Optimising CasPer Network for Classification using a Genetic Algorithm: Determining Depression Severity

Edmund Hofflin[1]

Research School of Computer Science, The Australian National Univeristy
u6373930@anu.edu.au

Abstract. The Cascade-Correlation algorithm was shown to be an effective and useful technique for training neural networks. However, its performance was problematic in some settings. In response to this, the CasPer algorithm was proposed. It and its improvements were proven to be more effective, building more compact networks and solving regression problems. However, little work has been done to investigate its performance in classification problems. We took a CasPer variant - Layered CasPer with SARPROP - and tested its ability to gauge depression levels based on human physiological reactions. Setting effective hyperparameters proved to be a very difficult task, so a genetic algorithm was used to optimise them. The genetic algorithm performed well, converging on a minimum and reducing the network error. However, even with optimised parameters, all networks performed equally poorly, rendering any substantial conclusion impossible.

Keywords: SARPROP · Layered CasPer · Neural Network · Machine Learning · Depression

1 Introduction

The Cascade-Correlation algorithm (Cascor) [4] was one of the first constructive algorithms that built neural networks whilst simultaneously training them. It performed well, allowing the network structure to adapt its topology to the problem. However, it wasn't a universal solution, having issues with some classification problems [7]. To overcome these issues, the algorithm was adapted to employ Progressive Back Propagation (RPROP), resulting in the new CasPer algorithm [15]. This algorithm also used the cascading structure and method of adding units progressively, but removed the weight freezing technique in favour for a variable learning rate RPROP, allowing the new units to learn quickly while also allowing the older units to adapt gradually. The learning rates were also reset when a new neuron was added, giving older units even more opportunity to adapt to the old network. These changes aimed to prevent the first units from being poor feature detectors, as is sometimes the case in Cascor. The result was positive, as CasPer was shown to overcome the regression issues, performing well at the notoriously difficult two spiral problem.

Since then the CasPer algorithm has been improved. Most notably, the Simulated Annealing method, that has been shown to remove unneeded connections [13], was incorporated into RPROP. This led to the new Simulated Annealing RPROP (SARPROP) algorithm [16] that increased performance. Finally and more recently, another improvement was made with Layered CasPer [14] by moving away from the cascade structure to a more standard network structure: new units didn't always create a new layer but were added to the current layer, up to a maximum number.

Throughout these improvements, CasPer and its descendants were shown to perform well, overcoming many of the issues that Cascor had with regression. However, no research has been done to investigate whether CasPer overcomes the classification issues. Thus, we seek to apply a new CasPer algorithm to a classification problem and determine whether it has overcome the issues that Cascor faced.

To test CasPer's classification ability, we used an already solved classification problem. We chose data that examines depression levels based on human physiological reactions, which were classified with over 80% accuracy by a standard, fully connected, single hidden layer neural network [19]. Given the simplicity of the neural network used to classify the data originally, the problem should prove an easy challenge for the CasPer, opening the path to further investigation with more difficult problems.

CasPer, along with many neural network structures and algorithms, has many hyperparemeters that should be changed to reflect the

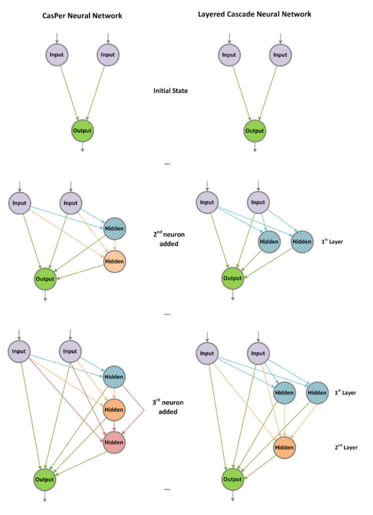

Fig. 1: Standard CasPer Algorithm vs. Layered CasPer with 2 Unit Per Layer Maximum

problem. Often though, the problem space is relatively unknown, thus the setting of hyperparameters becomes a process of trial and error. In order to overcome this issue, the process of choosing hyperparamaters is considered a problem in and of itself and so standard optimisation methods are used [2]. We decided to use a genetic algorithm, which has been shown to work in the past with complex deep learning structures [18].

