Classifying posed vs genuine observed anger, an investigation into the application of distinctiveness-based network pruning, and the application of genetic algorithms in hyperparameter optimisation

Dashiell van Wyk

Research School of Computer Science, Australian National University
u5826173@anu.edu.au

Abstract. A simple feed forward network was used to replicate the efforts of Chen et. Al's previous attempt to classify observed genuine or posed anger based off the pupillary response of the observer with a target of above 60% accuracy. A network reduction technique involving the identification of similarly behaving hidden neurons was then applied to the network. The classification task was reproduced, however where previous work achieved 95% accuracy in classification, our model achieved 78% accuracy, exceeding the target accuracy. The application of the pruning technique resulted in a significant decrease in the model's accuracy. A genetic algorithm was then implemented in an effort to fine-tune and optimize the hyperparameters for a subset of the problem, resulting in a minor increase in accuracy.

Keywords: Binary Classification, Network Pruning, Neuron Behaviour, Network Reduction Technique, Hyperparameter Optimisation, Genetic Algorithm, Optimisation Search

1 Introduction

A data set used in a previous study was chosen in order to model a neural network problem. This could be a reproduction of the problem discussed in the original study's work, or an entirely new problem utilising their data set. In addition to this, we would then investigate the application of a technique from relevant literature and observe its effect on the modelled problem. The problem was then to be further extended by the implementation of deep learning techniques such as convolution neural networks, long short-term memory networks etc. or some form of evolutionary/genetic algorithm. In particular, we implement a basic genetic algorithm with the purpose of finding optimal hyperparameter settings to increase the network's accuracy.

Our data set was "dataset-anger" from Chen et. Al's 2017 paper *Are you really angry? Detecting emotion veracity as a proposed tool for interaction* [1]. This data represents the pupillary responses of subjects who observed short clips of either posed or genuine anger. This data consists of numerical representations of pupillary responses associated with a label 'genuine' or 'posed'. Categorical data such as this lends itself to prediction/classification tasks, so we decided to create a prediction model attempting to classify whether a subject has observed posed or genuine anger based upon their pupillary response patterns. This is a replication of the problem outlined briefly in the discussion of Chen et. al's results.

Gedeon and Harris' paper *Network Reduction Techniques* [2] outlines a number of methods to evaluate and identify potential candidate neurons for pruning from a network. The most deeply explored is the evaluation of neuron distinctiveness, and so this specific technique was chosen for implementation with Chen et. al's dataset, however it was adapted for our implementation.

Genetic algorithms are commonly used in optimisation, employing methods inspired from biological reproduction and evolutionary processes. A solution may be represented by a chromosome, an encoding that represent the variables associated with a particular solution. A population of these chromosomes are evaluated by some fitness function that determines their value, and are selected for a crossover process that produces offspring, some new chromosome. These offspring are then subject to a potential mutation, slightly altering the chromosome. The selection, crossover, and mutation methods can vary between algorithms, as some methods are more conducive to different problems.

Broadly, the aim of this research was to implement and apply techniques previously learned in a practical manner. Specifically that is to create a neural network capable of modelling and predicting whether a subject has perceived posed or genuine anger, based upon the subject's pupillary response patterns. An ideal outcome would be a model that is of equal or superior performance to the model in the original paper [1], however this was not expected nor necessarily possible in the case of some data-sets due to the modern techniques that may have been applied in their work. Instead we set our benchmarks as a model that is better than guessing (accuracy > 50%) and further a model that is superior to the subject's own verbal response (60%) [1]. The accuracy of this model was then used as a benchmark for the model(s) generated by the genetic algorithm.

2 Method

The research was split into three components: an initial building and testing of the prediction network, followed by the application and testing of the network pruning technique on the resulting models. Finally the implementation of a genetic algorithm with the purpose of finding optimal hyperparameter settings for the initial problem.

2.1 Design and implementation of the prediction network

The goal of the problem was to classify whether a subject had observed posed or genuine anger based upon the pupillary response of the subject. This is easily abstracted into a binary classification problem, which are often approached with simple back-propagation networks such as the one implemented here.

The original dataset consists of 400 points of data (20 subjects × 20 videos), each with 9 features: Observer, Video, Mean, Std, Diff1, Diff2, PCAd1, PCAd2, and Label. The features Mean, Std, Diff1, Diff2, PCAd1, PCAd2 were chosen as the inputs for the network, and normalised in the pre-processing by the StandardScaler normalisation functionality provided in sklearn's pre-processing library. The target data (test/target separation discussed below), was normalised using the parameters found in the test data normalisation process. Min-max normalisation was considered and tested, motivated by Isik, Ozden and Kuntalp's 2012 paper [3] stating that neither method was superior. However, min-max normalisation tested poorly on the data set, so statistical normalisation was chosen as method of data transformation.

