

# Evolutionary Neural Networks on Predicting Preferred Personality for Friend Recommendation in Social Networks - A Comparative Study

Nafis Neehal

*Technical Report Submitted to: Professor Dr. M. Sohel Rahman  
Department of CSE, Bangladesh University of Engineering and Technology  
Dhaka, Bangladesh.*

---

## Abstract

Neural Networks are one of the most powerful classification models till date. But, it suffers from several problems like - getting stuck in the local minima, or static structure selection, or slow convergence of synaptic weights, etc. Development of Hybrid Evolutionary Neural Networks using Evolutionary Algorithms and Swarm Algorithms is becoming more and more popular to avoid these kind of problems. In this paper, I have developed 6 different Evolutionary Neural Networks and conducted a comparative study on their performance on a particular learning problem. I've also proposed a novel architecture of a evolutionary neural network which shows potential to perform well in complex and large learning problems.

*Keywords:* Evolutionary Neural Networks, Neuroevolution, PSO, Evolution Strategies, Artificial Neural Networks

---

## 1. Introduction

The use of evolutionary algorithms (EAs) to aid in artificial neural network (ANNs) learning is nothing new. It has been a popular approach for a long time. But mostly, these algorithms have been used to cover for the shortcomings of

---

\*Corresponding author

Email address: [nafisneehal@iut-dhaka.edu](mailto:nafisneehal@iut-dhaka.edu) (Nafis Neehal)

5 backpropagation, which is - it often gets stuck in the local minima [1]. In this  
6 study that I've conducted, I've developed 6 different evolutionary neural net-  
7 works, each have different combinations of Particle Swarm Optimization (PSO),  
8 Evolution Strategies (ES) and Backpropagation (BP) to depict the structure of  
9 the network (choose number of hidden layer nodes and activation function used  
10 in hidden layer), initialize the weights of the network and optimize it.

### 11 1.1. motivation

12 While studying neural networks, I've always felt that it is somewhat being  
13 forced to underperform despite having the potential to do more. I believe,  
14 the fixed structure of neural network, that we determine beforehand, and the  
15 random initial weights that we use when instantiating a neural network object  
16 are the main two reasons for neural nets underperforming. And, to solve this  
17 two problems altogether using a single neural network was my main motivation.

### 18 1.2. Research Contribution

19 My research contributions are as follows -

- 20 • Developing 6 different hybrid evolutionary neural networks, among which  
21 4 are complete new as far as I know. Because, ES has not been used  
22 (maybe in one or two, although I couldn't find any) in neural networks,  
23 and also I didn't find any neural networks which solely had PSO for weight  
24 optimization.
- 25 • Comparing the performances of these 6 networks on a specific learning  
26 problem, on which much work not have been done also.
- 27 • Pointing out the limitations of each of the network, by developing which  
28 the network's performance can be increased to an excellent level.
- 29 • Proposed a novel neural network (ESPSB\_Net) which can select its struc-  
30 ture by itself using ES, use PSO to curate initial weights and then optimize  
31 those weights using typical backpropagation. I believe, this neural network

enforces a balanced application of exploration (PSO, ES) and exploitation (Backpropagation) which shows a very good potential of becoming a powerful self-evolving neural network.

### 1.3. Literature Review

As I stated earlier, uses of Evolutionary Algorithms and Swarm Algorithms to optimize its structure and weights is nothing new (but implementing both in the same network has not been done yet which is my major contribution). Palmes et. al. [1] proposed a mutation based genetic neural network to address the problem of backpropagation getting stuck in local optima. Lam et. al. [2] presented a neural network where they tuned the structure and parameters of neural network using an improved version of GA. Settles et. al. [3] presented a Hybrid GA/PSO Neural Network where they used both GA and PSO for weight optimization, but not for defining structure.

Mohammad et. al. [4] proposed a neural network, similar to PB\_Net that I've developed. They had initialized a neural network with weights optimized by PSO for a few rounds, and then used those weights to initialize the final network and optimize those weights later on using backpropagation. Juang et.al. [5] proposed an evolutionary recurrent neural network, where he used and hybrid of PSO and GA for structure optimization only. Garro et. al. [6] showed a comparison of performance between backpropagation and PSO in optimizing synaptic weights of an ANN and showed that PSO performed better most of the time than backpropagation. Meissner et. al. [7] used an Optimized version of PSO for training neural networks and achieved a time decrease by a factor of four and two while comparing with other methods. Gudise et. al. [8] also compared PSO and Backpropagation for training neural networks and showed that PSO converges faster than Backpropagation.

## 59 2. Methodology

60 Evolutionary Neural Networks have become a cutting edge research tool.  
61 Many well-known research groups have started developing optimized evolution-  
62 ary neural networks and incorporating them in their research works where the  
63 hyper-parameter space is huge. Particle Swarm Optimization is a Stochastic Op-  
64 timization Method which recently has become very popular for fusing with Ar-  
65 tificial Neural Networks. Also, among typical Evolutionary Algorithms (where  
66 re-sampling occurs), Genetic Algorithm is the most popular one to use with  
67 Neural Networks.

### 69 2.1. Problem Definition

70 In this work, I've used a fusion of ( $\mu$ ,  $\lambda$ ) Evolution Strategies, Particle  
71 Swarm Optimization and Backpropagation in different combinations for build-  
72 ing 6 different evolutionary neural networks and assess their performance on a  
73 particular learning problem.

### 75 2.2. The Learning Problem Description

76 The learning problem I've used here is - "Prediction of Preferred Personal-  
77 ity for Friend Recommendation in Social Networks". Normally, social networks,  
78 like Facebook, recommends friends to a user based on number of mutual friends,  
79 match in Geo-location (same office, same educational institution etc.), common  
80 interests etc. But, in real life, we typically become friends with the person who  
81 has similar personality like us or has the personality that we like. So, being  
82 motivated from that, in my B.Sc thesis I started to work on developing a Friend  
83 Recommendation Framework that will recommend friends to users based on  
84 personality preference. In my earlier thesis work, I applied a plain vanilla form  
85 of Artificial Neural Network. And in this work, I applied the different evolu-  
86 tionary neural network models on this particular learning problem and assessed

87 their performance.

