Automated Evolution and Explanation of Deep Neural Network Architectures

Owen Pendrigh Elliott

Research School of Computer Science, Australian National University, Canberra, Australia
u5788450@anu.edu.au

Abstract. Artificial Neural Networks (ANNs) are powerful statistical models that outperform other machine learning techniques in many domains. The main shortcoming of ANNs is their black box nature, which makes the design of a network topology as well as the explainability of the final trained network difficult. This paper explores a solution to this problem by proposing an end-to-end system for building a deep dense neural network architecture and explaining its predictions. The architecture is designed using a Genetic Algorithm (GA). Subsequently, the network predictions are explained through Deep neural network Rule Extraction via Decision tree induction (DeepRED) [22] and an explanation framework proposed by T. D. Gedeon, et al. [8, 7]. This system is applied to a dataset of electroencephalogram (EEG) readings with target values being calm or stressful. The network designed by the GA achieves exceptional accuracy with good generalisation capabilities; scoring 100.0% accuracy on test sets over numerous K-fold cross validation experiments in just 50 epochs. The rules obtained with DeepRED achieve 93.8% fidelity meaning they are not as accurate as the neural network and do not generalise as well however are of high enough quality to aid interpretation.

Keywords: Explainable AI · Neural Networks · Genetic Algorithm · Electroencephalogram

1 Introduction

Neural networks are proven to be powerful classifiers and regressors in many domains however, their black box nature often makes their application harder to justify. There are two downsides to the black box nature of neural networks that this paper tackles: architecture design and explainability. In this paper only dense feedforward neural networks are considered.

Deep neural network architecture design is often done in a somewhat arbitrary fashion. General rules of thumb exist and domain knowledge can be applied but the problem is often approached as trial and error. Additionally, the deeper a neural network becomes the harder it is to explain. To combat both of these issues, a Genetic Algorithm (GA) to design neural network architectures is proposed. This is inspired by the L-NEAT algorithm which is an extension of NEAT [19, 4].

In many domains it is of great importance that the decision process used to reach a conclusion is human interpretable and auditable. Many approaches to explanation of neural network decision processes have been proposed, these largely fall into three categories: pedagogical, decompositional and eclectic [2]. This paper explores the use of a decompositional approach to rule extraction alongside additional explanation mechanisms generated in a pedagogical fashion. In particular this paper explores an explanation process proposed by T. D. Gedeon, et al. [8, 7]. For the rule extraction component Deep neural network Rule Extraction via Decision tree induction (DeepRED) proposed by J. R. Zilke, et al. [22] is used. For additional explanation, *characteristic patterns* and a *causal index* are used to gain insight into what the network considers important [8].

The end-to-end framework for neural network architecture design and explanation proposed in this paper is achieved by combining the aforementioned genetic algorithm and explanation techniques. This framework allows for a task specific neural network architecture to be constructed without sacrificing the explainability of the end system.

This framework is applied to and evaluated on a dataset of electroencephalogram (EEG) readings with target values being calm or stressful. EEG readings are not inherently easy to understand, generating predictions from them with a neural network only exacerbates this. The dataset used for this paper is made from summary statistics of EEG readings, combining this with an explanation framework makes the results of the neural network classification process much clearer to users.

2 Data

For this project, a dataset of EEG readings from individuals being exposed to calm and stressful videos is used. This dataset contains 144 readings from 24 participants, all labelled as either calm or stressful. This data is related

to work in bimodal stress recognition done by Irani et al., in this case utilising EEG readings instead of thermal video [12]. The pre-processing of the data into summarised features is inspired by work from Rahmin et al. [15].

2.1 EEG Data

There are two classes in the data: calm and stressful. The data is balanced with 72 examples of each class. The raw data contains 15 features that have been extracted from EEG readings, these features are extracted for each of the 14 channels from the EEG creating a total of 210 attributes. As is discussed in the subsequent section only one of the 15 features is deemed useful for the classification task at hand.