2 Method

There were three distinct parts to our method: data production/acquisition and processing; optimising hyperparamters for a standard neural network model and a CasPer, or newer variation, algorithm using a genetic algorithm; and, a final validation and confirmation of results. We detail the decisions and methods surrounding each in turn.

2.1 Data and Processing

Data Production and Acquisition The data was produced by a previous investigation of depression detecting through physiological reactions [19]. The paper goes through in depth the experimental design and data recording. Here, we only give a brief overview so that later decisions and results have context.

Videos from the 2014 Audio-Visual Emotion Challenge (AVEC 2014) dataset [17] were used as stimuli. These videos were recordings of individual participants, either reading a paragraph or answering a set of questions. Each participant self-reported a depression index, by completing the Beck Depression Inventory - II [1]. The depression index, ranging from 0 to 63, can then be used to produce one of four depression labels: no or minimal depression $(0 - 13)$; mild depression $(14 - 19)$; moderate depression $(20 - 28)$; and, severe depression $(29 - 63)$. These depression levels were then used as the depression label for each video.

From the 300 video dataset, 16 videos were chosen for the experiment. These videos had similar durations and were uniformly spread across the depression levels.

Fourteen participants were chosen to watch these videos. We note that no participant understood the German being spoken in the videos and additionally had no prior knowledge about depression identification. Two participants were excluded due to technical issues. Even after this exclusion, the group was balanced in sex. The final group size of twelve was deemed viable upon comparison to standard preliminary studies in medicine.

Each participant watched all videos, with small gaps between. During the experiment, participants were attached to sensors that recorded three different physiological responses:

(i) Galvanic Skin Response (GSR): A measure of the electricity flow through the skin, which varies due to sweating responses, and has been shown to correlate with increasing mental activity [6].
(ii) Pupillary Dilation (PD): Pupil dilation has been linked to changes in mental state [11], as well as positive stimuli [12].
(iii) Skin Temperature (ST): Has previously been show to be negatively correlated with stress and fear [8].

The data from these sensors was then attributed to the video being shown at the time. From this raw data, various different features were collected; specifically, per physiological response: 23 for GSR, 39 for PD and 23 for ST.

Three different datasets were created, one for each physiological response, and each with 192 data points, one for each combination of the 12 final participants and 16 videos.

Data Processing Before being fed into any model, the raw data was processed.

Firstly, the original paper [19] tested the model performance on four different dataset combinations: GSR + PD + ST, GSR, PD and ST. We implemented methods for each combination to be tested, but our results will focus on the combined dataset, GSR, PD and ST.

Secondly, the raw data was processed using a min-max scaler. This was done to ensure that all values were between 0 and 1, preventing any misinformation about relative data importance. This was also done in the original paper.

Finally, in order to validate the results effectively, we employed cross-validation methods. The dataset was quite limited, so we wished to use Leave-One-Out cross-validation in order to maximise the training dataset size. However, this is impossible given the data: each participants entire dataset is needed as it forms one continuous data representation. Thus, we use the next best option: Leave-One-Participant-Out cross-validation [10]. This practice is similar to Leave-One-Out cross-validation, but instead of using one single data point as the test set and cycling through, one participant and their associated data is used as the test set and the participants are cycled through. Thus, we end up with 12 sets of train/test pairs that we can do cross-validation on.

2.2 Optimising Hyperparameters

There were two distinct parts to our method of optimising hyperparameters. First and foremost, we had to develop models for a standard neural network and CasPer algorithm whose hyperparameters could be optimised. Secondly, we developed a genetic algorithm to optimise those parameters. We will cover each of the steps and the two model designs separately.

Developing Network Models: Standard Neural Network In order to validate the data and form a good basis of comparison, we attempted to replicate the results in the paper [19]. Thus, we followed the paper and constructed a standard, fully connected neural network model. Each network had a single hidden layer and used the sigmoid activation function for these hidden units. As required by the data, the output layer was 4 units. The original paper used networks with 50 hidden neurons. We didn't follow this guideline, but instead allowed the number of hidden neurons to be a hyperparameter. However, we did follow the training methods in the paper, using the Adam optimiser [9] for back propagation and the Cross-Entropy loss function.