88

89 The Big Five personality traits, also known as the five-factor model (FFM),  
90 and the OCEAN model, gives 5 different categories for human personalities,  
91 namely -

- 92 • Openness to experience (inventive/curious vs. consistent/cautious)
- 93 • Conscientiousness (efficient/organized vs. easy-going/careless)
- 94 • Extraversion (outgoing/energetic vs. solitary/reserved)
- 95 • Agreeableness (friendly/compassionate vs. challenging/detached)
- 96 • Neuroticism (sensitive/nervous vs. secure/confident)

97 For personality assessment, a 50 Question Survey form was distributed among  
98 surveyees. From the answer to the questions of this survey, a surveyee's person-  
99 ality score in each of the 5 personality categories can be derived. These 5 score  
100 values in 5 different personality categories are used as 5 different feature values  
101 for our learning problem. This looks like a float vector like [2.3, 4.4, 3.2, 4.1,  
102 2.4].

103 In the survey, it was also questioned that what type of personality among  
104 the five would the user choose to be friends with. Users could choose one or  
105 more than one personalities of their preference for friendship. This is a size 5  
106 boolean vector like [0, 1, 0, 0, 1].

107 The survey was done by, David Stillwell of The Psychometrics Centre, Uni-  
108 versity of Cambridge. [9]. But, unfortunately, this dataset is now closed after  
109 the Cambridge Analytical event [10]. However, I started working on the dataset  
110 way before they closed it.

### 111 2.3. Dataset Description

112 The dataset contains 9917 samples and 11 columns. The first column depicts  
113 the sample number (a facebook user), column [2-6] depicts the float vector of

114 feature values (five personality scores), and column [7-11] depicts the boolean  
115 vector for friendship preference.

#### 116 2.4. Github Repository for Code and Dataset

117 This is the Github Repository link -

118

119 <https://github.com/Nafis-Neehal-IUT/Neuroevolution-BUET>

120 It contains the following files -

- 121 • Iris Neural Network.ipynb - A draft version of (5,20) ES on Neural Network  
122 [not part of the main project, just saving it here for any future reference]
- 123 • Main.ipynb - The main notebook, contains all the 6 Evolutionary Neural  
124 Network implementations that I build from scratch
- 125 • NeuralNetwork.py - The NeuralNetwork module I wrote from scratch.
- 126 • PSO.py - The Particle Swarm Optimizaiton module I wrote from scratch.
- 127 • Readme.md - Contains installation guidelines
- 128 • input.csv - The Personality Dataset (9917 x 11)

129 *[If you face difficulty to Render the Main.ipynb notebook in Github, please*  
130 *refer to this link in NBViewer - <http://bit.ly/nbviewerneuro>]*

#### 131 2.5. Overview of the Approach

132 A brief outline of my approach -

- 133 • At first, I imported the data and all other necessary modules and packages  
134 into my program
- 135 • Then I did the necessary data preprocessing. I converted all the data  
136 which were "object" type into "float" type. Also, omitted the first col-  
137 umn containing the sample number, and splitted the remaining dataset  
138 (dimension 9917 x 10) into train, validation and test set. The training set

139 contained 8000 samples, validation set contained 1000 samples and test  
140 set contained 917 samples. All these samples were randomly taken from  
141 the main dataset. Also wrote a function for threshold mapping. This is  
142 because, Sigmoid function always compresses the output between  $[0,1]$ .  
143 So, if the final layer output of Neural Network is greater than or equal to  
144 0.5 in any of the output node, then I converted it to 1, and if it was less  
145 than 0.5 then converted it to 0.

146 • Then, I designed the outline for 6 Neural Network Models that I was going  
147 to test -

- 148 (1) **N\_Net:** Plain Vanilla Neural Network with randomly initialized  
149 weights which will be optimized using Backpropagation only
- 150 (2) **NP\_Net:** Neural Network with randomly initialized weights which  
151 will be optimized using PSO only, No Backpropagation
- 152 (3) **PB\_Net:** Neural Network with initial weights optimized by PSO,  
153 and then further optimized using Backpropagation
- 154 (4) **ES\_Net:** Neural Network with structure selected by (5,20) Evolution  
155 Strategies and randomly initialized weights which will be optimized  
156 using Backpropagation
- 157 (5) **ESPS\_Net:** Neural Network with structure selected by (5,20) Evo-  
158 lution Strategies and randomly initialized weights which will be op-  
159 timed using PSO only, No Backpropagation
- 160 (6) **ESPSB\_Net:** Neural Network with with structure selected by (5,20)  
161 Evolution Strategies and initial weights optimized by PSO (not ran-  
162 domly initialized) and finally updated using Backpropagation

163 I applied each of those nets on the personality dataset and plotted training  
164 and validation loss and also calculated precision, recall, f-measure and  
165 accuracy acquired by using those nets on test dataset.