Data Exploration Inspection of five number summaries and histograms for each attribute shows that some features are skewed and/or have notable outliers. One example of this is the *sum* feature, which is depicted in Figure 1. This skewed distribution is important to keep in mind when observing neural network architectures in subsequent sections.

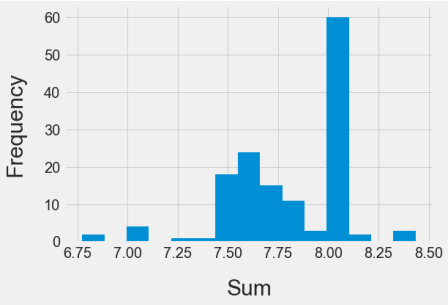

Fig. 1: Histogram of the *sum* feature aggregated by averaging all channels.

Plotting different combinations of channels for each feature allows for visual identification of class separable features. Inspection of the such plots showed that the *sum* features provided good class separation. Thus, all other features are dropped and only the 14 channels of *sum* features are used. An example of the clear separation between the classes for the first 3 channels of the *sum* features are shown in Figure 2. There are a small number of points that appear more entangled between the classes however these appear amongst outliers. This means that a neural network should be able to disentangle these with complex enough decision boundaries.

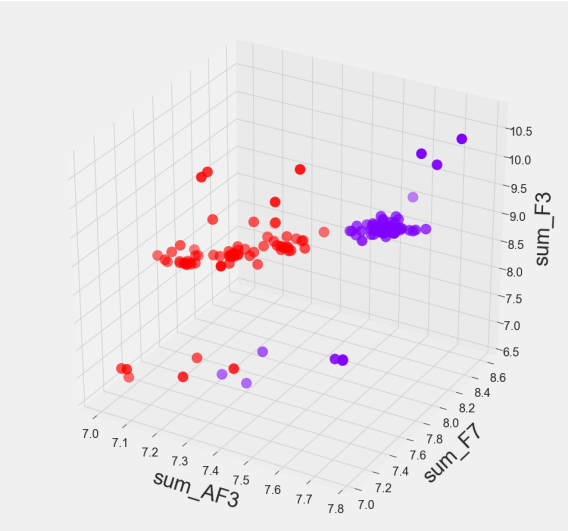

Fig. 2: Class separability for first three channels of the Sum feature.

Selected Features The usefulness of the *sum* features is supported by other research into EEG classification which identifies that frequency is an important feature [20], something which heavily effects *sum*.

2.2 Data Pre-processing

Pre-processing of the data was kept simple, a min-max $[0-1]$ normalisation was used to form a training dataset with all points in the range $[0-1]$. This simple pre-processing was used for two reasons. Firstly, it simplified the denormalization process for generating explanations. Secondly, it proved more than adequate to achieve high accuracy classification.

3 Evaluation Methods

In this paper, testing was done with K-fold cross validation repeated multiple times. This method was chosen to ensure that the generalisation of any models used was tested exhaustively, despite the small amount of data. All experiments were done using $K = 5$ and repeated 10 times. The average test accuracy over all folds and experiments is thus the measure of how good a model is. In the case of this dataset K-fold cross validation with $K = 5$ means 115 instances for training and 29 instances for testing.

As a baseline for classification a decision tree was used. As the ultimate goal is explainability, the network must outperform a decision tree for it to be worth considering. A decision tree evaluated with the above method obtains 95.2% accuracy and 95.5% F1 score across all folds and experiments, making this the baseline.

4 Neural Network

4.1 Network Architecture Design via a Genetic Algorithm

The process of designing a neural network can be viewed as having two main steps: design of network architecture and training of the chosen network architecture. These are two inherently different problems. The first can be viewed as a non-differentiable global search where the goal is to find a suitable local neighbourhood while the second can be viewed as a differentiable local search of a selected neighbourhood. It follows logically that the first problem being non-differentiable makes it a perfect candidate for a genetic algorithm and the second a perfect candidate for backpropogation [4].

