Neural Structure Optimization Algorithm on Depression Level Detection: Casper or Evolutionary Algorithm based Feed-forward network? *

Yan Yang[1]

Australia National University
u6169130@anu.edu.au.com

Abstract. The depression is interesting at the emotional level and can be physically sensed by others. Meanwhile, the neural network can effectively detect the depression level by using the statistical summary of observers' physiological signals, and [CJ19] achieves 92% accuracy. Cascade network algorithm employing Progressive RPROP (Casper) [TG97] enables self-optimized neural network structures, where the network topology starts from minimal. In feed-forward neural networks, the structures of networks need to be defined, which usually relies on the experience. To aid this, we use evolutionary algorithm to optimize the structure of feed-forward neural network and the hyper-parameters. In this paper, the two approach are applied to a multi-classification problem, detection of different depression levels. By large number of experiments, the Casper algorithm outperforms evolutionary algorithm optimized feed forward neural networks by 4.26% accuracy.

Keywords: Neural Network · Emotion Recognition · Depression detection.

1 Introduction

Depression heavily affects people's daily life and is a common mood disorder illness [Se15]. Constant feelings of sadness and loss of interest or enthusiasm on normal daily activity are to be considered as depression. In the worst case, it can trigger suicidal behaviors. In medical science, depression is not only a low mood but serious conditions that influence people's physical and mental health. [CJ19]The good news is the effective treatment of depression are exists. However, it passes the buck to the detection of depression and it's level. While different types of depression exist but can be simply classified by the degree of depression. The manual diagnosis of depression is subjective and time-consuming while relying on the patients' subjective feedback. It increases uncertainty and decreases the accuracy in the measurement of depression. [CJ19] It is necessary and urgent to propose a reliable depression detection system to assist the doctors in related treatments.

1.1 Motivation

Accompany the improvement in the physiological computing technologies, it has been proofed as a powerful auxiliary means in the depression diagnosing systems. [Ce16]. The paper investigate the performance of two different neural network architecture optimization algorithm, Casper and evolutionary algorithm optimized feed-forward neural networks on depression level detection by using the statistical summaries of observer's physiological signals. The Caspar algorithm are designed to adaptively optimize the network topology, where the structure of the network will be learned during the training. The hyper-parameters are much more less compared to the traditional feed-forward neural networks. However, the structure of feed-forward neural networks are predefined before training. Usually, large number of experiments and sophisticated experience are required to tune a well performed network structures and hyper-parameters. Thus, we use evolutionary algorithm to aid the shortcomings. The paper investigate the performance comparison between the two neural structure optimization techniques on depression level detection tasks, also the comparison between the state of the art models are provided.

1.2 Database

In our experiments, the three databases are collected by [CJ19] has been used. Each of which is collected by using the different physiological signal sensor. They are Galvanic Skin Response (GSR), Skin Temperature

* Supported by organization of COMP4660 Bio-inspired course group

(ST) and Pupillary Dilation (PD). [CJ19] For the sake of simplicity and clarity, they will be referred as GSR database, ST database, and PD database respectively in the later chapters. Each of the databases has 192 instances from 12 subjects. GSR database, ST databases, and PD databases contain 23,39, and 23 features respectively. The *(1) minimum, (2) maximum, (3) Mean, (4) standard deviation, (5) variance, (6) root mean square, (7)(8) means of absolute values of first/second different, and (9)(10)(11) number, amplitude and ratio of particular signals* are used as the statistical summary for each of the signals. Features are extracted twice based on normalized and unnormalized signal respectively. The age of participant are spread from 18 to 27 with an average 21.1 and standard deviation of 2.8 [CJ19], and it has a perfectly balanced distribution on gender. The distribution for the target class of each database is equally spread over each of the target classes. There are 4 investigated depression categories:

- *0 indicates no or slight depression*
- *1 indicates middle-level depression*
- *2 indicates strong level depression*
- *3 indicates grievous level depression*

2 Methodology

The section provides the theory behind our models. The overview of Casper algorithm and evolutionary algorithm are provided, as well as the evaluation metrics are introduced. The introduction of feed forward neural network is omitted.