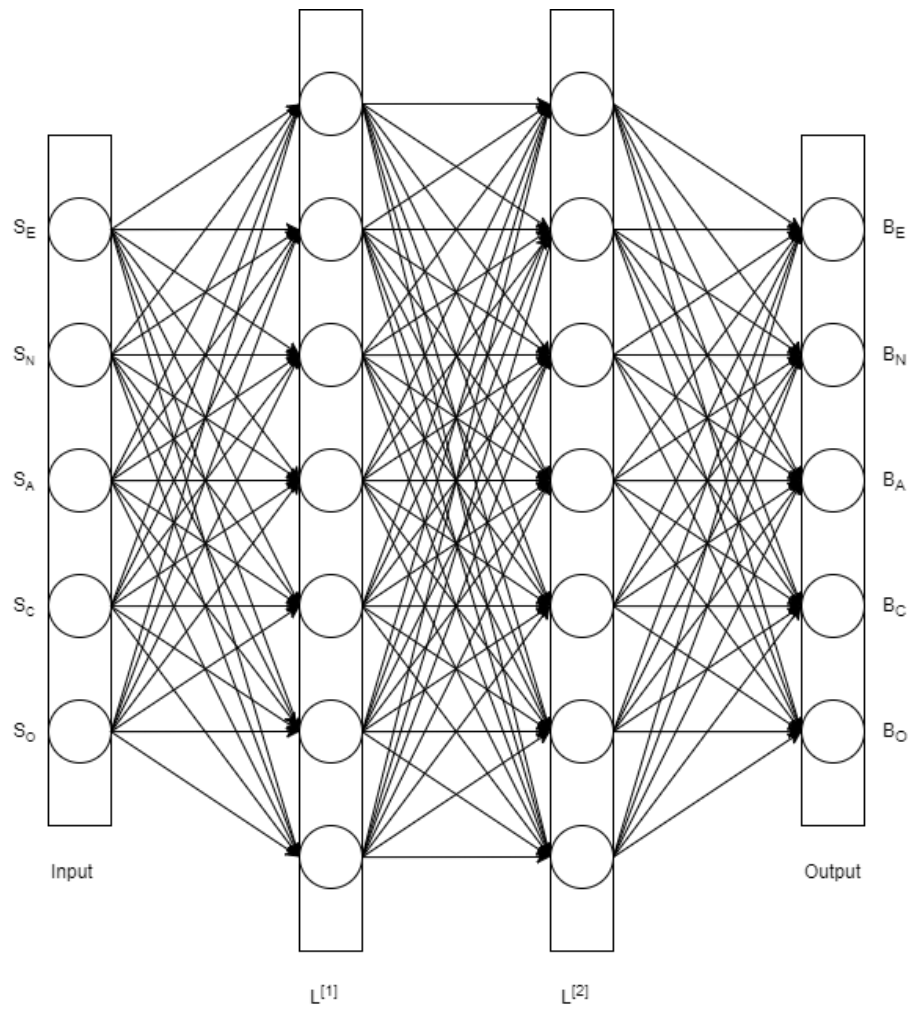


Figure 1: Basic Structure of My Neural Network



### 166 3. Results

167 Detailed Result of my experiments are described in the following sections.

#### 168 3.1. Experimental Setup

169 My experimental setup details are given below (Basic Neural Network's  
170 Structure in Fig. 1) -

##### 171 3.1.1. PC Configuration

- 172 • RAM: 8GB
- 173 • OS: 64 Bit (Windows 8.1)
- 174 • Processor: Core i5 (5th Gen)
- 175 • Processor Speed: 2.20 GHz (No Overclocking)

##### 176 3.1.2. Neural Network, PSO and ES Configurations

- 177 • **N\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean vector),  
178 2 hidden layers, 8 hidden nodes each, Output activation: Sigmoid, Hidden  
179 Activation: ReLU. PSO and ES not used. Neural Net Epoch: 2000.
- 180 • **NP\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean vector),  
181 2 hidden layers, 16 hidden nodes each, Output activation: Sigmoid, Hidden  
182 Activation: ReLU. PSO used for 1000 Generations, 50 particles each.
- 183 • **PB\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean vector),  
184 2 hidden layers, 8 hidden nodes each, Output activation: Sigmoid, Hid-  
185 den Activation: ReLU. PSO used for 100 Generations 30 Particles, each.  
186 Neural Net Epoch: 2000.
- 187 • **ES\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean vector),  
188 2 hidden layers, 17 hidden nodes each (determined by ES), Output acti-  
189 vation: Sigmoid, Hidden Activation: Tanh (determined by ES). (5,20) ES  
190 used for 30 generations, PSO not used. Neural Net Epoch: 2000.

- 191 • **ESPS\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean vec-  
192 tor), 2 hidden layers, 17 hidden nodes each (determined by ES), Output  
193 activation: Sigmoid, Hidden Activation: Tanh (determined by ES). (5,20)  
194 ES used for 30 generations. PSO used for 1000 Generations, 50 particles  
195 each.
- 196 • **ESPSB\_Net:** 5 Input Nodes (float vector), 5 Output Nodes (boolean  
197 vector), 2 hidden layers, 17 hidden nodes each (determined by ES), Output  
198 activation: Sigmoid, Hidden Activation: Tanh (determined by ES). (5,20)  
199 ES used for 30 generations. PSO used for 100 Generations, 20 particles  
200 each. Neural Net Epoch: 2000.

### 201 3.2. *Experimental Results*

202 Experimental Results of my models are given below -

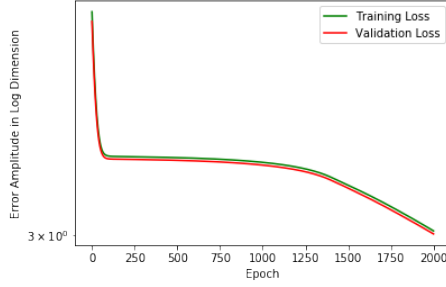
#### 203 3.2.1. *Training-Validation Loss VS Epoch*

204 Figure 1 contains Training-Validation Loss for all the 6 Neural Nets. Among  
205 them NP\_Net and ESPS\_Net - in which only PSO was used to optimize the  
206 neural network's weights are trained on 1000 Epochs. The rest four neural  
207 networks were trained on 2000 epochs each.

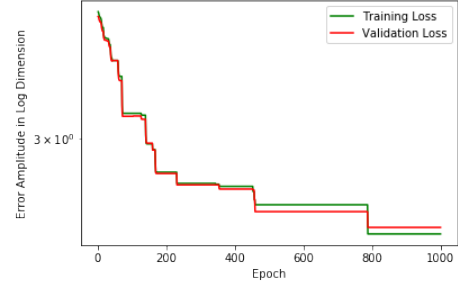
#### 208 3.2.2. *Performance on Test Dataset*

209 Figure 2 contains all the performance measurements data of all 6 nets on the  
210 test dataset. All of these values are the average values for all 917 test samples.