As with all machine learning tasks, it is important to consider the nature of the data when designing not only the network topology, but also the method in which the network is trained and tested. Traditionally a data set may be split, for example, into 50% training, 10% validation and 40% testing at random or with respect to class representation. However with the dataset in question it must be considered that a single subject is generating multiple points of data, and training indiscriminately may result in a subject bias. Hossian and Gedeon (2017) [4] propose a testing approach for a similar problem by leaving out such a subject during training, and testing on the excluded subject in order to achieve subject-independence.

We tested this approach in our model and additionally took this idea a step further by removing both a single subject and single video from our training data, testing on just the excluded subject and video in order to achieve both subject and video independence. We repeated the training process on every combination of excluded subject and video for a total of 400 (20 subjects × 20 videos) models and took the average result as the expected performance of the model.

The topology of our model consisted of six input neurons corresponding to the features Mean, Std, Diff1, Diff2, PCAd1, PCAd2. 12 neurons using pytorch's built in leaky-ReLU activation function were used as the sole hidden layer. Leaky-ReLU was chosen over both sigmoid and regular ReLU activation in order to avoid potential issues with vanishing gradients or dead neurons, respectively. It should be noted that while the former is unlikely to occur in a shallow network, it was still seen that the model performed significantly worse using a sigmoid activation in the hidden layer. A single output neuron with a sigmoid activation function was chosen as the simplest method for binary classification. The outputs for each test were rounded to either 1 (genuine) or 0 (posed) as a classification process and compared to the target outputs for evaluation as is typical in classification problems.

Pytorch's Binary Cross Entropy was used as a loss function, as it is a special case of cross entropy for binary classification tasks. AdamW, an improvement[1] of the Adam algorithm, was used as an optimisation algorithm. This was chosen over other stochastic optimisation methods as it has been shown by Kinga and Ba (2015) that Adam performs favourably over such methods [5]. Hyperparameters such as learning rate, hidden layer size and epochs were manually tuned, resulting in 0.03, 12 and 450 respectively. The disadvantages of this tuning method will be explored within the discussion.

[1] AdamW was shown to perform better then Adam with the same hyperparameter settings in the 2019 paper *Decoupled Weight Decay Regularization* from Ilya Loshchilov and Frank Hutter [6]

2.2 Implementation of distinctiveness-based pruning

The pruning method used is outlined in detail by Gedeon & Harris (1991) [2], but was adapted for use on our models. The technique operates by identifying neurons with close to identical or complementary functionality by producing a hidden neuron functionality vector (the activations of a single hidden neuron across the entirety of the pattern space) for each hidden neuron in a layer and calculating the angle between each of them pairwise. These vectors are then normalised to 0.5, 0.5 in the case of Gedeon & Harris' paper in order to achieve an angular range of 0-180°. Hidden neurons with sufficiently small angular separation (15° from either 0° or 180°) become candidates for pruning. For those under 15°, one is removed and the associated weight added to the other. Those pairs over 165° are removed from the network.

The technique applied to our models was adapted in that there was no initial normalisation to 0.5, 0.5 since the purpose of this was to account for the hidden neuron outputs in the paper being bound to a range 0-1, limiting the angular separation calculation to 0-90°. Our hidden neuron outputs are not limited in such a fashion, so the step was removed (normalising standard to the angular separation formula was still applied). In addition to this only similar, not complementary, neurons were considered and removed from the network.

Two removal methods were possible: redefining the model with new topology and new weight tensors imported from the old model post-pruning process, or setting the appropriate 'outgoing' weights (those between the pruned hidden layer and the subsequent layer) to 0, effectively removing their influence on the model. The latter was chosen for our models. The potential advantages and consequences of these adaptations and methods will be discussed later.

The impact this technique had on the network was evaluated for each of the models generated, and the average was taken in a similar fashion as discussed above in order to estimate the expected performance of the pruned model.

