

Course code and name:	F20BC: Biologically Inspired Computing
Type of assessment:	Group 5
Coursework Title:	Biologically Inspired Computation (F20BC) Coursework
Student Names:	Riffat Khan, Fatima Hanif Mohammed Patel
Student ID Numbers:	H00328074, H00339652

**Declaration of authorship. By signing and uploading this form:**

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.
- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the [University's website](#), and that I am aware of the penalties that I will face should I not adhere to the University Regulations.
- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](#)

**Student Signature:** *Riffat Khan, Fatima Hanif Mohammed Patel*

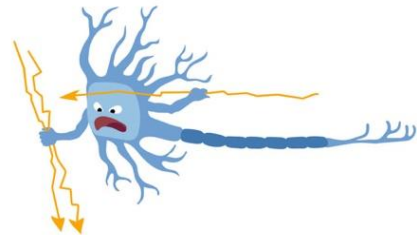
**Date:** 24/11/2023

# F20BC

## Biologically Inspired Computation Coursework

*By Riffat Khan & Fatima Patel*

*Group 5*



**Link to the Notebook:** [F20BC/F20BC Coursework.ipynb \(github.com\)](https://github.com/F20BC/F20BC_Coursework.ipynb)

**Link to the GUI:** [F20BC/ann-gui \(github.com\)](https://github.com/F20BC/ann-gui)

## Table of Contents

<i>1 Setup to Run the PSO Application .....</i>	<i>Error! Bookmark not defined.</i>
<i>2 Implementation.....</i>	<i>4</i>
2.1 Design Choices .....	4
2.2 High-Level Description of Implementation.....	4
<i>3 Experimental investigations.....</i>	<i>5</i>
3.1 Neural Network Hyperparameters .....	5
3.2 PSO Hyperparameters .....	6
<i>4 Results .....</i>	<i>7</i>
<i>5 Discussion and Conclusions.....</i>	<i>10</i>
<i>6 References.....</i>	<i>10</i>
<i>7 Appendices .....</i>	<i>10</i>

Abbreviation	Full Form
Acc	Accuracy
GBL	Global best loss

# 1 Implementation

## 1.1 Design Choices

The ANN has a flexible architecture with a varying number of layers, number of nodes and activation function per layer. This is inputted by the user in the GUI which is then saved in a matrix called *configuration*:

```
[
    number_of_input_nodes,          ← number of input nodes
    [number_of nodes, activation function], ← hidden layer 1
    ...                             ← rest of the hidden layers
    [number_of nodes, activation function] ← output layer
]
```

The weights and biases are saved in a *layer* matrix:

```
[
    [ [weights], [biases] ],          ← hidden layer 1
    ...                             ← rest of the hidden layers
    [ [weights], [biases] ]          ← output layer
]
```

The *position* of a particle is a 1-d array of the above layer matrix flattened:

```
[
    weights_of_first_hidden_layer, biases_of_first_hidden_layer,
    ...,
    weights_of_output_layer, biases_of_output_layer
]
```

The *velocity* of a particle is a 1-d array the same length as *position* of the particle.

## 1.2 High-Level Description of Implementation

*Step 1*, To optimize a neural network using PSO, the user inputs their choice of hyper parameters.

*Step 2*, Then, the *Class Swarm* initializes the swarm with the specified number of particles which each represent a possible solution.

*NOTE*: The position of a particle is a 1-d vector of the weights and biases, and the velocity of the particle is a 1-d vector of the same length.

*Step 3*, The particles are constructed using the *Neural Network class* by looping through the *configuration* and using the *Layer class*, the weights and biases for the hidden layers and output layer are randomly initialized.

*Step 4*, Each particle is assigned a chosen number of *informants* randomly.

*Step 5*, Now to optimize the weights and biases for every particle using a chosen *loss function* for a chosen *number of iterations*, the *fitness* of each particle is calculated at each iteration using *forward pass*.

*Step 6*, The *forward pass* rebuilds the neural network from the *position* vector using the function *unflatten\_weights\_and\_biases*, which takes in the *position* vector and the *configuration* and outputs the *layer matrix*.

The *Neural Network class* is then used to build the structure of the neural network using *config* and sets its weights and biases with this *layer matrix*. The *predictions* are calculated by running the *forward* method from the *Layer Class* on this Neural network. Finally, the loss is calculated by evaluating these *predictions* against the true labels. This is the *fitness* that the *forward pass* returns.