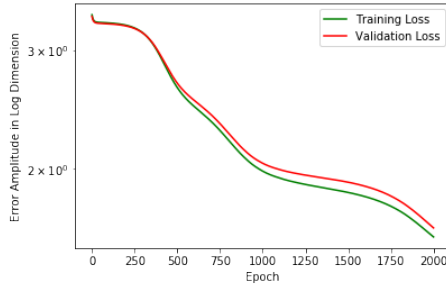
211 For testing, at first output for all the 917 values were predicted using a single  
212 forward pass of the neural network. And then, those predictions were converted  
213 to 0 or 1 based on - if any of the 5 output nodes emitted a value less than 0.5, it  
214 was converted to 0 and if it emitted a value greater than or equal to 0.5 then it  
215 was converted to 1. It is noteworthy, Sigmoid function was used at the output  
216 layer which always squeezes outputs to a float number between [0,1]



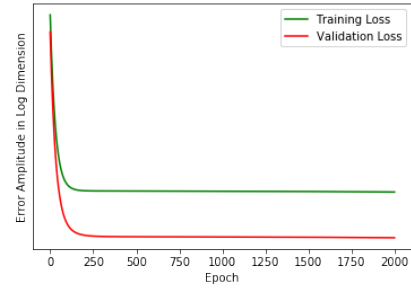
(a) N\_Net



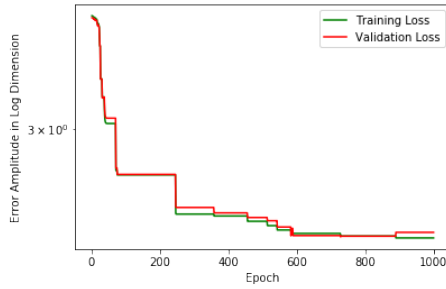
(b) NP\_Net



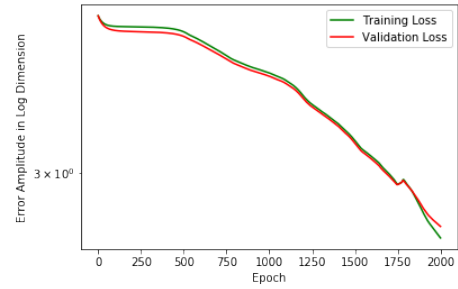
(c) PB\_Net



(d) ES\_Net



(e) ESPS\_Net



(f) ESPSB\_Net

Figure 2: Training-Validation Loss VS Epoch Plot for all 6 Nets

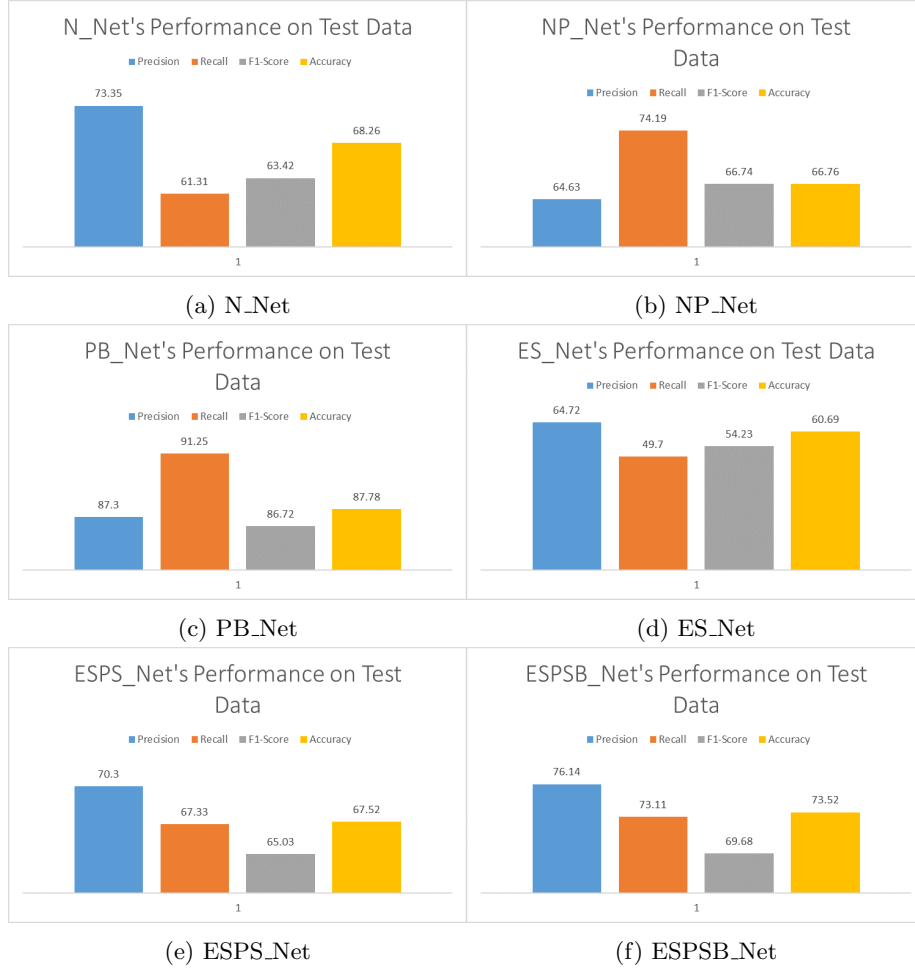


Figure 3: Performance of All 6 Nets on Test Data

### 217 3.3. Discussions

218 *N\_Net.* N\_Net was the typical plain vanilla form of Neural Network. I trained  
219 it for 2000 epochs on training set (8000 samples) without using any kind of  
220 added parameters like momentum, weight decreasing, dropout, regularization  
221 etc. for avoiding overfitting, or more optimization. From the training-validation  
222 loss result, it can be seen, that after around 120 epochs, the loss decrease graph  
223 got flattened for around 1100 epochs, and then slowly started going downwards  
224 again. It maybe the case that it was stuck in a local optima for all those epochs  
225 and couldn't find the right weights because I set the learning rate to 0.01. It  
226 can be tuned a little bit avoid this flattening. Although, this network only  
227 shows a very small sign of overfitting (huge dataset helped in this regard), but  
228 as it flattened after 120 epochs for quite a long time, I think I could have  
229 adopted the early stopping method here to avoid this overfitting because the  
230 later decrease in weights were somewhat negligible. Its performance on the test  
231 dataset was also not that much satisfactory (depicts the overfitting) as I didn't  
232 use any extra optimization techniques. Adopting those will certainly increase  
233 the overall performance on test data and reduce overfitting.