2.3 Hyperparameter optimising of leave-one-subject-out models via genetic algorithm

Running an effective genetic algorithm (GA) often requires a relatively large population in order to ensure sufficient initial genetic diversity, and as such is often time intensive to run. The average subject and video independent test (400 fold validation) takes an average of approximately 2 minutes to run, and as a result a GA with a population of 50 across 20 generations would take up to 30hrs to complete. This was impractical given time restraints and a compromise was made, instead testing on the subject independent models and reducing the run time of each test to ~1hr. This was deemed acceptable as the purpose of the research was to investigate the effectiveness of a GA to find optimal hyperparameter settings for a given model, not specifically the video-subject independent models. We also chose to include an optional second hidden layer as a hyperparameter though two sets of tests were conducted: a strictly single hidden layer test for a more fair comparison to the original results, and the optional additional layer tests.

The basic genetic algorithm is described below, discussion and detail of implementation will follow.

Chromosome Structure: [n_hidden1, n_hidden2, activation1, activation2, epochs, learning rate]

1. Initialise a starting population, choosing values randomly from a list of possible values per gene
2. Generate a model and evaluate its accuracy (fitness) for each chromosome in the population
3. Take note of the highest fitness models for elitism
4. Perform a crossover with selected individuals to generate offspring and apply mutation
5. Randomly replace offspring with elite of the previous generation
6. Repeat steps 2-5 for a defined number of generations

Key: n_hidden1 – number of neurons in the first hidden layer
n_hidden2 – number of neurons in the second hidden layer
activation1 – activation function of the first hidden layer
activation2 – activation function of the second hidden layer

2.3.1 General algorithm settings

In each of our tests the following settings were used: *Population*: 50, *Crossover Probability*: 0.9, *Initial Crossover Points:* 3, *Mutation Rate:* 0.02, *Generations:* 20, *Number of Elite*: 3

2.3.2 Chromosome detail

The encoding of the chromosome is simplified in comparison to the traditional binary encoding, instead each gene in the chromosome is from a set of possible values associated with the gene. For example, the n_hidden1 gene is an integer value in range 5-20. The activation functions are mapped from integers that are fed into the network structure, i.e. 0 → leaky ReLU, 1 → tanh, 2 → sigmoid. An example encoding of a chromosome may be: [17, 4, 0, 0, 350, 0.03], and can be read as a network with two hidden layers using the leaky ReLU activation function, possessing 17 and 4 neurons in the first and second layer respectively, trained for 350 epochs at a learning rate of 0.03. The exact detail of the possible gene values can be found in the supporting code file.

2.3.3 Selection methods

A range of methods exist for the selection of chromosomes taking part in the reproduction (crossover) process. In our research we chose to implement two methods, Roulette Wheel Selection, and Rank Selection. Roulette Wheel Selection is a simplistic selection method, where a chromosome is given a probability of being selected by its proportional fitness with respect to the total fitness of the population. This method is effective for simple solution spaces however suffers from premature convergence, meaning the offspring will often trend towards (and get stuck in) a locally optimum solution, thus failing to find the global optimum [7]. In Rank Selection, each chromosome is given a rank according to its fitness and is selected with a probability proportional (whether it be linearly, geometrically etc.) to its rank. In contrast with Roulette Wheel Selection, Rank Selection maintains selection pressure throughout the algorithm, encouraging the solution to go beyond local optimums. However, it suffers from slower convergence as a result [7].

2.3.4 Crossover method, Elitism, and Mutation

In a typical case crossover reproduction is performed either with a defined number of 'points' or randomly, where the resulting mask determines which genes from each parent are selected for the offspring. Hasançebi and Erbatur suggest that the number of crossover points implemented by the algorithm correspond to 'exploration' and 'exploitation' characteristics of the search [8], where a high number of crossover points (3) encourages exploration, and a low number (1) encourages exploitation. Similarly to how learning rate scheduling is implemented in neural networks, a crossover point schedule was implemented into the genetic algorithm. Crossover points decreased from 3 to 2 in the 5^{th} generation, then again to 1 in the 10^{th} generation, leaving 10 remaining generations for exploitation. To ensure no particularly exceptional individuals are lost in the crossover process, an elitism method was used where the fittest three individuals of a generation are copied over to the next generation, randomly replacing offspring. Basic mutation method was also applied, where each gene in the chromosome of the offspring has a chance of changing to a different value from the set of possible values.

3 Experimental results and discussion

3.1 Initial model and network reduction

The goal of the model was to perform a binary classification on pupillary data to determine whether a subject had observed posed or genuine anger. Being a binary classification, a standard of 50% successful classification sets the bar for what would be considered better than a random selection, the next bar being set at 60%, the subject's verbal response accuracy in the original paper [1]. An ideal outcome would be achieving a similar classification success rate as in the original paper (95%) [1].