*Step 7*, The particles *best fitness* and *best position* is updated and if this *best fitness* and *position* is better than the *global best fitness* and *position*, the *global best fitness* and *position* are updated. The *informants best position* is updated.

*Step 8*, The particles velocity, and position are then updated using the formula provided in the appendix (and lecture slides).

*Step 9*, After all the *iterations* the train and test data are run on the best solution (i.e., particle) and the accuracy is calculated.

- Informants are introduced to the particles to avoid local minima and improve the global search capability.
- Each particle keeps track of the best solution it has seen and the best solution its informants have seen, fostering both exploitation and exploration.
- The method *update\_velocity* incorporates cognitive, social, and global components, which guide the particles through the search space. This is a unique way to balance the exploration-exploitation trade-off.

## 2 Experimental investigations

We explored various hyperparameters of both the neural network and the Particle Swarm Optimization (PSO) algorithm. This included testing different structures and experimenting with PSO's parameters such as the number of particles. The goal was to observe how these changes impact the performance of the model, particularly in terms of accuracy and loss.

### 2.1 Neural Network Hyperparameters

To find optimal neural network structures, grid search/ random search with ranges (including steps to reduce computational time) of the neural network hyper parameters were run and the network's accuracies and loss were plotted to investigate further.

### Number of Input Nodes

**REASON** User can pick how many features in the dataset to train the model on. Dropping features is useful if they are too many features in the dataset, therefore:

- Reduce the chance of overfitting: since features dropped could be noisy/irrelevant
- Improve Training Efficiency: Fewer features can lead to faster training times, as the model has fewer parameters to learn.
- Explore Feature Importance: users can experiment with different subsets of features, potentially uncovering insights about which features are most predictive.

**RANGE** The range of input nodes that can be tested were between 1 and the number of features in the dataset.

### Number of hidden layers and Number of nodes in each hidden layer

**REASON** To investigate different architectures. More layers/nodes can increase model capacity but also the risk of overfitting and higher computational cost.

**RANGE** The range of the number of hidden layers tested were between 1 - 5 and the number of nodes per layer was 5, 10, 15, 20, 25, 50, 100.

### Activation function used in each hidden layer

**REASON** test different functions (ReLU, sigmoid, tanh) to see how they affect performance. Different functions can lead to varying levels of model non-linearity and convergence speed.

**RANGE** ReLU, sigmoid, tanh was tried for all the hidden layers.

### Number of Output Nodes and Activation function used in output layer

**REASON** Depending on the problem (binary classification, multi-class, regression), different output activations (sigmoid, ReLU, tanh, Identity) can have significant effects.

**RANGE** Since our task was binary classification, sigmoid was mostly used. For other datasets, other output activations were experimented with.

## 2.2 PSO Hyperparameters

To find out the effect of changing the hyper parameters,

### Number of particles

**REASON** More particles can explore the search space more thoroughly but increase computational demand.

**RANGE** 20, 25, 30, 35 and 40.

### Inertia weight (alpha)

**REASON** Influences the exploration and exploitation balance. Investigate how different values affect convergence speed and solution quality.

**RANGE** user can only pick between 0 and 1 from GUI

### Cognitive weight $\beta$ , Social weight $\gamma$ and Global weight $\delta$

**REASON** balance personal best, informants' best, and global best influences on particles velocity.

**RANGE** between 0.1 and 4 since these are the wights that decide the priority of the cognitive/ social/ global

#### Number of iterations

**REASON** More iterations allow more time for convergence but increase computational time. Running too many iterations can lead to overfitting.

**RANGE** 50, 70, 100, 200, 400, 600, 1000

#### Loss function used to calculate fitness

**REASON** The choice of loss function can affect the training direction and speed. It should align with the problem's nature (e.g., classification vs. regression).

**RANGE** For binary classification, we used binary cross entropy loss.

#### Number of informants

**REASON** Affects local search capabilities and how particles influence each other.

**RANGE** Between 4 and 20. If the number of particles is less, the number of informants is also picked to be less so that the social weight is not becoming similar to global weight.