234 *NP\_Net.* In NP\_Net, I didn't use any kind of backpropagation at all and solely  
235 depended on PSO for weight optimization. From the loss figure, it is prominent,  
236 that there are frequent flattening for certain period of epochs in between. As,  
237 PSO is a global optimization algorithm and enforces exploration, I guess, hence  
238 the periodic irregular flattened areas. It can also be seen, that upto 200 gener-  
239 ations, the weight decrease was significant, but after that, the weight decrease  
240 got somewhat slowed down. So, here also, I could have adopted early stopping  
241 after around the first 200 epochs. It also shows some sign of overfitting (de-  
242 picted from test performance), which could be avoided using regularization or  
243 dropout.

244 *PB\_Net.* PB\_net, depicted the best performance on test data in terms of all the  
245 performance metrics. In PB\_Net, I initialized the neural network with optimized  
246 by PSO for a specific number of rounds, and then optimized those weights using

typical backpropagation. Most probably, due to the initial use of curated weights this network performed the best. Also, from its training-validation loss graph, it shows a general trend of going downwards without any significant flattening. However, the rate of decrease for validation data was slight lower than training data. But, as validation graph is generally trending downwards, so it is safe to say that the model is not overfitted (this is also because the model is performing equally well on test data).

*ES\_Net.* The ES\_Net's performance was moderate. From training-validation loss graph, it can be seen that weight decrease stopped strictly after around 130 epochs. This early convergence can be due to the earlier structure selection of neural network by ES. As the model does not perform very well on the test data, then either this model is overfitted or it is stuck in local optima. The second one has better chance of happening, because for the model to be overfitted, the training accuracy needs to be very good and continuously decreasing or at least showing a downward trend, but here it flattened. So, it is very likely, that the model got stuck in a local optima which is very typical for gradient based optimizations.

**Chromosome and Gene Details:** In (5,20) ES, each chromosome will be a vector like this  $[4, [1, 0, 0]]$ . Here, the first element will be an integer between 4-32 inclusive. This will depict the number of nodes in each of the hidden layer. And the second element is a boolean vector which can be mapped -  $[1, 0, 0] = \text{ReLU}$ ,  $[0, 1, 0] = \text{tanh}$  and  $[0, 0, 1] = \text{Sigmoid}$ .

**Selection for Mutation:** Coin toss will be done for selecting whether any gene of the chromosome will be mutated or not. For gene A (number of nodes in hidden layer), selection threshold is set to 30% and for Gene B (boolean vector for activation function selection in hidden layer) is set to 50

**Mutation Operators:** Mutation operator for A is selecting a random number within  $[4, 32]$ . Used this instead of random walk to enforce exploration a bit to avoid possible local minima. Mutation operator for Gene B is simple bitwise rightshift. For instance,  $[1,0,0] \rightarrow [0,1,0]$ ;  $[0,1,0] \rightarrow [0,0,1]$  and  $[0,0,1] \rightarrow [1,0,0]$

277     **Taboo List:** I’ve also maintained a Taboo list of  $\lambda$  (population size = 20)  
 278 size. Whenever a new child is created, it is first checked whether it is in the  
 279 taboo list. If yes, then another new child is created. If not, then the first element  
 280 is popped from the taboo list and this new child is appended at the last of the  
 281 taboo list besides being added as the new member of the main population. So,  
 282 each new candidate will get a fair lifetime of 19 new not-present-in-taboo birth.

283     **Fitness Assessment:** is done using a simple forward pass without any back-  
 284 ward pass. For each candidate structure, a neural network with random weight  
 285 is created. Then a single forward pass is made and then cost is calculated. This  
 286 is the measurement of fitness. Lower the cost means fitter the candidate.

287 *ESPS\_Net.* ESPS\_Net performed slightly better than its predecessor ES\_Net.  
 288 Although, it had the same structure, the training and validation loss decrease  
 289 trend of ESPS\_Net is generally downwards which shows that it is probably not  
 290 stuck any local optima (also, as PSO was used for weight optimization, not  
 291 backprop, so very few chance of getting stuck in local optima). However, it  
 292 still didn’t perform well on the test set which depicts the sign of being slightly  
 293 overfitted. This can be omitted by adopting early stopping after around 250  
 294 epochs (from train-validation loss graph) and use regularization or dropout.

295 *ESPSB\_Net.* ESPSB\_Net was my target neural network. It performed the sec-  
 296 ond best on test dataset in terms of performance metrics. It also has a very  
 297 good training-validation loss decreasing trend without any flattening or discon-  
 298 tinuity. Performance on test data still has room for improvement as it still shows  
 299 very small sign of overfitting which can be improved by applying regularization  
 300 or dropout (early stopping will not work well on this model - depicted from  
 301 training-validation loss graph).

## 302 4. Limitations of the Work

- 303     • My Laptop is not that much high-end, so couldn’t perform all the simu-  
 304 lations.

- 305 • Didn't take any measurement for overfitting reduction like - Regulariza-  
306 tion, Dropout, Early Stopping etc.
- 307 • Didn't incorporate runtime measurement in performance assessment. Need  
308 to work on that aspect as well.
- 309 • Also, I need to refactor my code. I've directly implemented ES which I  
310 shouldn't have done. Instead, I should have implemented the ES code as  
311 a class, like I did with NeuralNetwork.py and PSO.py.

## 312 5. Conclusion

313 This study depicts the potential of my desired network, which was the last  
314 one ESPSB.Net. Although it performed second best, but I believe it's perfor-  
315 mance can be enhanced by taking measures to prevent overfitting, as well as  
316 tune the hyper-parameters a little bit. Further performance assessment of this  
317 network on benchmark optimization problems like Rastrigin Function, Matyas  
318 Function, Ackley Function, Peaks Function etc can be added. Also, instead of  
319 PSO, other swarm algorithms like Firefly algorithm can be used here, as well as  
320 other EAs like Genetic Algorithm can be experimented with too. Overall, build-  
321 ing these hybrid networks and applying them on different learning problems can  
322 be a new research domain.