We conducted training on two different separations of data (leave-one-subject-out and leave-one-subject-and-video-out) and evaluated the performance of these models both pre- and post-pruning.

Table 1. Expected model performance pre- and post-pruning.

	leave-one-subject-out	*leave-one-subject-and-video-out*
Pre-pruning avg. accuracy	**78.38%**	**77.34%**
Post-pruning avg. accuracy	**71.24%**	**55.59%**

These results come from taking the average accuracy of 20 (1 × 20 subjects left out) models and 400 (20 videos × 20 subjects left out) models for leave-one-subject-out and leave-one-subject-and-video-out respectively, and then further taking the average of these averages across 100 full tests to account for the random nature of the weight initialisations during training. We also performed a (non-exhaustive) selection of random seeds in order to explore the models potential, and achieved accuracies as high as 84% in the leave-one-subject-out models, and 82% in leave-one-subject-and-video-out. It was seen that while the models fall short of the accuracy cited in the original paper, they have achieved a higher accuracy than that of the subject's verbal responses (60%).

It should be noted that this form of testing is much more similar to validation than keeping a completely separate training set to assess a model, sharing many similarities with k-fold or leave-one-out validation. A criticism of this is that if hyperparameters are tuned based off the results of these tests, it can be argued that the model is being exposed to the test data in some sense and the results do not provide an accurate performance on unseen data, but rather provide an estimation of the model's performance. We chose this method of testing rather than a traditional split in order to test the model for subject and video independence without sacrificing a significant portion of the data set for training.

It can be seen that after the pruning technique has been applied the models suffer a loss in accuracy, and in the case of leave-one-subject-and-video-out, this loss is significant. A reason for this may be in the applicability of the technique to this data set, however it is more likely to be due to a misinterpretation or oversight on our end resulting in a poor implementation of the technique. It is also possible that some of the 'nuance' of the model is being lost in the merging of hidden neurons, though this averaging effect is said to be insignificant in the original paper [2]. Groups of similar neurons larger than three were also not considered in our implementation, and may have a significant impact on the model when not correctly handled. It is unclear why leave-one-subject-and-video-out models were more impacted, though we speculate that these models have greater sensitivity to slight variations in the weights when predicting against such small test sets.

3.2 Genetic algorithm hyperparameter optimisation

A benchmark for our GA results are set by the results of the leave-one-subject-out model accuracy, 78.38%. With the implementation of our GA, we find hyperparameter settings that differ from our original model and achieve a greater accuracy in the prediction task.

Table 2. Accuracy of fittest model generated by the GA

	Roulette Wheel Selection	*Rank Selection*
Strictly single hidden layer accuracy	**84.25%**	**84.25%**
Possible second hidden layer accuracy	**84.75%**	**85.50%**

3.2.1 Optimisation of strictly single hidden layer models

The GA achieves a significant increase in accuracy compared to the original results using both selection methods. The Roulette Selection Method resulted in a model that used 20 neurons in the hidden layer using a tanh activation function, trained for 350 epochs with a learning rate of 0.04 (figure 1). The Rank Selection method found a similar model, differing only in that it had 18 hidden neurons as opposed to 20 (figure 2). Each individual's fitness of each generation is tracked and the average fitness being indicated in red.

In figure 1, we see that the average fitness trends upwards, with an approximate increase of 7% across 20 generations. However it can be seen that the fittest individual was found in the 4th generation, indicating that this may be a local optimum that the GA was unable to overcome. The possibility of it being the global optimum is ruled out as the previous study [1] showed that an accuracy of 95% was achievable.

A similar trend is seen in figure 2, where the fittest individual was of equal accuracy as in the Roulette Wheel Selection test. However it is apparent that the population is less convergent, showing a greater variation in fitness even in the latter generations. This is likely due to the Rank Selection method which, as stated previously, is known to converge at a slower rate.

