded use values. The 26th gene is an integer from 4 to 50 stands for the number of hidden neurons. The lower bound of 4 neurons is chosen because the number of hidden neurons in a network for classification is usually not less than the number of output neurons. The upper limit of 50 neurons is chosen because there are only at most 25 input features, and too many neurons are redundant based on our experience. The 27th gene is a float from 0 to 1, representing our neural network's learning rate. The 28th gene describes the number of training epochs. We encode it as an integer from 4 to 20. If the value of this gene is 4, the number of training epochs is 4×500, which is 2000. The same holds for the rest of the values. The range of values for this gene was chosen empirically.

The F-measure of the validation set is chosen as our fitness value. Our goal is to maximize this fitness. The F-measure is used because the accuracy tends to be affected by the data distribution of the validation set, and our

validation set usually has data from only four people. The data for each person has only two categories. Therefore the distribution is likely to be unbalanced.

Each time we evaluate the fitness value of an individual, we randomly reassign the training and validation sets to avoid overfitting to particular training and validation set assignment. Usually, the genetic algorithm only recalculates the fitness value of an individual that has changed. However, we have observed in our experiments that some individuals who are not generalizable may happen to have a high fitness value due to the randomness of the training and validation set assignment, as well as the randomness in the initialization of the model parameters. Such bad individuals are survived in the generation due to their high fitness value and are output as the best choice. Therefore, in our implementation, we recalculate the fitness value of all individuals in the population at each generation to ensure that we end up with the best individual that performs well in every random situation. This also facilitates our offspring to search around generalizable individuals to find individuals that can combine generalization and good performance. Choosing to reassign the dataset at each iteration and recalculate the fitness value, instead of using k-fold cross validation to reduce the effect of randomness, is due to the limited computational resources we have and the high probability of an individual being changed. Besides, the measure we choose is sufficient for our needs according to our observation.

We use Tournament Selection of the tournament size three and Uniform Crossover method. We set the population size to be 100 and we terminate evolution when the number of generation achieve 40. The crossover probability of an individual is 0.5. The independent probability for each gene to be exchanged is 0.3. There is a 0.2 probability that an individual will be mutated. The independent probability for each gene to be mutated is 0.1.

2.6 Pruning and Different indicators

Pruning and Neuron Behaviour We prune the network according to the distinctiveness of hidden neurons[5]. Assume our input is an M $\times$ N matrix, M is the number of features, N is the number of samples. Assume the input layer is fully connected with the hidden layer. The size of the output activation vector of each hidden neuron is N $\times$ 1. It consists of the activated outputs of the specific hidden neuron from every sample in the dataset. Through those output activation vectors, we can derive the distinctiveness of hidden neurons. The output activation vector is an indicator of hidden neuron functionality called neuron behavior in [4]. We can use the angle between two output activation vectors to recognize the similarity of two hidden neuron's functionality. Before calculating the angle, we normalize the vectors to [-0.5, 0.5], which can help us get the angle between 0-180°.

In standard pruning, if this angle is less than 15°, those two hidden neurons are considered to be similar to each other. One of them can be pruned and add its weight matrix to the other without harming the network's capability. If this angle is greater than 165°, the two vectors are believed to complement each other. Both of them could be removed, and the network should still act almost the same.

Nevertheless, in our experiment, same as the original test of weight matrix indicator [4], we only focus on the quality of different hidden neuron indicators. We are not concerned with whether it is similar or complementary, which can all be regarded as identical. We just want to distinguish between identical and orthogonal. Hence, we remapped the angles to 0-90° and progressively pruned one of the two hidden neurons whose angle is the minimum between all angles each time.

Before each pruning, we calculate the minimum angle among the angles of all hidden neuron pairs based on three indicators, i.e., the weight matrix indicator, the improved weight matrix indicator, and the neuron behavior indicator, for comparison. The angle here is also between 0 and 90°. The calculation steps for the weight matrix indicator and the improved weight matrix indicator are as follows.