2.1 Casper Algorithm

The Casper [TG97] is a constructive algorithm, which is designed to overcome the bad generalization problem of Cascor[BF94], a similar cascade neural network architecture. Compare with the traditional neural network, Casper starts with minimum network topology, then automatic train and add the new hidden units one by one. Typically, it could learn much more quickly than the traditional feed-forward neural networks.[BF94]. The networks start with minimal typology which does not contain any hidden layers. Then, the network generates the candidate units which fully connected with all input and existing hidden units, and maximize the correlation between the neurons connected to it and the residual error. Note, the residual errors refer to the error between prediction and target in current status. In the next steps, add the candidate units with maximum correlation to the residual error of networks, and keep the weights. Based on the time to add weights, it maintains 3 different learning rates as shown below and demonstrated in figure 1.

- L1 rates: new added neuron that connect the latest hidden units with all input and the rest hidden units
- L2 rates: the neuron that connect the latest hidden units with output
- L3 rest of neurons

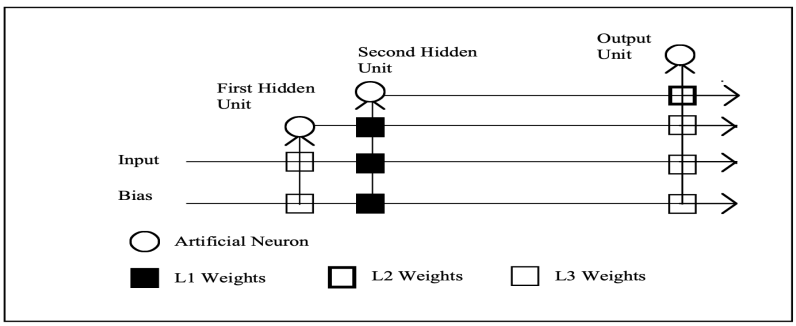

Fig. 1. The structure of Casper [TG97]

Conventionally, the L1 is much larger than L2 and L3, and L2 is larger than L3. More importantly, the Casper [TG97] improves the generalization by using the Resilient Backpropagation (Rprop) algorithm with simulated annealing weights decays as

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial W_{ij}} - D * sign(w_{ij} * w_{ij}^2 * 2^{-T*Hepoch})$$

where D is a hyperparameter, that usually known as decay parameter in the modern optimization algorithm. The overview of the Caspar algorithm is shown in pseudocode Algorithm 1 and Algorithm 2.

Algorithm 1 Cascade network algorithm employing Progressive RPROP (Casper)

Input: *threshold, max_iter,hpatience, hmax_iter*
net = net_initialize() ▷ A minimal network only consisted by input and output layer
for iter in max_iter **do**
    cLoss = net.train() ▷ use RPROP optimizer with Simulated Annealing weight decays
    if cLoss < threshold **then**
        Break
    end if
weights = addHidden(cLoss,hpatience,hmax_iter) ▷ Use algorithm 2 to calculate weights for new hidden units
net.addWeights(weights) ▷ maintain three container that represents 3 different learning rates
net.update() ▷ update learning rates by updating the aforementioned container
end for

Algorithm 2 Add new hidden units to Casper

Input: *cLoss,hpatience,hmax_iter*
Output: *weights*
net = newNet(candidate) ▷ all inputs and current hidden units fully connected with candidate units
X = init() ▷ all inputs and current hidden units
previous = INF
for iter in max_iter **do**
    y = net(X)
    corr = Correlation(cLoss,y) ▷ calculate correlation
    corr.backward() ▷ updates the net by backpropagation to maximum the correlations
    current = sum(corr)
    if previous - current < hpatience **then**
        Break
    end if
    previous = current
end for
idx = argmax(ss) ▷ select the most correlated candidates
weights = net.weights[idx,:]

By the nature of linear algebra, the linear combination of linear equations is still linear. Thus, we add different activation functions to the Caspar algorithm in our experiments. The Dropout[Se12] and BatchNormalization [IS15] are used. The details of experiments will be showed in next section.

2.2 Evolutionary Algorithm

Specifically, we use the generic evolutionary algorithm to optimize our neural network structures and hyper parameters. Generic evolutionary Algorithm mimics the natural selection for randomly chosen populations of individuals, to stochastically search for the optimal solutions for given problems. [Eng07]. Firstly, we define the domain of structures that will be used in the search. Then, we uniformly and randomly initialized several model structures and the hyperparameters used for training. In the context of evolution, each of them is know as individuals, and the group is the populations. The fitness of each candidate model\individuals will be the accuracy on the depression dataset. Then, we use tournament selections, uniform crossover and uniform mutation to breed the next generations. The details are shown in algorithm 3. The hyper parameters, crossover method, mutation method and selection method will be justify in the next section.