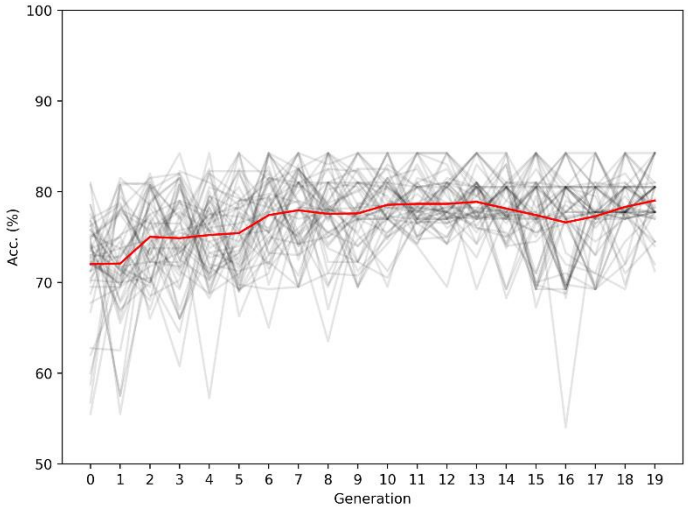

Fig 1. Roulette Wheel Selection, single layer

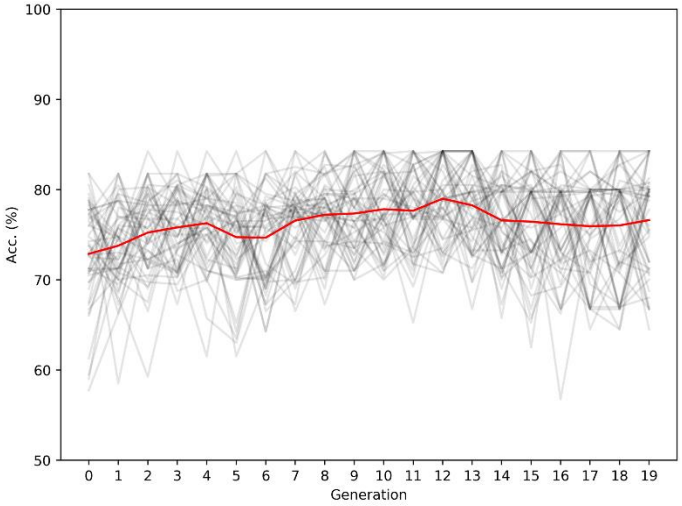

Fig 2. Rank Selection, single layer

3.2.2 Optimisation of one to two hidden layer models

When allowed to implement a second hidden layer into the model, the GA not only finds a more accurate model than the original, but a slightly more accurate model than the strictly single layer GA models. This suggests that the overall model may benefit from the addition of a second hidden layer, possibly due to its complex problem space. The Roulette Wheel Selection method resulted in a model with 17 neurons in the first hidden layer and 4 neurons in the second hidden layer, both using a leaky ReLU activation function, trained from 350 epochs at a learning rate of 0.03 (figure 3). The Rank Selection method resulted in a model with 18 neurons in the first hidden layer and 4 neurons in the second hidden layer, both using a leaky ReLU activation function, trained from 400 epochs at a learning rate of 0.035 (figure 4). Similar to the single layer testing, each individual's fitness for each generation is tracked, and the average fitness is indicated in red.

In figure 3 we see a similar trend as in figure 1, the main difference being that the peak fitness is slightly higher. We also see an increase in average fitness of approximately 6%, again similar to the single layer test counterpart. These similar results suggest that the addition of a second hidden layer does not solve previous tests issue of getting stuck in a local optimum, and that the approach of the GA could be improved.

Figure 4 shows the same trends as in figure 2, its single layer test counterpart. The average fitness increases slightly less with the Rank Selection method, and the variance in fitness is greatly increased across all generations. However this method does manage to find the overall fittest individual across all tests. This may be due to the additional selective pressure of the Rank Selection method, though the improvement is minimal so it cannot be ruled out that the improvement occurred randomly via mutation or simply luck.