Weight Matrix Besides the output activation vector, we are also interested in checking the distinctiveness of the weight matrix since it was declared to be not sufficient to describe the hidden neuron functionality in [4] for their task. According to their design, we use the weight matrix between the input layer and the hidden layer. Assume our input is an M $\times$ N matrix, M is the number of features, N is the number of samples as we described before. Assume we have H hidden neurons in the hidden layer. The weight matrix we get should be an M $\times$ H matrix. The weight matrix should be normalized by its global minimum and maximum and then divide into H M $\times$ 1 vectors as the weight indicator for each hidden neuron.

Improved Weight Matrix We change the normalization method to improve the weight matrix indicator. Instead of normalizing the entire weight matrix using its global maximum and minimum, we divide the weight matrix into weight vectors first and then apply Min-Max normalization to every vector independently. In this way, each hidden neuron is treated separately. It is beneficial for us to calculate the angles.

The experiment uses Python, PyTorch, and DEAP[3] with an Intel® Core^{TM} i7-8650U CPU with 1.90 GHz, 16.00 GB of RAM, and Microsoft Windows 10 64-bit operating system. The parts of preprocessing, and selecting feature and hyperparameter with the genetic algorithm are run only once. The other parts, i.e., classifying use neural network with the feature and hyperparameter we selected, progressively pruning, and calculating the minimum angle of the three indicators before each pruning, are repeated 20 times to avoid the effects of randomness. The averages of the results are used for evaluation.

3 Results and Discussion

3.1 Feature and Hyperparameter selection

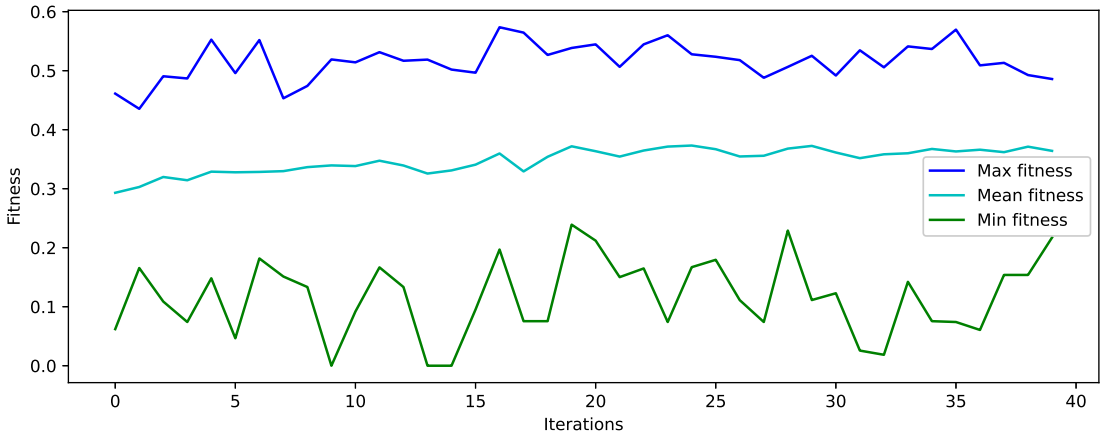

Fig. 1. Maximum Mean and Minimium Fitness during Evolution

Fig. 1 shows the change in maximum, mean, and minimum fitness values as the number of generations increases. Unlike the general genetic algorithm results graph, our maximum fitness line is very uneven. This is not due to a genetic selection error or a too high probability of variation, but because our fitness is affected by randomness and we choose to recalculate the individual fitness values every generation. As we can see from the graph, there is little difference between fitness values after 40 generations and the initial generation, with only a slight increase in the mean fitness value. Given that the initialization, crossovers, and mutations are random and the probability of mutation is not very small, we can basically be sure that we are searching in a space where convergence is not very good. Therefore, the best individual retained to the last must be better than an entirely randomly generated individual, since at least this one performs consistently over some iterations, but we cannot ensure that the one we have selected is the global optimal. The best individual at the last generation is shown in Table 3. We use the selection it represents to complete the subsequent experiments.