## 323 References

- 324 [1] P. Palmes, T. Hayasaka, S. Usui, Mutation-Based Genetic Neural Net-  
325 work, IEEE Transactions on Neural Networks 16 (3) (2005) 587–600.  
326 doi:10.1109/TNN.2005.844858.  
327 URL <http://www.ncbi.nlm.nih.gov/pubmed/15940989><http://ieeexplore.ieee.org/document/1427764/>  
328
- 329 [2] F. Leung, H. Lam, S. Ling, P. Tam, Tuning of the structure and parameters  
330 of a neural network using an improved genetic algorithm, IEEE Transac-  
331 tions on Neural Networks 14 (1) (2003) 79–88. doi:10.1109/TNN.2002.



- 332 804317.  
333 URL <http://ieeexplore.ieee.org/document/1176129/>
- 334 [3] M. Settles, T. Soule, Breeding swarms, in: Proceedings of the 2005 confer-  
335 ence on Genetic and evolutionary computation - GECCO '05, ACM Press,  
336 New York, New York, USA, 2005, p. 161. doi:10.1145/1068009.1068035.  
337 URL <http://portal.acm.org/citation.cfm?doid=1068009.1068035>
- 338 [4] E. T. Mohamad, D. J. Armaghani, E. Momeni, A. H. Yazdavar,  
339 M. Ebrahimi, Rock strength estimation: a PSO-based BP approach, Neu-  
340 ral Computing and Applications 30 (5) (2018) 1635–1646. doi:10.1007/  
341 s00521-016-2728-3.  
342 URL <http://link.springer.com/10.1007/s00521-016-2728-3>
- 343 [5] C.-F. Juang, A Hybrid of Genetic Algorithm and Particle Swarm Opti-  
344 mization for Recurrent Network Design, IEEE Transactions on Systems,  
345 Man and Cybernetics, Part B (Cybernetics) 34 (2) (2004) 997–1006.  
346 doi:10.1109/TSMCB.2003.818557.  
347 URL <http://ieeexplore.ieee.org/document/1275532/>
- 348 [6] B. A. Impr. Bourg-offset), H. Sossa, R. A. Vázquez, Communication, BEP  
349 terminale professionnelle : corrige., Vol. 7, Fontaine Picard, 1997.  
350 URL [http://www.ceser.in/ceserp/index.php/ijai/article/view/](http://www.ceser.in/ceserp/index.php/ijai/article/view/2260)  
351 2260
- 352 [7] M. Meissner, M. Schmuker, G. Schneider, Optimized Particle Swarm Opti-  
353 mization (OPSO) and its application to artificial neural network training,  
354 BMC Bioinformatics 7 (1) (2006) 125. doi:10.1186/1471-2105-7-125.  
355 URL [http://bmcbioinformatics.biomedcentral.com/articles/10.](http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-125)  
356 1186/1471-2105-7-125
- 357 [8] V. Gudise, G. Venayagamoorthy, Comparison of particle swarm optimiza-  
358 tion and backpropagation as training algorithms for neural networks, in:  
359 Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat.

- 360 No.03EX706), IEEE, pp. 110–117. doi:10.1109/SIS.2003.1202255.  
361 URL <http://ieeexplore.ieee.org/document/1202255/>
- 362 [9] David Stillwell, MyPersonality Database from The Psychometrics Centre,  
363 University of Cambridge.  
364 URL [https://www.psychometrics.cam.ac.uk/productsservices/](https://www.psychometrics.cam.ac.uk/productsservices/mypersonality)  
365 [mypersonality](https://www.psychometrics.cam.ac.uk/productsservices/mypersonality)
- 366 [10] M. Kosinski, My Personality Project.  
367 URL <https://sites.google.com/michalkosinski.com/mypersonality>

## 368 6. Appendix

### 369 6.1. Neural Network Equations

#### 370 6.1.1. Forward Propagation

371 Forward pass is basically the prediction of neural networks. The notations  
372 that have been used in forward propagation is described below:

- 373 •  $Z^{[i]}$  = Output of  $i^{\text{th}}$  Layer of Neural Network
- 374 •  $A^{[i]}$  = Activation of  $i^{\text{th}}$  Layer of Neural Network
- 375 •  $X$  = input
- 376 •  $W^{[i]}$  = Weight from  $(i-1)^{\text{th}}$  to  $i^{\text{th}}$  Layer of Neural Network
- 377 •  $b^{[i]}$  = Biases for  $i^{\text{th}}$  Layer of Neural Network

378 Equations for Forward Pass is shown in Algorithm 1.

---

**Algorithm 1** Forward Propagation in Neural Network

---

```
1: procedure FORWARDPROPAGATION( $X$ )           ▷ Forward Propagation
2:    $Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$ 
3:    $A^{[1]} = \text{ReLU}(Z^{[1]})$ 
4:    $Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$ 
5:    $A^{[2]} = \text{ReLU}(Z^{[2]})$ 
6:    $Z^{[3]} = W^{[3]} \cdot A^{[2]} + b^{[3]}$ 
7:    $A^{[3]} = \text{Sigmoid}(Z^{[3]})$ 
8: end procedure
```

---

#### 379 6.1.2. Cross-Entropy Loss Function

380 Cross-Entropy is the most popular loss function for neural networks. It is  
shown in Algorithm 2.

---

**Algorithm 2** Loss Function in Neural Network

---

```
1: procedure LOSSFUNCTION( $Y$ )
2:   RETURN  $\frac{1}{m} \sum_{i=1}^m (-Y^{(i)} \log(Y^{(i)}) - (1 - Y^{(i)}) \log(1 - Y^{(i)}))$ 
3: end procedure
```