#### Min – Max range for R1, R2, and R3

**REASON** used when generating random coefficients in the velocity update formula since the smaller the random number the smaller the step size in the direction of the cognitive, global, and social weights respectively.

**RANGE** between 0 and 5

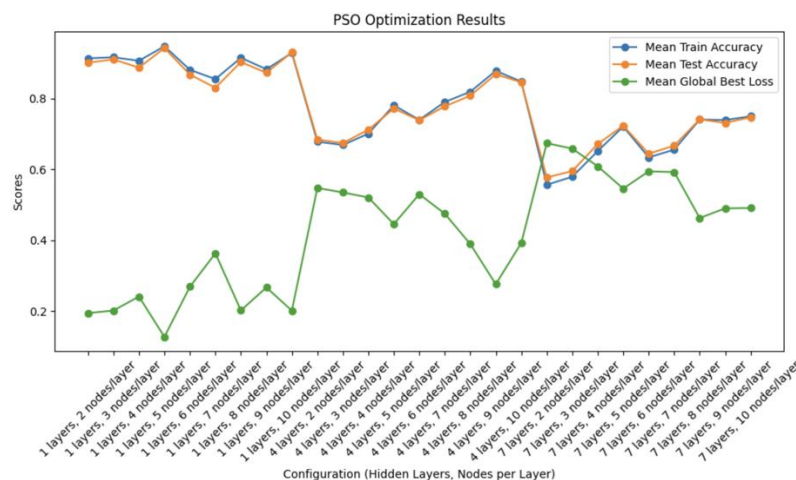
#### Test: Train split

**REASON** to check for overfitting and see if the model works well on unseen data.

**RANGE** From 0.1 up to 0.4 so that the train set is not too small.

## 3 Results

### 3.1 Neural Network Hyperparameter Experimenting & Tuning



The graph above shows the results of running feed forward on a varying number of nodes and layers. As seen increasing the number of layers increases the loss and decreases the accuracy. 1 layer is the optimum number for neural network. Increasing the number of nodes in the layer also decreases the accuracy. The least loss is for 1 layer and 5 nodes.

### 3.2 PSO Hyperparameter Experimenting & Tuning

For optimizing and experimenting with the effects of the PSO hypermeters, we optimized the same neural network with 4 input nodes, 1 hidden layer with 5 nodes and sigmoid activation, and one output node with sigmoid activation. We ran combinations of different hyper parameters and recorded the mean Train accuracy, Test accuracy and mean Global Best Loss after running the PSO 10 times. The train test split used was 0.2

ID	Num of Particles	Num of Informants	$\alpha$ C3	$\beta$ c 0	$\gamma$ c1	$\delta$ c2	Num Iterations	GBL	Test Acc	Train Acc
1	20	4	0.5	0.3	0.2	0.9	200	0.3441	0.828	0.846
2	25							0.1523	0.929	0.943
3	30							0.1142	0.943	0.955
4	35							0.0542	0.977	0.980
5	40							0.0685	0.966	0.975
6	35	6						0.0794	0.972	0.966
7		8						0.1124	0.957	0.945
8		10						0.1752	0.935	0.926
9		13						0.2390	0.914	0.892
10		17						0.1935	0.926	0.911
11		20						0.2906	0.874	0.862
12	40	6					100	0.3305	0.844	0.840
13	40	10						0.2773	0.880	0.867
14	35	4						0.2693	0.870	0.855
15	25	3						0.4571	0.764	0.751
16	20	3						0.4315	0.773	0.748
17	40	4	0.3	0.5	0.3	1.0	120	0.6835	0.577	0.555
18			0.2		0.4	0.8		0.7363	0.454	0.467
19		6	0.3	0.5	0.3	1.0		0.6853	0.577	0.555
20		9	0.2		0.4	0.8		0.7379	0.477	0.482
21		12	0.4		0.2	0.8		0.6788	0.577	0.555
22		16	0.5		0.5	0.5		0.6761	0.577	0.556
23		20	0.5		0.5	0.3		0.6774	0.577	0.556
24		6	0.3		0.3	1.0	200	0.6811	0.577	0.555
25		9	0.2		0.4	0.8		0.7277	0.469	0.478
26		12	0.4		0.2	0.8		0.6823	0.577	0.555
27		16	0.5		0.5	0.5		0.6702	0.605	0.580
28		20	0.5		0.5	0.3		0.6801	0.577	0.555
29						0.7		0.6555	0.571	0.573
30						1.0		0.6026	0.628	0.631
31						1.2		0.5716	0.678	0.686