Table 3. The best individual and its meaning

best individual	[0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 25, 0.06733994248408581, 10]
selected features	['max_F7', 'min_F7', 'var_F7', 'sum_F7', 'krt_F7', 'mean_first_diff_F7', 'abssum_F7', 'var_F7.1', 'log_F7', 'aac_F7', 'shannon_F7', 'pe_F7']
selected number of hidden neurons	25
selected learning rate	0.06733994248408581
selected number of epochs	5000

3.2 Classification

Table 4. Classification Results

	Accuracy	Precision	Recall	Specificity	F-measure
Training	86.78%	86.77%	86.64%	93.37%	86.71%
Testing	35.38%	35.14%	34.91%	67.44%	35.02%

Table 4 shows the result of how our classifier performs on the music effect dataset with the features we selected. All evaluations for the training performance are over 85%. The classifier can correctly classify all three labels in most of the situations in the training set. However, according to the testing performance, the accuracy of testing is 35.38%, the classifier performance is only slightly better than randomly choosing a label; it is not enough. Essentially, we can conclude that the classifier is a failure product. The precision, recall, specificity, and f-measure also prove that. Notice that the specificity 67.44% is much higher than the other criteria is because we calculate it by the mean of three binary classifications of each label. Bases on the data distribution, 33% of data are labeled positive, the remaining 67% of data are labeled negative. As a result, the specificity of a random choice classifier is 67%, which is very close to the result 67.44% of our classifier. In the original paper [9], their accuracy is over 95%, and all other evaluators are over 90%.

Since we already use the genetic algorithm to select features and hyperparameters, we can conclude that the failure of our classification is because the F7-channel data we used is not enough to train a classifier. There are a total of 14 channels of data the original paper[9] used. The different channel contains different information, and there might be some synergistic effect between different channels. We cannot train an effective classifier only by the F7-channel data. In addition, the original paper use Levenberg—Marquardt methods as the training function of their network. That is different from our choice. This difference may somehow influence the classifier's performance, but it should not be the main reason.

3.3 Indicators of Hidden Neuron Functionality

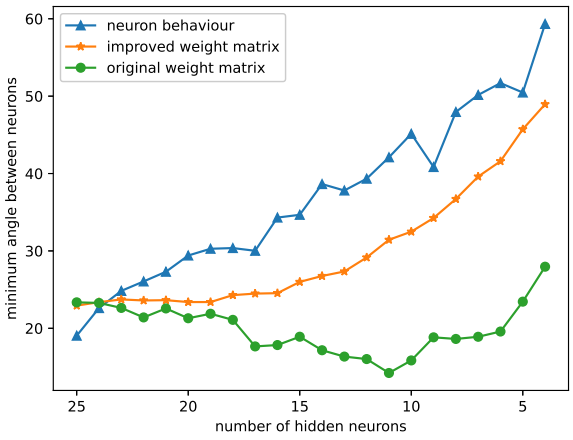

Fig. 2. Original weights, improved weights, and neuron behavior during pruning

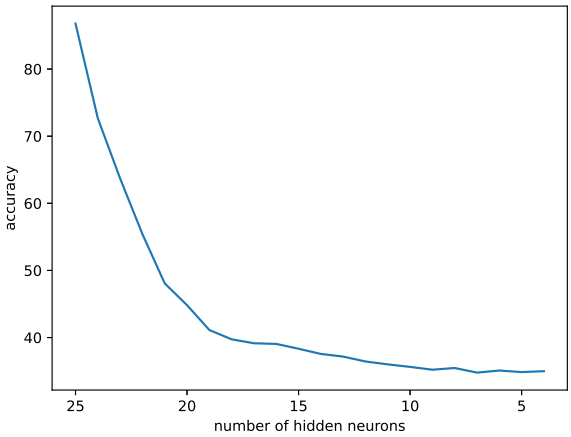

Fig. 3. Change of accuracy during pruning

From the neuron behavior in Fig. 2 we can see that, in the beginning, the minimum angle between all pairs of hidden neurons is already about 20, which is higher than the standard 15. Therefore, all 25 hidden neurons are somehow helpful for the classification. This observation verifies that the selection result of our genetic algorithm is meaningful. Fig. 3 shows the change in classification accuracy on the training set as the number of hidden neurons decreases. The pruning of the first five or six least significant neurons strongly influences the performance of our

classifier on the training set. Due to the accuracy after the sixth pruning is already less than 40% close to the accuracy of a random choice, we cannot say the rest of the hidden neurons are not important. Actually, they should also be essential because the angle between them and other hidden neurons is higher than the first five or six neurons we pruned. This also proves that all hidden neurons in our network are crucial for our classifier.