The genetic algorithm used in this work differs from others such as NEAT and L-NEAT [19, 4] in that all layers are fully connected. However, like L-NEAT, this genetic algorithm deals exclusively with architecture, not weights and biases [4, 6]; weights and biases are learnt during fitness evaluation. The population of network architectures is then evaluated by the method described in Section 3. The Adam optimiser was used with a learning rate of 0.001 when evaluating a network architecture, as is the case with many domains Adam provided the fastest convergence in training and performed at least as well as stochastic gradient descent (SGD) over multiple trials [13]. SGD and Adadelta were also tested with various learning rates but underperformed Adam.

The genotype of a network is represented as three lists of numbers: LayerIDs, ActivationIDs and OutFeatures. These are each as long as the maximum depth specified by the user. The IDs for layers and activations correspond different possible layers and activations. OutFeatures simply represents the number of out features a linear layer at the corresponding index will have. There may also be no linear layer at a given index meaning the number of OutFeatures at that same index has no effect on the network architecture.

This paper only considers potential architectures as fully connected dense networks that are comprised of linear and batch normalisation layers. Additionally at each of the possible layers one of the following activation functions can be used: Sigmoid, LogSigmoid, ReLU, Leaky ReLU, CELU, SELU, GELU, ReLU6 or Tanh. Instead of an activation function, the genetic algorithm can also use 'drop out' with a 0.2 probability [18].

There are three possible mutation operations. Layer mutation will replace a point in the LayerIDs with either no layer, a linear layer or a batch normalisation layer. Activation mutation will replace a point in the ActivationIDs with one of the activation functions listed above or none. Finally, neuron mutation replaces one of the OutFeatures values with a random number between 1 and the maximum number of neurons. Crossover is done by concatenating the LayerIDs, ActivationIDs and OutFeatures for each of the parents and randomly exchanging some number of points between a minimum and maximum.

The fitness of a network is the accuracy it achieves when trained for a small number of epochs using the method described in Section 3. It is important to use a small number of epochs to train the networks during evaluation

to ensure that the fittest networks are easier to train. Networks that are easier to train are usually smaller and subsequently these networks are easier to generate explanations for.

Fitness is accuracy and is in the range [0,1]. For selection, the fitnesses are scaled as shown in equation (1).

$$fitness' = (fitness + 1)^2 \tag{1}$$

Scaling the fitnesses like this increases separation between high and low performing candidates. For any individual, its probability of being selected is its scaled fitness divided by the total scaled fitnesses of the population. Elitism is also used, meaning the fittest network is guaranteed a spot in the next population.

This process is defined by the hyperparameters in Table 1.

Table 1: Hyperparameters for Genetic Algorithm

Hyperparameter	**Value**	**Description**
Max Neurons	256	Maximum out features of a linear layer
Max Layers	5	Maximum depth of network
Population Size	100	Size of population
Max Initial Mutations	5	Maximum number of mutations to diversify initial population
Crossover Rate	80%	Probability to crossover
Max Cross Points	6	Maximum points in DNA to exchange
Min Cross Points	1	Minimum points in DNA to exchange
Layer Mutate Chance	2%	Chance to mutate a layer
Activation Mutate Chance	4%	Chance to mutate an activation
Neuron Mutate Chance	5%	Chance to mutate an out feature size
N Generations	10	Number of generations
K	5	Number of folds for cross fold validation
N Experiments	10	Number of times to run K-fold cross validation
Epochs	10	Number of epochs to use when evaluating fitness
LR	0.001	Learning rate for Adam optimiser

4.2 Architecture

The neural network created by the genetic algorithm is depicted in Table 2:

Table 2: Neural Network Architecture

Component Number	**Component**	**Trainable Weights**	**Trainable Biases**
1	Linear	14 × 220	220
2	GELU	N/A	N/A
3	Linear	220 x 84	84
4	BatchNorm1d	84	84
5	ReLU	N/A	N/A
6	Linear	84 x 1	1
7	Sigmoid	N/A	N/A

The architecture above was the fittest network from an example run of the genetic algorithm, running the genetic algorithm again will potentially result in a different architecture. Weights for the linear layers are initialised by sampling the Kaiming uniform distribution using a negative slope of $\sqrt{5}$ for the rectifier [14, 9].