As is standard, the result for each test was averaged across the cross-validation runs. Additionally, we ran each test 5 times and averaged those results. Thus, the final result of the network was reported as the average loss and the average accuracy of classification.

This model has three hyperparameters:

(i) Number of Hidden Neurons: The amount of neurons in the single hidden layer ($[1, 100]$).
(ii) Learning Rate: The Adam optimiser uses one learning rate ($[0.00001, 0.5]$).
(iii) Number of Epochs: The number of epochs for training ($[10, 2512]$).

The parameters were optimised by the genetic algorithm.

Developing Network Models: Layered CasPer with SARPROP We chose to use the latest CasPer network, given that each improvement should further the network's ability to solve classification problems. Thus, we opted to construct Layered CasPer networks that used the SARPROP optimiser.

Both layered CasPer and SARPROP have many different parameters that can be tweaked and varied to suit the problem. Thus, we have a long list of hyperparameters that can be optimised:

- Learning Rate 1: The learning rate for connections feeding into the newest neuron ($[0.05, 1]$).
- Learning Rate 2: The learning rate for connections leading out of the newest neuron ($[0.001, 0.05]$).
- Learning Rate 3: The learning rate for all other connections ($[0.0001, 0.001]$).
- Eta Up: The multiplicative factor for increasing the learning rates ($[0.1, 0.9977]$).
- Eta Down: The multiplicate factor for decreasing the learning rates ($[1.002, 5]$).
- CasPer Epoch Parameter: The network is given $15 + P \times N$ epochs to train the new neuron, where N is the number of neurons and P is the CasPer epoch parameter ($[1, 100]$).
- CasPer Error Parameter: Training is halted if the error hasn't decreased by this multiplicative factor within the given epochs ($[0.9, 0.99999]$).
- Hidden Layer Max: The maximum number of neurons in each hidden layer ($[1, 10]$).
- Weight Decay: The weight decay constant used in the simulated annealing equation:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial w_{ij}} - k \times \operatorname{sign}(w_{ij}) \times {w_{ij}}^2 \times 2^{-0.01 \times E_p}$$

  where E_p is the number of epochs since the last neuron was installed, and k is this parameter ($[0.00001, 0.01]$).

The values after each parameter outline designate the ranges allowed for each parameter. The only possible hyperparameter that we didn't vary was the activation function for the hidden neurons. Instead we followed previous work [15] and the standard network model and used the sigmoid function.

Before using the genetic algorithm, we tried to optimise a subset of the hyperparameters by setting possible values and cycling through all possibilities and reporting the best combination. The hyperparamaters and their values were:

- CasPer Epoch Parameter: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Hidden Layer Max: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Weight Decay Constant: $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$

This was done in order to evaluate the effectiveness of the genetic algorithm: if a simpler approach yields the same results then the genetic algorithm would be redundant.

Genetic Algoithm In order to optimise the hyperparameters of the two neural network models, a genetic algorithm was constructed. The algorithm is fairly standard, following the underlying structure of the canonical genetic algorithm. However, the specifics were tweaked to fit the problem.

Significant changes were made to minimise the computational requirements of the algorithm: the fitness evaluation already required training and testing neural networks, which is very computationally expensive, and thus prevents some methods and techniques from being used. Firstly, the canonical model uses (λ, μ) selection: it chooses the next population only from the offspring. Instead, we used $(\lambda + \mu)$ selection with an offspring population size of 20%. Secondly, the fitness of each chromosome was only calculated once and then stored. To overcome the natural noise, each chromosome was evaluated 5 times and the result averaged. Finally, the chromosome is not encoded as bitstrings, but instead tuples of each parameter. This was done as each parameter has very specific bounds and the accuracy needed for some parameters is very high, thus a bitstring representation would be incredibly long and difficult to decipher.