Algorithm 3 Generic Evolutionary Algorithm for Feed-forward Neural Networks Optimization

Input: *maxGenerations,num,crossover_rate, mutation_rate*
t = 0 ▷ t records the current generations
p = initializePopulations() ▷ Random initialization of networks structures and hyper-parameters
set(mutation_rate)

```
set(crossover_rate)
set(num)                          ▷ the number of candidate that used in tournament selection method
while t < maxGenerations or stopping criteria not meet do
    fit = p.fitness()             ▷ we use accuracy to represent the fitness value
    new_populations = []
    while length(new_populations) < length(p) do
        child = tournament_selection(p).uniform_crossover(tournament_selection(p))
        child.uniform_mutate()
        new_populations.append(child)
    end while
    p = new_populations
end while
```

2.3 Feature Selection

As we mentioned before, the features contain the statistical summary both from normalized and unnormalized signals. It result in certain number of redundant features. Meanwhile, not every statistical summary of signals are relevant to the tasks. To deal with this issue, we adopt Min Redundancy Max Relevance (mRMR)[LD05] to select the most discriminative features. Let S be the set of selected features, and U be the set of candidate features, then the next selected features will be

$$arg \max_{f_j \in U} \left[I\left(f_j, c\right) - \frac{1}{|S|} \sum_{f_i \in S} I\left(f_j, f_i\right) \right]$$

where I stands for the mutual information. During the training of the model, the most discriminate features sets are selected based on training data, and the number of selected features are tunned by the validation errors.

2.4 Evaluation

For the sake of fair comparison, we will adopt leave one participant out cross validation strategy which is the same validation method as [CJ19]. For each iteration in cross validation, this method use all the data for one human on validation, and training on the rest of data. To validate the effectiveness of the Casper algorithm, we use the *precision, recall, accuracy*. And, we use accuracy to evaluate the evolutionary algorithm optimized feed-forward neural network models. *Precision* is defined as the percentage of instances that are correctly predicted for depression level L over the instances that have the depression level L. *Recall* is the proportion of instances that are correctly predicted with depression level L over all instances that have depression level L. *Accuracy* is defined as the proportion of instances that are correctly predict with depression level L or non-depression level L.

3 Results and Discussion

All of the experiments are conducted on MacBook Pro 2015 with 2.7 GHz Dual-Core Intel Core i5 processor and 8 GB 1867 MHz DDR3 memory. The system versions is 10.15.3 (19D76). The best performed Casper model [TG97] and evolutionary algorithm optimized feed-forward neural networks are provided in this section. Note, the GSR databases, PD databases, and ST databases contain the same instances. And, all features denote concatenating these features into a single feature with the corresponding instance. As a reminder on hyper parameters settings of Casper, we always initialize 5 candidate hidden neurons and add the one with highest correlation on residual errors. For the fair comparison, we adopt the same training and validation strategy as [CJ19], which optimizes the model performance on the validation dataset based on the feedback on the training dataset. Another reason for excluding the testing dataset is the dataset contains limited samples.

3.1 Casper

In current settings, to our best knowledge, we will obtain the best performance Casper model by adding modern neural network techniques and using mRMR feature selections. We add ReLu activation function, batch normalization, and dropouts to the Casper model. For every hidden layers, the dropout [Se12] with probability 0.1 and ReLu activation function are added, while the batch normalization [IS15] are only added to the first layer of Casper [TG97]. As table 1 shows, the Casper algorithm is able to archive 64.25% accuracy by using PD databases only.

	Macro Precision	**Macro Recall**	**Accuracy**
GSR database	0.6232	0.6933	0.6011
PD database	0.6833	0.6567	0.5925
ST database	0.5364	0.5635	0.5033
All feature	0.6035	0.6732	0.5301

Table 1. Performance Measure for Casper model

3.2 Evolutionary Algorithm Optimized Feed-forward Neural Networks

The domain of structures and hyper parameters for searching are listed below.

1. The number of hidden layers are up to 4.
2. Each hidden layers have 80 hidden neurons at most.
3. The activation function used for the hidden layers are ReLu, Sigmoid, LeaklyRelu or Softplus
4. The number of data in each mini batch are searched from 8 to 32.
5. The optimizer are selected from Adam and RMSprop.
6. The dropout for each hidden layers are 0, 0.1,0.2 or 0.5
7. The learning rate are 0.1, 0.05,0.01,0.005 or 0.001

Note, we will always use cross entropy as the lose functions, and softmax activation function for the output layers. It is hard to dynamically set the reasonable training epoch to different candidate solutions. As the fitness score of a candidate solution will be the highest accuracy among the training epochs on the validation dataset, the large training epoch usually brings more chance to get good performance. Certainly, the evolutionary algorithm will converge to the candidate solutions with large training epochs. Thus we use 50 epoch every candidate solutions. And if the best fitness scores did not increase for 5 epochs, the evolution terminates, which means the evolutionary algorithm stuck in the local minimal.