32	35	4	0.5	0.2	0.5	0.5		0.6648	0.565	0.570
33						1.5		0.4227	0.795	0.804
34						2.0		0.2864	0.875	0.893
35						2.5		0.2249	0.897	0.909

## Experiment Hypothesis

*H1 - With greater number of iterations, the PSO will optimize the neural network to output higher accuracies.*

In ID4 and 14, we observe that despite the same hyper parameters besides number of iterations, the difference in accuracy is 12.5% because a higher number of iterations allows the PSO algorithm to explore and refine the solution space more, potentially leading to better convergence towards the optimal neural network configuration.

*H2- If we increase the number of particles and informants, then give the social weight more priority. The PSO will converge in less iterations since there is sufficient exploration.*

In ID 17 to 28, we observe that this is not the case but the contrary. The accuracies were lower than before and within the same range 57% to 61% besides ID20 and ID25. The Inertia/ Cognitive/ Social/ Global were the same but despite the increase in iterations, the accuracies remained similar at 45 – 47 %.

*H3- Increasing global weight will increase accuracy when they are many particles and many iterations while keeping alpha and gamma at 0.5*

As seen in ID29 – 33, keeping the alpha and gamma at 0.5, while increasing delta from 0.5 to 2.5, increases the accuracy from 57% to 89%. This is because it effectively directed the collective exploration towards optimal solutions while maintaining a balance with moderate cognitive and social weights.

*H4- Increasing test split ratio will increase the loss.*

The same hyper parameters as ID 4 were used with varying split ratios. As seen below, the loss increases (i.e., fitness decreases). Additionally, when the test set is the smallest, the train set gets 98% acc compared to 95.8% on the test set suggesting slight over fitting.

Test train Split	GBL	Test Acc	Train Acc
0.1	0.0534	0.958	0.980
0.2	0.0542	0.977	0.980
0.3	0.0481	0.976	0.985
0.4	0.1052	0.947	0.957
0.5	0.1200	0.944	0.954

*From ID6 to ID11:* As the number of informants increase for many particles (in this case, 35) with the cognitive weight & social weight set relatively less compared to the global weight, the accuracy drops. This is because when the number of informants is high, each particle receives a lot of information from many other particles. However, if the social

weight is low, this information does not strongly influence the particle's movement. Additionally, if the cognitive weight is also low, it means particles are not significantly influenced by their own past experiences either. A high global weight, in contrast, makes particles predominantly influenced by the global best position found so far. Therefore, with less emphasis on individual and social learning (due to lower cognitive and social weights) and more emphasis on global information (due to higher global weight), the swarm might converge quickly to suboptimal solutions, leading to a decrease in accuracy.

## 4 Final Conclusions

The accuracy for the best PSO optimization was 98%. Changing the hyper parameters drastically changed the results for the accuracy.

The results highlight the importance of carefully balancing exploration and exploitation in PSO to enhance neural network training. This balance is important, as too much of either can lead to suboptimal solutions or inefficient convergence as shown in H2.

Increasing the test train split ratio led to higher losses due to the reduced amount of training data and therefore impacting the ability of the model to learn effectively.

Increasing the global weight (delta) while keeping cognitive and social weights moderate showed a marked improvement in accuracy. This suggests that in scenarios with many particles and iterations, emphasizing the global best solution helps steer the entire swarm more effectively towards optimal solutions. It aligns with the literature stating that a higher global weight encourages convergence by prioritizing the collective knowledge of the swarm over individual exploration (Kennedy & Eberhart, 1995).