The functional specifics of the genetic algorithm were constructed to cover as many settings as possible: no information was known about the search space, so the functions were designed to explore as much as possible early in the process and then focus only on exploiting the minima in the later generations. Thus, the following functions were used:

- Initialisation: A standard approach that uses random uniform sampling over the entire space.
- Mutation: Each chromosome and each gene has its own chance for mutation. Mutation of a gene was performed by sampling from a Gaussian distribution centered on the original value and with a standard deviation equal to the square of the distance to the nearest bound. The chance of mutation was gradually decreased over time.
- Reproduction: Two offspring were created from two randomly selected parents. The offspring's values used Simulated Binary Crosover (SBX), which has proven effective in the past [3].
- Selection: A standard approach using proportional Boltzman, with universal stochastic sampling. To reduce the selective pressure, no replacement was allowed.
- Fitness: As stated above, the fitness method trained and tested the neural network with the hyperparamaters five times. The fitness is equal to $\left((1.001 - \text{accuracy}) \times \text{loss}\right)^2$. We used 1.001 instead of 1 so that the accuracy term could never equal 0, rendering the loss term insignificant.

Additionally, we let the population size slowly decrease over time, so that early generations could search more but not inflate the computational requirements over time.

The algorithm produced four values:

(i) Average Fitness List: the average fitness of each generation.
(ii) Minimum Fitness List: the minimum fitness of each generation.
(iii) Best Chromosome: the chromosome values that had the lowest fitness.
(iv) Minimum Fitness: the fitness of the best chromosome.

In order to validate the genetic algorithm, we first tested it using the standard network model and a glass identification dataset [5]. This also allowed us to tweak the parameters of the genetic algorithm, optimising it to perform well on classification problems, before we ran it on the harder problems.

2.3 Final Test and Result Confirmation

We ran the functioning genetic algorithm for optimising network hyperparamters on both models: the standard feed-forward, fully connected, single hidden layer neural network model and the layered CasPer with SARPROP model.

After the genetic algorithm, the best chromosome was validated by evaluting its fitness 10 more times. Additionally, the loss and average were separated out so that they could be analysed.

3 Results and Discussion

We first analyse and evaluate the results of the standard neural network model, before moving onto the Layered CasPer with SARPROP algorithm. Finally, we compare and discuss both results.

3.1 Standard Neural Network

The results of the genetic algorithm were promising, showing clear convergence towards a minimum:

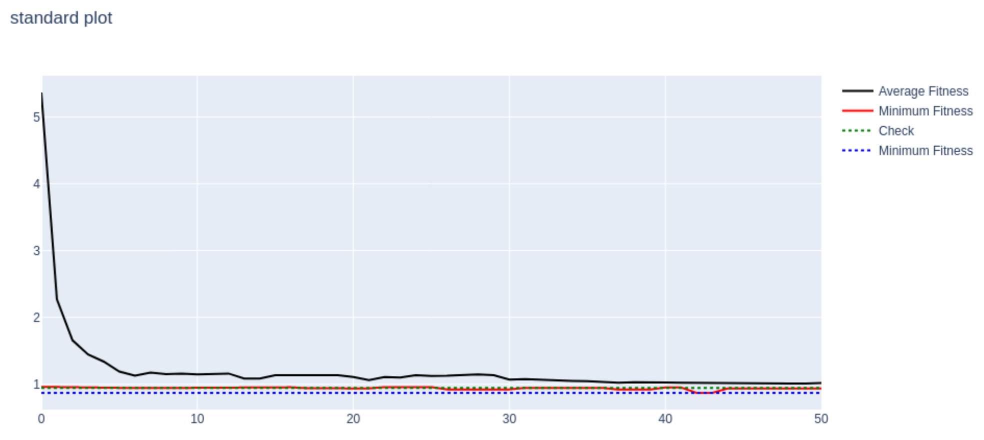

Fig. 2: Standard Model: Average and Minimum Fitness vs Generation

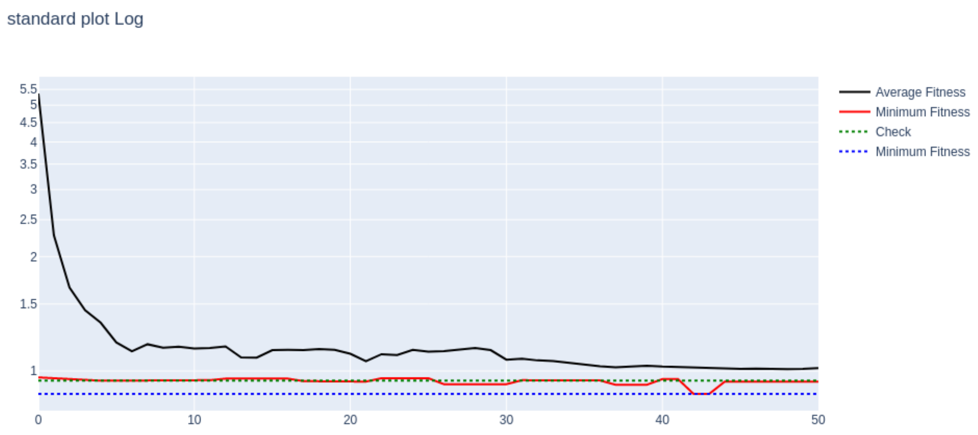

Fig. 3: Standard Model: Average and Minimum Fitness vs Generation (logarithmic)

Note that the ‘check’ value is the validation result, showing how the best chromosome performed when evaluated 10 more times. As expected due to the noise, the ‘check’ fitness is slightly higher. However, the validation process also yielded a disappointing discovery: the network with optimised hyperparameters only achieved an accuracy of 29.22%. This was far lower than anticipated, especially considering the significant optimising done by the genetic algorithm.

3.2 Layered CasPer with SARPROP Results

As noted, before the genetic algorithm was run on the CasPer model, we attempted to optimise the subset of hyperparameters - Casper Error Parameter, Hidden Layer Maximum, and Weight Decay - by exhaustively searching through all combinations of preselected fixed values. After running this test 50 times, we got the following results:

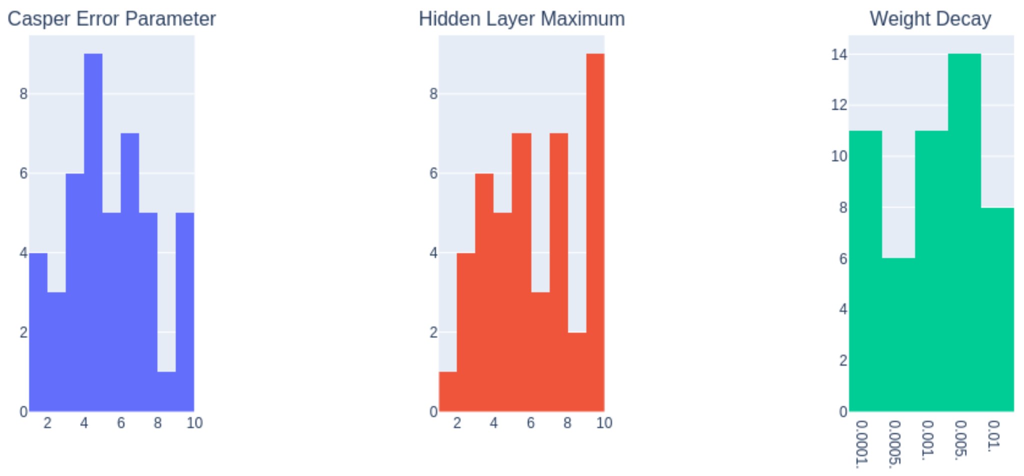

Fig. 4: CasPer Model: Histograms of the Best Parameters over 50 Runs.

These histograms are rather chaotic: no value for any parameter performed significantly better than any other, providing no insight into the optimal hyperparameter values.