From multiple tests of the genetic algorithm it was observed that the best networks always had at least one batch normalisation layer. This makes sense as batch normalisation tends to accelerate training and the networks were evaluated on a very small number of epochs [11]. Batch normalisation also helps the network remain robust to the skewed distribution of certain features, as was identified in the data exploration phase. It was also observed that the Gaussian Error Linear Units (GELU) activation would appear in most of the fittest network architectures. Evidently GELU nonlinearity improves performance in this classification task - the original paper proposing the function also observed better performance for GELU compared ReLU and ELU on multiple benchmarks [10].

4.3 Evaluation of Performance

The genetic algorithm proved effective for designing an optimal topology for the network. Over the ten generations a steady increase in the performance of the fittest network can be observed. Figure 3 shows this increase, with the last generations fittest network topology scoring over 10% higher accuracy than the initial population. This outperforms the baseline decision tree from Section 3 with only 10 epochs of training. Also noticeable is that the minimum accuracy does not trend upwards. This is because some crossovers can produce significantly worse neural networks.

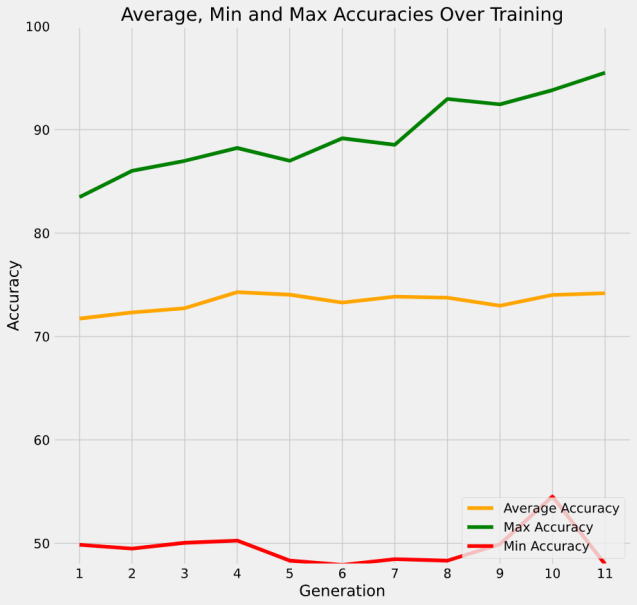

Fig. 3: Average, max and min test accuracies for the population at each generation

Further training of the fittest model, detailed in Table 2, resulted in an average test accuracy of 100.0% in 50 epochs using the evaluation method described in Section 3.

5 Generating Explanations

5.1 Causal Index

As part of the explanation process, a mechanism allowing the identification of inputs that can be considered important to the neural network is required, for this, a *causal index* is used. The *causal index* is defined as the rate of change in the output Y with respect to each input X_i in the input vector X [3]. This means that the *causal index* is proportional to the partial derivative of each input X_i with respect to the single output neuron Y, this is shown in equation (2).

$$C_i \propto \frac{\partial Y}{\partial X_i} \tag{2}$$

where C_i is the gradient at the index i in the *causal index* C. This represents the relationship between the i-th input neuron and the output neuron. The magnitude of a number C_i indicates the strength of the relationship and the sign indicates a positive or negative correlation. To aid interpretation, a *relative causal index* (denoted by RC) is also created; this represents the relative importance of each input in relation to the given output [5]. RC is given in (3).

$$RC = \frac{C}{\|C\|_1} \tag{3}$$

The magnitude of the values in the *causal index* is dependant on how confident the neural network is in its predictions as the gradients are smaller closer to the minimum and maximum of the Sigmoid activation. Thus, the

relative causal index is used to identify important inputs by instead looking at their relative importance to the output.