Method	Value
Maximum Generation	30
Population Size	15
Tournament Selection	6
Uniform Crossover	0.5
Uniform Mutation	0.05
Stop Condition	No Fitness increase for 5 recent Generation

Table 2. Hyper parameters of Evolutionary Algorithm

	First Hidden layer		**Second Hidden layer**		**Third Hidden layer**		**Fourth Hidden layer**		**Other Parameters**			
	neurons	activation	neurons	activation	neurons	activation	neurons	activation	minibatch	optimizer	dropout	learning rate
GSR database	80	SoftPlus	50	LeakyReLu	30	SoftPlus	30	SoftPlus	16	Adam	0.1	0.001
ST database	70	LeakyReLu	50	ReLu	60	ReLu	40	SoftPlus	16	Adam	0.1	0.01
PD database	60	LeakyReLu	20	LeakyReLu	10	SoftPlus	50	ReLu	24	RMSprop	0.2	0.01
All features	80	ReLu	20	SoftPlus	40	LeakyReLu	NA	NA	24	RMSprop	0.2	0.005

Table 3. The Structures of Evolutionary Algorithm Optimized Feed-forward Neural Networks

The hyper parameters of the evolutionary algorithm is summarized in table 2. Considering the size of the databases, that only 192 instances in total. We maintain 15 candidate models (Population Size) in each

generation, and the evolution takes up to 30 generations. The tournament selection with candidate pool size 6 are used since we do not want the bad candidate models (individuals) influence the next generations. While the uniform cross over with 0.5 probability are used to ensure the sufficient explorations and exploitation, that the search would not over concentrate on the local area. The uniform mutation with 0.05 probability is used, that we want the convergence happens within 30 generations. The optimized structures on each database are shown in table 3, where NA indicates that do not have the layers or activation functions. The variations of fitness score (accuracy) of each generation is shown in figure 2 and figure 3. As the generations increases, the average fitness and best fitness among the populations significantly increases. By using ST databases, the algorithm can achieve up to 53.5% accuracy. The detail performance are shown in next subsection.

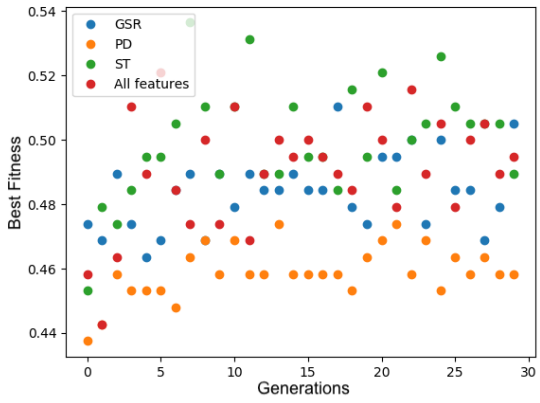

Fig. 2. Best Fitness Variations of Generations on Different Datasets

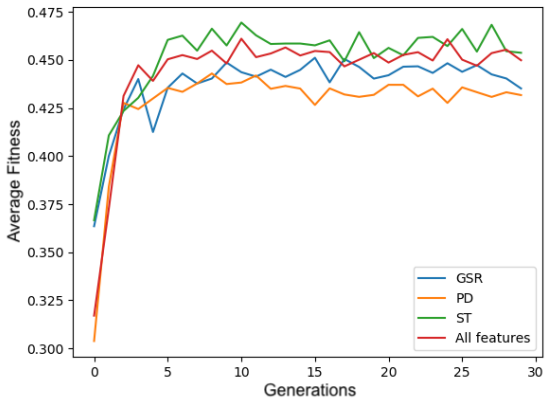

Fig. 3. Average Fitness Variations of Generations on Different Datasets

3.3 Discussion

Firstly, we want to talk about the training time on Casper. Withing our implementation of Casper [TG97], the model are not efficient as it reported in the original papers. And, though the convergence of Casper on hidden neuron selection is fast, it involves a large number of sequential calculations which are unable to be paralleled. In our experiments, the training time on Casper usually requires more than 20 minutes on the small dataset, while the Casper is implemented within the Pytorch library. During our investigation, we would like to conclude that the Casper is not efficient within the modern computers.