---

381

---

**Algorithm 3** Back Propagation in Neural Network

---

```
1: procedure BACKPROPAGATION( $X, Y$ )
2:    $dZ^{[3]} = A^{[3]} - Y$  ▷ Derivatives Calculation
3:    $dW^{[3]} = (1/m) dZ^{[3]} \cdot A^{[2].T}$ 
4:    $dB^{[3]} = (1/m) \text{np.sum}(dZ^{[3]}, \text{axis}=1, \text{Keepdims} = \text{True})$ 
5:    $dZ^{[2]} = W^{[3].T} \cdot dZ^{[3]} * \text{DReLU}(Z^{[2]})$ 
6:    $dW^{[2]} = (1/m) dZ^{[2]} \cdot A^{[1].T}$ 
7:    $dB^{[2]} = (1/m) \text{np.sum}(dZ^{[2]}, \text{axis}=1, \text{Keepdims} = \text{True})$ 
8:    $dZ^{[1]} = W^{[2].T} \cdot dZ^{[2]} * \text{DReLU}(Z^{[1]})$ 
9:    $dW^{[1]} = (1/m) dZ^{[1]} \cdot X^T$ 
10:   $dB^{[1]} = (1/m) \text{np.sum}(dZ^{[1]}, \text{axis}=1, \text{Keepdims} = \text{True})$ 
11:
12:   $W^{[1]} = W^{[1]} - 0.01 * dW^{[1]}$  ▷ Parameter Update
13:   $b^{[1]} = b^{[1]} - 0.001 * dB^{[1]}$ 
14:   $W^{[2]} = W^{[1]} - 0.01 * dW^{[2]}$ 
15:   $b^{[2]} = b^{[1]} - 0.001 * dB^{[2]}$ 
16:   $W^{[3]} = W^{[1]} - 0.01 * dW^{[3]}$ 
17:   $b^{[3]} = b^{[1]} - 0.001 * dB^{[3]}$ 
18: end procedure
```

---

### 6.1.3. Backward Propagation

Backpropagation is basically a gradient based learning and optimization method which is prone to local optima. The notations that have been used in back propagation is described below:

- $dZ^{[i]} =$  Derivative of  $Z$  of  $i^{\text{th}}$  Layer of Neural Network
- $A^{[i].T} =$  Transpose of  $A^{[i]}$
- $dW^{[i]} =$  Derivative of  $W$  of  $i^{\text{th}}$  Layer of Neural Network
- $dB^{[i]} =$  Derivative of  $B$  of  $i^{\text{th}}$  Layer of Neural Network

Equations for Backprop is shown in Algorithm 3.

## 391 6.2. PSO Pseudocode

392 In computational science, particle swarm optimization (PSO) is a computa-  
 393 tional method that optimizes a problem by iteratively trying to improve a can-  
 394 didate solution with regard to a given measure of quality. It solves a problem  
 395 by having a population of candidate solutions, here dubbed particles, and mov-  
 396 ing these particles around in the search-space according to simple mathematical  
 397 formulae over the particle's position and velocity. Each particle's movement is  
 398 influenced by its local best known position, but is also guided toward the best  
 399 known positions in the search-space, which are updated as better positions are  
 400 found by other particles. This is expected to move the swarm toward the best  
 401 solutions. Fig 4 and Fig 5 presents my implementation of PSO.

402 Main Equation of Particle Swarm Optimization -

- 403 •  $WV_{ij}(t) = \text{Inertia Component}$
- 404 •  $r_1 c_1 (P_{ij}(t) - x_{ij}(t)) = \text{Cognitive Component}$
- 405 •  $r_2 c_2 (g_j(t) - x_{ij}(t)) = \text{Social Component}$
- 406 •  $c_1, c_2 = \text{Acceleration Coefficients}$
- 407 •  $W = \text{Inertia Coefficient}$
- 408 •  $t, t(t+1) = \text{timestamps}$
- 409 •  $V_{ij}(t) = j^{\text{th}}$  velocity component of  $i^{\text{th}}$  particle
- 410 •  $P_{ij}(t) = j^{\text{th}}$  component of  $i^{\text{th}}$  particle's best position
- 411 •  $x_{ij}(t) = j^{\text{th}}$  component of  $i^{\text{th}}$  particle's current position
- 412 •  $r_1, r_2 = N(\mu, \sigma^2)$

413 For determining the value of  $W, c_1$  and  $c_2$ , I've used Constriction coefficients  
 414 proposed by Clerc and Kennedy, 2002.

415

Let,

$$\phi = \phi_1 + \phi_2 \geq 4$$

$$0 \leq \kappa \leq 1$$

$$\chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}$$

416

417

418 Commonly,  $\kappa = 1$ ,  $\phi_1 = 2.05$ ,  $\phi_2 = 2.05$

419

420 So, Finally,

421  $W = \chi$ ,  $C_1 = \kappa\phi_1$ ,  $C_2 = \kappa\phi_2$

422

And for Velocity and Position Update,

$$V_{ij}(t+1) = WV_{ij}(t) + r_1c_1(P_{ij}(t) - x_{ij}(t)) + r_2c_2(g_j(t) - x_{ij}(t))$$

$$x_{ij}(t+1) = x_{ij}(t) + V_{ij}(t+1)$$

---

**Algorithm 4** Particle Swarm Optimization in Neural Network

---

```

1: procedure PSOLOOP(NeuralNetwork)
2:   for each epoch(i) in range(1, maxIt) do
3:     for each particle(j) in range(1, nPop) do
4:        $V_j(t+1) \leftarrow WV_j(t) + r_1c_1(P_j(t) - x_j(t)) + r_2c_2(g(t) - x_j(t))$ 
5:        $x_j(t+1) \leftarrow x_j(t) + V_j(t+1)$ 
6:       particle(j).cost  $\leftarrow$  COSTPSO(NeuralNetwork,particlej)
7:       if particle(j).cost < particle(j).bestCost then
8:         particle(j).bestPosition  $\leftarrow$  particle(j).position
9:         particle(j).bestCost  $\leftarrow$  particle(j).cost
10:      if particle(j).bestCost < globalBestCost then
11:        globalBestPosition  $\leftarrow$  particle(j).bestPosition
12:        globalBestCost  $\leftarrow$  particle(j).cost.bestCost
13:      end if
14:    end if
15:  end for
16: end for
17: end procedure