In light of these results, the genetic algorithm performed extraordinarily well:

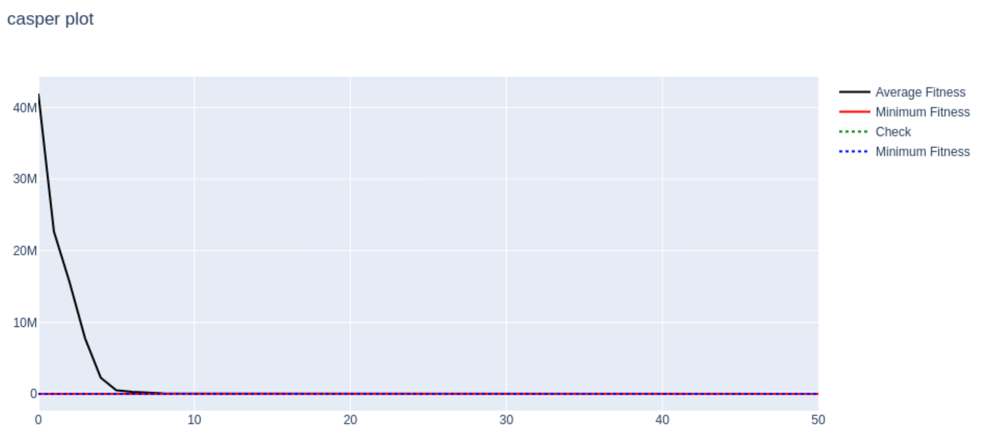

Fig. 5: CasPer Model: Average and Minimum Fitness vs Generation

The convergence continued for the entire duration, improving the initial average by several orders of magnitude:

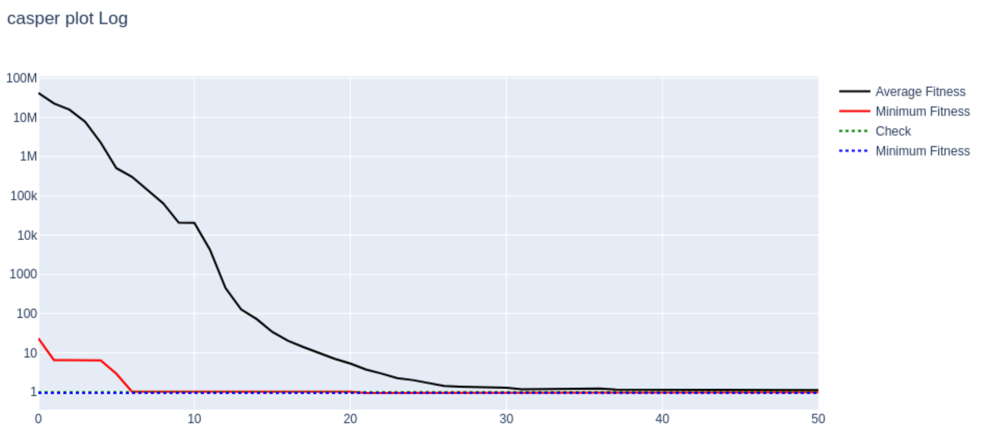

Fig. 6: CasPer Model: Average and Minimum Fitness vs Generation (logarithmic)

The reduction in the fitness was even more significant than the standard model. Once again however, the validation step produced poor results: the best chromosome had an accuracy of 29.17%.

3.3 Comparison and Discussion

The accuracy of both models was bad, only slightly above guessing. This was poor in and of itself, however it is dismal when contrasted to the 88% accuracy that the original paper achieved [19]. Not only is this disappointing, but it is also quite surprising. As discussed, the data is sourced from, and then processed in an identical, or near identical, manner to the original paper. Thus, we would expect the standard network, being the same model, to have at least approximately the same accuracy. Futhermore, the original paper didn't outline any method for optimising the network hyperparameters, so we would expect the hyperparameters extensively optimised by the genetic algorithm to result in better performance. Therefore, the results are disappointing and very difficult to explain.

Setting aside the results from the original paper, both the standard model and the layered CasPer with SARPROP algorithm had incredibly similar performances, with only a 0.04% absolute difference in accuracy. Given this, we could surmise that the layered CasPer with SARPROP algorithm performs just as well as standard network models. However, as the accuracy is so poor, a proper conclusion could potentially be quite flawed.

Overall, the only positive outcome was the performance of the genetic algorithm. It optimised its fitness incredibly well, reducing the average fitness by a magnitude of 8 in the CasPer case. Additionally, the convergence continued as the algorithm progressed, with little variance. So the genetic algorithm is robust and effective.