In our experiments, the batch normalization [IS15] layer and dropout layer [Se12] consistently increase the model performance of Casper. For the evolutionary algorithm optimized dense model, the trend of convergences can be saw in figure 2 and figure 3. Due to the computation and time limits, we do not have much experiments on the selection of maximum generations, number of populations, and the domain to search. The result are likely to be improved. Though, the mRMR feature selections increase the model performance in majority times in our experiments. And, the relative increase of accuracy are shown in table 4. Usually feature selection should be tuned before evolutionary algorithm, but each run requires nearly one day. Thus, for evolutionary algorithm optimized feed forward neural network, we first search for the best network architectures and hyper parameters, and then fine tuning the feature selections. The process of feature selection is computationally expensive, usually requires more time on that than the model training times. The more time-efficient feature selection method may need to be selected or proposed afterward. Table 3 below shows the accuracy value for different neural network models on different depression datasets. The performance reported in [CJ19] significantly beat our best performed models, and referred as Dense network in the table. Our re-implementation of the model described in [CJ19] is referred as Dense network 2. For self-comparison, except ST database and All features, the Casper model outperforms the evolutionary optimized feed-forward neural networks both in performance and training times.

	Casper	Evolutionary algorithm optimized network
GSR database	+0.0061	+0.0125
PD database	+0.0175	-0.0034
ST database	+0.0401	+0.0235
All features	+0.0108	+0.0155

Table 4. Performance increase for feature selections

	Casper	Dense network	Dense network 2	Evolutionary algorithm optimized network
GSR database	0.6011	0.89	0.4115	0.5225
PD database	0.5925	0.87	0.4010	0.4834
ST database	0.5033	0.92	0.3958	0.5585
All features	0.5301	0.92	0.4375	0.5355

Table 5. Accuracy for different Models

4 Conclusion and Future works

In this paper, we investigate the performance of two different neural network architecture optimization algorithms, Casper [TG97] and evolutionary algorithm optimized feed-forward neural networks on the depression datasets. The Casper algorithm outperforms the evolutionary algorithm optimized feed-forward neural networks in majority databases.By using the Casper, we can achieve 60.11% accuracy on predicating the depression level. Also, it proves the effectiveness of using three different physiological signals of observers in depression diagnose. However, we only explored feature fusion in a single models that all features are concatenated before passing to the model. Future work could focus on training three models on GSR databases, PD databases and ST databases respectively, and the final prediction are fused by different strategy, for example, hard voting, soft voting and etc. It very likely increase the performance of the model, which will significantly benefit society. Also, the Casper is no longer to gain the advantage on training time compared to dense connected neural networks, the Casper have lots of sequential operations which are unable to be processed at Graphic Unit Processor at once. Future work could also be focusing on the compatibility of Casper on the modern computation devices.

References

[BF94] S Baluja and SE Fahlman. “Reducing network depth in the cascade-correlation learning architecture”. In: *Pittsburgh* (1994).

[TG97] Nickt Treadgold and Tom Gedeon. “A Cascade Network Algorithm Employing Progressive RPROP”. In: *International Work-Conference on Artificial Neural Networks* (1997).

[LD05] H Peng; F Long and C Ding. “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-re- dundancy”. In: *IEEE Trans* (2005).

[Eng07] Andries P. Engelbrecht. *Computational Intelligence An Introduction*. EVOLUTIONARY COMPUTATION. British Library, 2007. ISBN: 9780470035610.

[Se12] Geoffrey E. Hinton; Nitish Srivastava and etc. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *Neural and Evolutionary Computing* (2012).

[IS15] Sergey Ioffe and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Machine Learning* (2015).

[Se15] N Cummins; S Scherer and etc. “A review of depression and suicide risk assessment using speech analysis”. In: *Speech Commun* (2015).

[Ce16] S Potvin; G Charbonneau and etc. “Self-evaluation and objective assessment of cognition in major depression and attention deficit disorder: Implications for clinical practice”. In: *Compr. Psychiatry* (2016).

[CJ19] Xuanying Zhu;Tom Gedeon;Sabrina Caldwell and Richard Jones. “Detecting emotional reactions to videos of depression”. In: *IEEE International Conference on Intelligent Intelligent Robots and Systems* (2019).