Another observation is that, as we can see in Fig. 2, the improved weight matrix performs better than the original weight matrix. The minimum angle of the original weight matrix is as coarsen as described in [4]. Moreover, the minimum angle of the neuron behavior basically progressively increases during the pruning. Although the improved weight matrix indicator is not as good as the neuron behavior indicator for the first 10 neurons, the latter part of its curve is also on the rise, which is roughly consistent with our baseline neuron behavior indicator. On the contrary, the original weight matrix indicator behaves very inconsistently with the baseline—the minimum angle decrease in the first 15 pruning. We can conclude that the original weight matrix is not a good indicator; its value is unstable and meaningless. The improved weight matrix is a better indicator. It is not as good as neuron behavior in terms of the span of angle values and it may be a little wrong about the first few angles, but it can be in rough agreement with neuron behavior in other aspects.

4 Conclusion and Future Work

This paper tries to use a neural network to classify the EEG brainwave data from the F7 channel to see whether it can correctly determine the type of music the participant is listening to. We first delete some outliers from the dataset and then use Min-Max to normalize the data so that features can be learned balanced. Then we use the genetic algorithm to select the features and hyperparameters of our neural network to find the optimal settings. The experiments demonstrate that a qualified classifier cannot be trained by a neural network using only the F7 channel EEG data. Of course, the loss function used in our experiments is slightly different from that in the original paper[9], but we do not think this is the main factor for the failure.

Although the classifier does not work well, it has highly accurate for the training set, indicating that the network has been trained. We perform a gradual pruning on this network by cutting off one of the two most similar hidden neurons decided through the output activation vector, and calculating the minimum angle between the original weight matrix, the improved weight matrix and the neuron behavior indicator before each pruning. Comparing with the minimum angle of neuron behavior indicator, We conclude that by changing the weight matrix normalization method, the improved weight matrix is a better indicator than the weight matrix in the original paper. The trend of the minimum angle calculated by the improved weight matrix and neuron behavior is basically the same after each pruning.

Since the neural network does not well classify the dataset used in our experiments, we are not sure whether the improved weight matrix indicator can correctly reflect the neuron's function because it is evident that the accuracy of the training set alone is not objective. So if we want to know whether the improved weight matrix is a good indicator that can be used for pruning, we should also test on a good classifier. For example, use angles obtained from the improved weight matrix to determine the functionally similar neurons and prune them based on this result. Using this pruned classifier performs a test on the testing set to see whether the function of the neural network remains basically the same as the original one after pruning the similar neurons. If our improved weight matrix can pass those tests, it will be a qualified hidden neuron indicator and can be used more widely.

In addition, bias should also reflect the function of neurons to some extent. We can consider whether bias can also be part of the weight matrix indicator somehow in the future, instead of simply discarding this vital element.

References

1. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
2. Dahl, G.E., Yu, D., Deng, L., Acero, A.: Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Transactions on audio, speech, and language processing **20**(1), 30–42 (2011)
3. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (jul 2012)
4. Gedeon, T.D.: 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 (1995)
5. Gedeon, T., Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications. vol. 1, pp. 119–126 (1991)