5.2 Characteristic Input Patterns

Generating a *causal index* for every input in the dataset complicates the explanation process. The problem is thus simplified through the use of *characteristic input patterns* proposed by T. D. Gedeon, et al. [8, 7]. *Characteristic input patterns* are patterns which are representative of the networks' decisions rather than of the ground truth labels. In this paper characteristic on (CON) and characteristic off (COFF) patterns are generated in relation to the output neuron. Thus CON corresponds to a prediction of *stressful* and COFF corresponds to a prediction of *calm*.

Characteristic patterns are created by taking the mean of each vector component for all vectors in the input data that cause a prediction of a given class.

5.3 Rule Extraction

For rule extraction the DeepRED algorithm is used [22]. Rules are evaluated by their fidelity given in (4).

$$fidelity = \frac{agreements}{N} \tag{4}$$

where *agreements* is the number of times the rules when applied to an input x agree with the neural networks classification of x and N is the number of testing examples.

DeepRED is an algorithm to induce rules from a neural network using decision trees. DeepRED is a decompositional approach however it will work with any feedforward neural network [22] and is an extension of the CRED algorithm [17]. To decompose the network, the architecture is split at every linear layer as shown in Table 3:

Table 3: Neural Network Decomposition

Component Group ID	**LayerID**	**Layer**
1	1	Linear
	2	GELU
2	3	Linear
	4	BatchNorm1d
	5	ReLU
3	6	Linear
	7	Sigmoid

Using the decomposed version of the network (Table 3), decision tree classifiers are trained to predict the output of a component group given an input. This is done by first training a decision tree to predict the output of the last component group using the output from the previous component group. The rules generated from this decision tree are then converted into class labels for each rule. Another decision tree is then trained to predict these new labels using the input of the component group that comes before them. This is repeated until there are trained decision trees to model each component group. The rules from each decision tree are then consolidated to form rules that map from input to output. Intermediated inequalities in the consolidated rules that pertain to hidden layers can be removed to simplify the rules and aid interpretation. This approach yields very accurate rules which, most importantly, are easy for a human to understand.

For the subsequent evaluation, the same K-fold cross validation experiment as outlined in section 3 was performed with the neural networks predictions as targets. The rules extracted using DeepRED obtained 97.1% fidelity on the training set and 93.8% fidelity on the testing set. While the extracted rules perform worse than the neural network and generalise worse than the baseline decision tree they are still of high enough fidelity to be useful in aiding explanation of neural network decision processes.

6 Results

Explanations of the neural networks decision process for a given input are generated as follows:

1. Present the prediction of the neural network.
2. Present the prediction of the extracted rules and the rule that was satisfied.
3. Find the closest characteristic pattern.
4. Identify the important inputs and their characteristic values.
5. Show how far away the important inputs were from their characteristic values.

In Section 4.3 it was shown that the genetic algorithm successfully evolved a robust neural network capable of achieving 100.0% accuracy on unseen testing data from multiple possible test partitions. Additionally, in Section 5.3, the capabilities of DeepRED were assessed proving its ability to perform robust rule extraction, albeit with a slightly worse ability to generalise than the baseline decision tree.

Given the provable robustness of the neural network, all components of the framework can now be combined to form an explanation system. The neural network, DeepRED rule extraction, *characteristic input patterns* and *causal index* are thus trained/calculated using the entire dataset.

With 50 epochs of training the neural network obtains 100.0% accuracy (as expected). The rules extracted by DeepRED for the entire dataset achieve 97.92% fidelity - similar to the training data fidelity observed in Section 5.3. The *characteristic input patterns* are closest by Euclidean distance to 91.67% of examples from their respective classes.

The extracted rules with 97.92% fidelity are provided below, followed by an example explanation that was generated by the system. Note that the example explanation also includes the true label that is known from the data, this is simply for illustrative purposes and would not be included in a productionised version of the system.