4 Future

Given the null result for the effectiveness of CasPer networks on classification problems, any future work will have significant difficulties building on these results. That said, there are two avenues that should be explored.

Firstly, further research should be done to answer the question of whether CasPer and its descendants do solve the classification issues of Cascor. We put particular emphasis on this as CasPer has proven itself in many other areas, and so it would be very useful if this success transfers to classification problems. The models and methods used in this paper can be employed for this, making the process easier, but a new dataset should be used.

Secondly, the genetic algorithm did prove to be very effective. Thus, an in-depth analysis into its effectiveness on optimising network hyperparameters should be conducted. Network hyperparameters are notoriously difficult to set and optimise, so a good and robust technique would be very useful.

5 Conclusion

The goal was to assess whether either CasPer or one of its descendants had overcome the issues that Cascor faced with classification problems. Unfortunately, no progress was made in relation to that goal, as the both standard network and Layered CasPer with SARPROP networks failed to perform well to identify depression levels based on human physiological reactions. Our only conclusions are that more work should be done to investigate the issue, and that genetic algorithms are effective for optimising hyperparamters.

References

1. Beck, A.T., Steer, R.A., Brown, G.K., et al.: Beck depression inventory-ii. San Antonio **78**(2), 490–498 (1996)
2. Bengio, Y.: Gradient-based optimization of hyperparameters. Neural computation **12**(8), 1889–1900 (2000)
3. Deb, K., Agrawal, R.B., et al.: Simulated binary crossover for continuous search space. Complex systems **9**(2), 115–148 (1995)
4. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture. In: Advances in neural information processing systems. pp. 524–532 (1990)
5. German, B.: Glass identification dataset, 1987. URL http://archive. ics. uci. edu/ml/datasets/Glass+ Identification
6. Hossain, M.Z.: Observers galvanic skin response for discriminating real from fake smiles (2016)
7. Hwang, J.N., You, S.S., Lay, S.R., Jou, I.C.: The cascade-correlation learning: A projection pursuit learning perspective. IEEE Transactions on Neural Networks **7**(2), 278–289 (1996)
8. Kim, J., André, E.: Emotion recognition based on physiological changes in music listening. IEEE transactions on pattern analysis and machine intelligence **30**(12), 2067–2083 (2008)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Kukolja, D., Popović, S., Horvat, M., Kovač, B., Ćosić, K.: Comparative analysis of emotion estimation methods based on physiological measurements for real-time applications. International journal of human-computer studies **72**(10-11), 717–727 (2014)
11. Laeng, B., Sirois, S., Gredebäck, G.: Pupillometry: A window to the preconscious? Perspectives on psychological science **7**(1), 18–27 (2012)
12. Partala, T., Surakka, V.: Pupil size variation as an indication of affective processing. International journal of human-computer studies **59**(1-2), 185–198 (2003)
13. Pham, D., Karaboga, D.: Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks. Springer Science & Business Media (2012)
14. Shen, T.: Layered cascade artificial neural network (2011)
15. Treadgold, N.K., Gedeon, T.D.: A cascade network algorithm employing progressive rprop. In: International Work-Conference on Artificial Neural Networks. pp. 733–742. Springer (1997)
16. Treadgold, N.K., Gedeon, T.D.: Extending casper: A regression survey. In: ICONIP (1). pp. 310–313. Citeseer (1997)
17. Valstar, M., Schuller, B., Smith, K., Almaev, T., Eyben, F., Krajewski, J., Cowie, R., Pantic, M.: Avec 2014: 3d dimensional affect and depression recognition challenge. In: Proceedings of the 4th international workshop on audio/visual emotion challenge. pp. 3–10 (2014)
18. Young, S.R., Rose, D.C., Karnowski, T.P., Lim, S.H., Patton, R.M.: Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments. pp. 1–5 (2015)
19. Zhu, X., Gedeon, T., Caldwell, S., Jones, R.: Detecting emotional reactions to videos of depression. In: IEEE International Conference on Intelligent Engineering Systems (2019)