6. Grigorescu, S., Trasnea, B., Cocias, T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. Journal of Field Robotics **37**(3), 362–386 (2020)
7. Harmat, L., Takács, J., Bódizs, R.: Music improves sleep quality in students. Journal of advanced nursing **62**(3), 327–335 (2008)
8. Kemper, K.J., Danhauer, S.C.: Music as therapy. South Med J **98**(3), 282–8 (2005)
9. Rahman, J.S., Gedeon, T., Caldwell, S., Jones, R.: Brain melody informatics: Analysing effects of music on brainwave patterns. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
10. Rahman, J.S., Gedeon, T., Caldwell, S., Jones, R., Hossain, M.Z., Zhu, X.: Melodious micro-frissons: detecting music genres from skin response. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
11. Rueda, F.M., Grzeszick, R., Fink, G.A.: Neuron pruning for compressing deep networks using maxout architectures. In: German Conference on Pattern Recognition. pp. 177–188. Springer (2017)
12. Safarik, J., Jalowiczor, J., Gresak, E., Rozhon, J.: Genetic algorithm for automatic tuning of neural network hyperparameters. In: Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything. vol. 10643, p. 106430Q. International Society for Optics and Photonics (2018)
13. Tan, F., Fu, X., Zhang, Y., Bourgeois, A.G.: A genetic algorithm-based method for feature subset selection. Soft Computing **12**(2), 111–120 (2008)
14. Tharwat, A.: Classification assessment methods. Applied Computing and Informatics (2020)

Appendix

Table 5. Descriptive statistic of the original data (1)

	mean_F7	max_F7	min_F7	std_F7	iqr_F7	var_F7	sum_F7	skw_F7
mean	1.8191	6.3541	0.2438	1.9005	1.9022	12.1885	24.8340	0.8573
std	2.5700	9.7837	0.7636	2.9311	2.9344	33.3888	37.0508	0.7728
min	0.1772	0.3311	0.0000	0.0581	0.0229	0.0034	1.3113	-1.3682
25%	0.4897	1.2277	0.0057	0.3637	0.5141	0.1323	6.0505	0.3202
50%	0.8148	2.0145	0.0153	0.5800	0.8076	0.3365	9.7277	0.7739
75%	1.6324	6.1330	0.2309	1.8853	1.5149	3.5542	23.7085	1.4211
max	21.3101	49.6533	14.5919	16.2437	22.0970	263.8577	230.5620	2.3908

Table 6. Descriptive statistic of the original data (2)

	krt_F7	mean_first_diff_F7	mean_second_diff_F7	rms_F7	abssum_F7	ssi_F7	var_F7.1
mean	3.1366	1.0634	1.5048	2.6298	24.8340	300.9961	23.3238
std	1.6263	1.6182	2.3566	3.7680	37.0508	820.3931	64.6009
min	1.1424	0.0708	0.0899	0.2281	1.3113	0.3272	0.0548
25%	1.8407	0.2250	0.3230	0.6410	6.0505	5.1826	0.4408
50%	2.5394	0.3711	0.5296	1.0047	9.7277	11.8466	1.1353
75%	4.0734	0.9841	1.3028	2.4432	23.7085	90.1521	6.6862
max	7.3014	9.9750	16.5380	22.2852	230.5620	6,994.3128	595.9555

Table 7. Descriptive statistic of the original data (3)

	mav_F7	log_F7	aac_F7	dasdv_F7	dfa_F7	fuzzy_F7	shannon_F7	pe_F7	hjorth_F7	hurst_F7
mean	1.8191	0.9327	1.0838	1.6615	1.1966	457.7391	0.9390	0.4759	0.6291	0.6917
std	2.5700	1.5831	1.6455	2.5528	1.8581	5,446.7663	0.2530	0.2042	0.1553	0.2865
min	0.1772	0.0916	0.0746	0.0972	0.0075	-0.0909	0.2338	0.0000	0.2529	0.1354
25%	0.4897	0.2133	0.2207	0.3129	0.1958	0.5054	0.6886	0.2181	0.5172	0.4910
50%	0.8148	0.4110	0.3715	0.5080	0.4145	0.9074	0.9503	0.5812	0.6574	0.6031
75%	1.6324	0.9400	0.9992	1.5659	1.2240	1.7471	1.1034	0.6373	0.7273	0.8874
max	21.3101	20.4723	10.6854	13.0023	15.1211	65,535.0000	1.3705	0.7245	1.0382	1.5707