```

---

---

**Algorithm 5** Particle Swarm Optimization in Neural Network

---

```
1: procedure INITIALIZEPARTICLE(nVar, nPop)
2:   globalBestPosition  $\leftarrow \vec{0}$  of dimension (nVar x 1)
3:   globalBestCost  $\leftarrow$  None
4:   for each particle(i) in range(1, nPop) do
5:     particle(i).position  $\leftarrow N(\mu, \sigma^2)$  of dimension (nVar x 1)
6:     particle(i).velocity  $\leftarrow \vec{0}$  of dimension (nVar x 1)
7:     particle(i).cost  $\leftarrow$  None
8:     particle(i).bestPosition  $\leftarrow$  particle(i).position
9:     particle(i).bestCost  $\leftarrow$  particle(i).cost
10:  end for
11: end procedure
12: procedure INITIALIZEPARAMETERS(NeuralNetwork, X, Y, maxIt, nPop)
13:   nVar  $\leftarrow$  NeuralNetwork.numWeights + NeuralNetwork.numBiases
14:    $\kappa \leftarrow 1$ 
15:    $\phi_1 \leftarrow 2.05$ 
16:    $\phi_2 \leftarrow 2.05$ 
17:    $\phi \leftarrow \phi_1 + \phi_2$ 
18:    $\chi \leftarrow \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}$ 
19:   maxIt  $\leftarrow 100$ 
20:   nPop  $\leftarrow 50$ 
21:    $w \leftarrow \chi$ 
22:   wdamp  $\leftarrow 0.99$ 
23:    $c_1 \leftarrow \chi \cdot \phi_1$ 
24:    $c_2 \leftarrow \chi \cdot \phi_2$ 
25: end procedure
26: procedure COSTPSO(NeuralNetwork, particle)
27:   NeuralNetwork.weights  $\leftarrow$  particle.position.slice
28:   NeuralNetwork.biases  $\leftarrow$  particle.position.slice
29:   NeuralNetwork.ForwardPropagation(X)
30:   RETURN NeuralNetwork.Loss(Y)
31: end procedure
32: procedure INITIALIZESWARM(NeuralNetwork)
33:   for each particle(i) in range(1, nPop) do
34:     particle(i).cost  $\leftarrow$  COSTPSO(NeuralNetwork, particle(i))
35:     if particle(i).bestCost  $<$  globalBestCost then
36:       globalBestPosition  $\leftarrow$  particle(i).bestPosition
37:       globalBestCost  $\leftarrow$  particle(i).bestCost
38:     end if
39:   end for
40: end procedure
```

---

6.3. Big 5 Personality Questionnaire

The most frequently used measures of the Big Five comprise either items that are self-descriptive sentences or, in the case of lexical measures, items that are single adjectives. Research has suggested that some methodologies in administering personality tests are inadequate in length and provide insufficient detail to truly evaluate personality. Usually, longer, more detailed questions will give a more accurate portrayal of personality. The five factor structure has been replicated in different peer reports. However, many of the substantive findings rely on self-reports. Much of the evidence on the measures of the Big 5 relies on self-report questionnaires

Test			
Rating	I....	Rating	I....
	1. Am the life of the party.		26. Have little to say.
	2. Feel little concern for others.		27. Have a soft heart.
	3. Am always prepared.		28. Often forget to put things back in their proper place.
	4. Get stressed out easily.		29. Get upset easily.
	5. Have a rich vocabulary.		30. Do not have a good imagination.
	6. Don't talk a lot.		31. Talk to a lot of different people at parties.
	7. Am interested in people.		32. Am not really interested in others.
	8. Leave my belongings around.		33. Like order.
	9. Am relaxed most of the time.		34. Change my mood a lot.
	10. Have difficulty understanding abstract ideas.		35. Am quick to understand things.
	11. Feel comfortable around people.		36. Don't like to draw attention to myself.
	12. Insult people.		37. Take time out for others.
	13. Pay attention to details.		38. Shirk my duties.
	14. Worry about things.		39. Have frequent mood swings.
	15. Have a vivid imagination.		40. Use difficult words.
	16. Keep in the background.		41. Don't mind being the center of attention.
	17. Sympathize with others' feelings.		42. Feel others' emotions.
	18. Make a mess of things.		43. Follow a schedule.
	19. Seldom feel blue.		44. Get irritated easily.
	20. Am not interested in abstract ideas.		45. Spend time reflecting on things.
	21. Start conversations.		46. Am quiet around strangers.
	22. Am not interested in other people's problems.		47. Make people feel at ease.
	23. Get chores done right away.		48. Am exacting in my work.
	24. Am easily disturbed.		49. Often feel blue.
	25. Have excellent ideas.		50. Am full of ideas.

Figure 4: Questionnaire for Big Five Personality Score



**Algorithm 18** *The  $(\mu, \lambda)$  Evolution Strategy*

```

1:  $\mu \leftarrow$  number of parents selected
2:  $\lambda \leftarrow$  number of children generated by the parents

3:  $P \leftarrow \{\}$ 
4: for  $\lambda$  times do
5:    $P \leftarrow P \cup \{\text{new random individual}\}$ 
6:  $Best \leftarrow \square$ 
7: repeat
8:   for each individual  $P_i \in P$  do
9:     AssessFitness( $P_i$ )
10:    if  $Best = \square$  or  $\text{Fitness}(P_i) > \text{Fitness}(Best)$  then
11:       $Best \leftarrow P_i$ 
12:   $Q \leftarrow$  the  $\mu$  individuals in  $P$  whose Fitness( ) are greatest
13:   $P \leftarrow \{\}$ 
14:  for each individual  $Q_j \in Q$  do
15:    for  $\lambda/\mu$  times do
16:       $P \leftarrow P \cup \{\text{Mutate}(\text{Copy}(Q_j))\}$ 
17: until  $Best$  is the ideal solution or we have run out of time
18: return  $Best$ 

```

Figure 5:  $(\mu, \lambda)$  Evolution Strategies