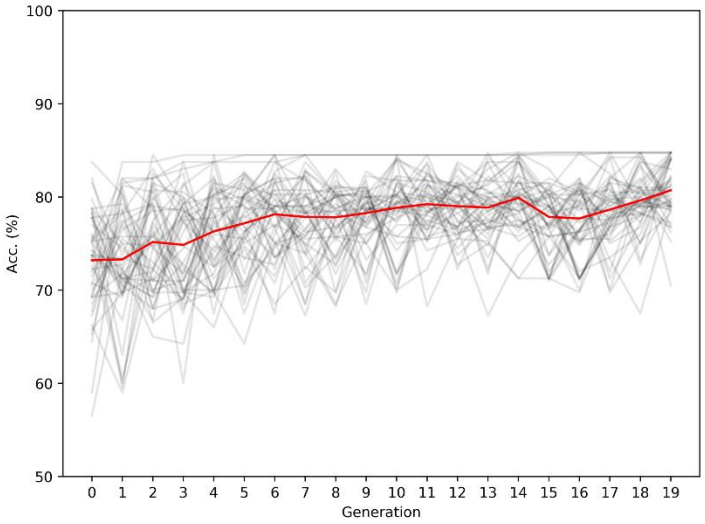

Fig 3. Roulette Wheel Selection, one or two layers

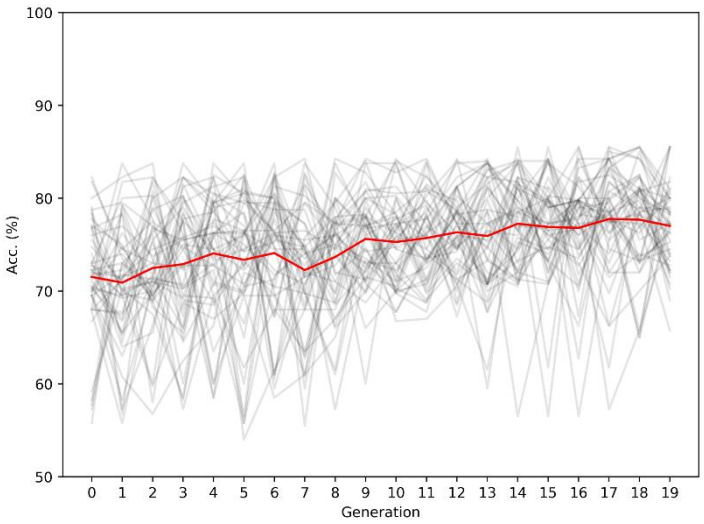

Fig 4. Rank Selection, one or two layers

4 Conclusion and future work

We showed that a simple feed forward neural network is capable of learning to identify observed posed or genuine anger to a higher degree than the observer's verbal response, using the pupillary patterns of the observer. It was further demonstrated that these models could benefit from hyperparameter optimisation via a genetic algorithm. Additionally, it was also shown that in some cases, the accuracy of a network is somewhat resistant to pruning based upon the identification of similarly behaving neurons in the hidden layer.

It is clear that there is room to improve in each of the techniques explored in this research, as none met the results of previous studies conducting similar techniques. Some areas to explore with respect to the base model may be looking to extend the networks depth by introducing a secondary hidden layer, allowing the network to more easily learn complex decision boundaries. Further investigation into the consequences of the 'validation style' testing may prove informative in order to push the network towards an accuracy shown to be achievable in Chen et. al's original work with this dataset [1].

A fuller implementation of the pruning technique may also give some insight into the shortcomings of its application on the models.

Further testing of different selection and crossover techniques for the genetic algorithm may yield more favourable results, and an investigation into the application of these techniques on similar problems may prove useful. However overall we achieved beyond our initially stated benchmarks, and while further improvements could be made to the implementations, this work demonstrated the potential utility of applying a genetic algorithm for the purpose of neural network hyperparameter optimisation.

References

1. Chen, L., Gedeon, T., Hossain, M.Z., Caldwell, S. (2017) Are you really angry? Detecting emotion veracity as a proposed tool for interaction In: Proceedings of the 29th Australian Conference on computer-human interaction, Brisbane Nov 28 – Dec 1
2. Gedeon, T.D., Harris, D. (1991) Network Reduction Techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 1: 119-126
3. Isik, F., Ozden, G., Kuntalp, M. (2012) Importance of data preprocessing for neural networks modeling: The case of estimating the compaction parameters of soils. EEST Part A: Energy Science and Research 29(2):871-882
4. Hossain, M.Z., Gedeon, T. (2017) Classifying Posed and Real Smiles from Observers' Peripheral Physiology In: PervasiveHealth '17: Proceedings of the 11th EAI International Conference on Pervasive Computing Technologies for Healthcare, May 23-26, ACM, Barcelona vol. 1:460-463
5. Kingma, D.P., Ba, J.L. (2015) Adam: A Method for Stochastic Optimization. Paper presented at the International Conference on Learning Representations, San Diego, May 7 – 9
6. Loshchilov, I., Hutter, F. (2019) Decoupled Weight Decay Regularization. Paper presented at the International Conference on Learning Representations, New Orleans, May 6 – 9
7. Saini, N. (2017) Review of Selection Methods in Genetic Algorithms. International journal of engineering and computer science vol. 6: 22261-22263
8. Hasançebi, O. Erbatur, F. (1999) Evaluation of crossover techniques in genetic algorithm based optimum structural design. Computers and structures, vol. 78: 435-448