Extracted Rules:

```
sum_T7 <= 7.806 and sum_O1 <= 6.575 then calm
sum_T7 <= 7.806 and sum_O1 > 6.575 and sum_P7 <= 8.449 then stressful
sum_T7 <= 7.806 and sum_O1 > 6.575 and sum_P7 > 8.449 then calm
sum_T7 > 7.806 then calm
```

Explanation example:

```
Explanation of prediction for participant 1
Real label is stressful

Explaining network output for input:
sum_AF3: 7.2103     sum_F7: 7.826
sum_F3: 8.5539      sum_FC5: 7.3452
sum_T7: 7.5841      sum_P7: 8.2268
sum_O1: 7.4689      sum_O2: 7.7649
sum_P8: 8.2481      sum_T8: 8.0818
sum_FC6: 6.3042     sum_F4: 7.635
sum_F8: 7.4892      sum_AF4: 7.3096

Neural network prediction: stressful

Rule set prediction: stressful

Satisfies the following rule:
sum_T7 <= 7.806 and sum_O1 > 6.575 and sum_P7 <= 8.449 then stressful

Input is closest to characteristic pattern of calm

Important attributes and their characteristic values:
sum_AF3: 7.582
sum_P7: 8.6496
sum_FC6: 6.7111

Difference between input and important characteristic values:
sum_AF3: -0.3717
sum_P7: -0.4229
sum_FC6: -0.4069
```

The example explanation provides confidence in the neural network prediction as it shows that the neural network and rule set agree. The *Sum P7* variable is identified as an important variable by the *relative causal index* and also appears in the satisfied rule. The extracted rules show that *Sum P7* is the only variable that allows the rule set to differentiate between *calm* and *stressful*; reinforcing the notion that DeepRED generates rules reflecting the inner workings of the neural network. The difference between the provided example and the characteristic values is very high; the example is in fact closer to the incorrect characteristic pattern. However, the *relative causal index* and the extracted rules are effective at contextualising this discrepancy: the *Sum P7* value is especially low in this example and the rules show that a lower value for *Sum P7* produces a prediction of *stressful*.

7 Discussion

The proposed framework proves itself to be highly effective at both the classification and explanation components of the task. Additionally, aside from the provision of a small selection of hyperparameters the framework presented is largely automated; architecture design, training, decomposition and explanation require minimal user input. The explanations generated are capable of giving a user invaluable insight into the decision process.

The genetic algorithm proposed for evolution of neural network architectures was capable of developing a very high performing network in a small number of generations. The process was not computationally expensive with the small dataset and constrained architecture used in this paper; each generation took approximately 80 seconds on a consumer graphics processing unit (Nvidia 2080 Super). In other applications however, the exact approach used in this work might be too computationally expensive. This could be counteracted by using a train-test split with only one experiment instead of the multiple experiments of K-fold cross validation used in this work. If a larger dataset was being used then a train-test split is more likely to have similar distributions in the train and test partitions. Additionally the process of evaluating each member of the population could be parallelised, especially if more compute hardware was available.

Normally decompositional approaches do not apply to all model architectures. However, DeepRED is applicable to any feedforward network, although the algorithm does work best with dense networks. Decomposing the network means that the rules extracted using DeepRED are representative of the complete decision process of the neural network. The extracted rules could even be left in their full form to detail the thresholds of important neurons at each layer in the network if that level of detail was required.

DeepRED has the potential to create a large number of rules although this can be somewhat limited by adjusting the hyperparameters of the decision trees to minimise their depth. In this paper no limitations are imposed on the depth of the decision trees and the number of rules extracted appears to remain reasonable however this is not guaranteed for other neural networks or datasets.

The baseline decision tree outperformed the DeepRED rule extraction process meaning the extracted rules do not act as a white box alternative to a neural network. This framework is better suited to an application where accuracy and explainability are both of great importance. The neural network and the explanation techniques work together to build a cohesive and user friendly system.

8 Further Work

There is still much work to be done in the realm of explainability for neural networks. Decompositional approaches have proven to be very effective in generating comprehensible rules with high fidelity but do not generalise well to all architectures. Pedagogical approaches that work independent of network architecture are far more flexible but often do not perform as well. Pedagogical algorithms such as VIA and HYPINV [21, 16] would likely be better approaches for pedagogical rule extraction with the framework and data used in this paper and this should be explored in future.

For higher dimensional data, the notion of *characteristic input patterns* becomes more complicated. The meaningfulness of averaging vector components and calculating Euclidean distances is diminished as dimensionality increases [1]. A new approach is required for more complex data.

One of the key challenges for work on neural network explainability going forward is creating an approach that operates independent of network architecture. This should include architectures such as convolutional neural networks, which are far more difficult to extract if-then-else rules from, as they are translation invariant; meaning the content of the input is important as oppose to the location of that content in the input.

The idea of network architecture independent approaches is also relevant to the genetic algorithm proposed in this paper. Only deep dense feedforward neural networks were considered however, for many forms of data this

is not sufficient. The framework proposed here would be relatively easy to expand to recurrent neural networks, convolutional neural networks or transformers. Ideally such a system would abstract almost all decisions about network architecture design away from the user.

References

1. Aggarwal, C.C., Hinneburg, A., Keim, D.A.: On the surprising behavior of distance metrics in high dimensional space. In: International conference on database theory. pp. 420–434. Springer (2001)
2. Augasta, M.G., Kathirvalavakumar, T.: Rule extraction from neural networks—a comparative study. In: International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012). pp. 404–408. IEEE (2012)
3. Baba, K., Enbutu, I., Yoda, M.: Explicit representation of knowledge acquired from plant historical data using neural network. In: 1990 IJCNN International Joint Conference on Neural Networks. pp. 155–160. IEEE (1990)
4. Chen, L., Alahakoon, D.: Neuroevolution of augmenting topologies with learning for data classification. In: 2006 International Conference on Information and Automation. pp. 367–371. IEEE (2006)
5. Enbutsu, I., Baba, K., Hara, N.: Fuzzy rule extraction from a multilayered neural network. In: IJCNN-91-Seattle International Joint Conference on Neural Networks. vol. 2, pp. 461–465. IEEE (1991)
6. Fiszelew, A., Britos, P., Ochoa, A., Merlino, H., Fernández, E., García-Martínez, R.: Finding optimal neural network architecture using genetic algorithms. Advances in computer science and engineering research in computing science **27**, 15–24 (2007)
7. Gedeon, T., Treadgold, N.: Extracting meaning from cascade networks. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. vol. 4, pp. 3019–3024. IEEE (1997)
8. Gedeon, T., Turner, H.: Explaining student grades predicted by a neural network. In: Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan). vol. 1, pp. 609–612. IEEE (1993)
9. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
10. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. PMLR (2015)
12. Irani, R., Nasrollahi, K., Dhall, A., Moeslund, T.B., Gedeon, T.: Thermal super-pixels for bimodal stress recognition. In: 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA). pp. 1–6. IEEE (2016)
13. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
14. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)
15. Rahman, J.S., Gedeon, T., Caldwell, S., Jones, R., Jin, Z.: 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), 5–20 (2021)
16. Saad, E.W., Wunsch II, D.C.: Neural network explanation using inversion. Neural networks **20**(1), 78–93 (2007)
17. Sato, M., Tsukimoto, H.: Rule extraction from neural networks via decision tree induction. In: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222). vol. 3, pp. 1870–1875. IEEE (2001)
18. Srivastava, N.: Improving neural networks with dropout. University of Toronto **182**(566), 7 (2013)
19. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation **10**(2), 99–127 (2002)
20. Subhani, A.R., Xia, L., Malik, A.S.: Eeg signals to measure mental stress. In: 2nd International Conference on Behavioral, Cognitive and Psychological Sciences. pp. 84–88. Maldives (2011)
21. Thrun, S.: Extracting rules from artificial neural networks with distributed representations. Advances in neural information processing systems pp. 505–512 (1995)
22. Zilke, J.R., Mencía, E.L., Janssen, F.: Deepred–rule extraction from deep neural networks. In: International Conference on Discovery Science. pp. 457–473. Springer (2016)