## 5 Appendices

### 5.1 Setup to Run the PSO Application

install node

check if node is installed by running `node -v`

install npm

check if node is installed by running `npm -v`

install yarn

check if yarn is installed by running `yarn`

`pip install flask-cors`

In the **ann-gui** directory, run, `yarn run dev` to run the server for server-side commands, then, to navigate to the backend directory, run `python ./api.py`

To view the webpage in your web browser open `localhost:3000`

## 5.2 Formula for Updating Particle Velocity and Position

For  $i$  in each particle,

For  $d$  in each dimension,

$$v_{i+1}^{(d)} \leftarrow \alpha v_i^{(d)} + \beta r_1 (p_i^{(d)} - x_i^{(d)}) + \gamma r_2 (l_i^{(d)} - x_i^{(d)}) + \delta r_3 (g_i^{(d)} - x_i^{(d)})$$

$$x_{i+1}^{(d)} \leftarrow x_i^{(d)} + v_{i+1}^{(d)}$$

Where:

$x$  ← current position

$p$  ← particles best position

$l$  ← informants' best position

$g$  ← global best position

$\alpha$  ← inertia weight

$\beta$  ← cognitive weight

$\gamma$  ← social weight

$\delta$  ← global weight

$r_1$  ← random number 1 within range chosen by user.

$r_2$  ← random number 2 within range chosen by user.

$r_3$  ← random number 3 within range chosen by user.

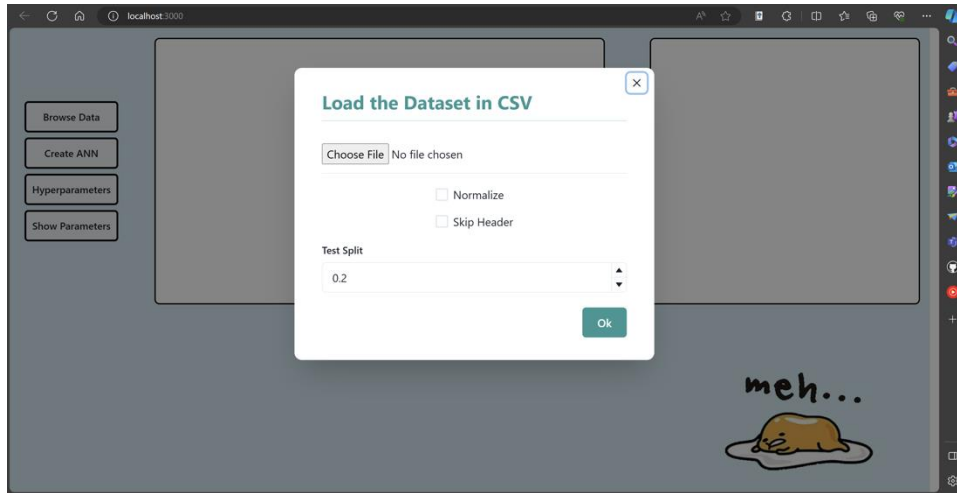
## 5.3 References

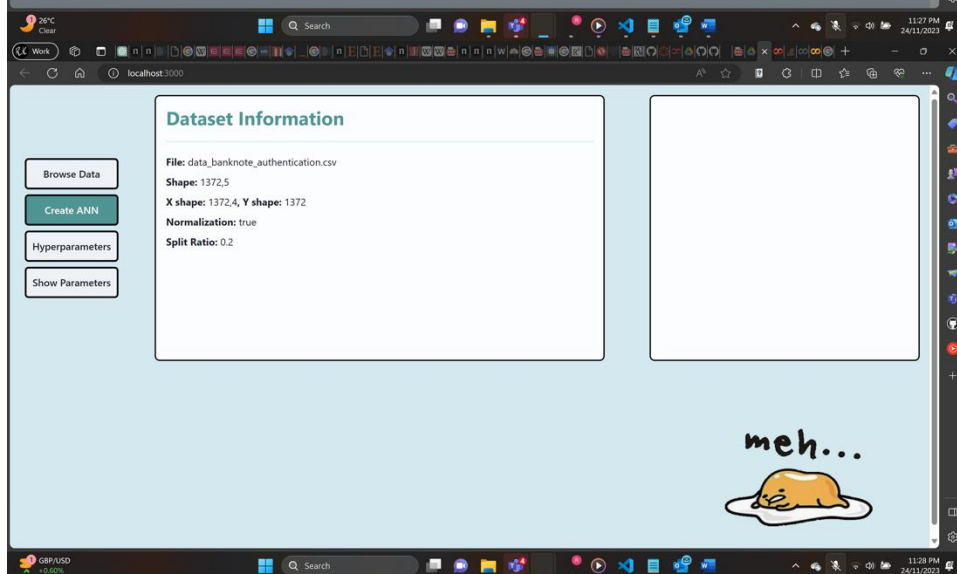
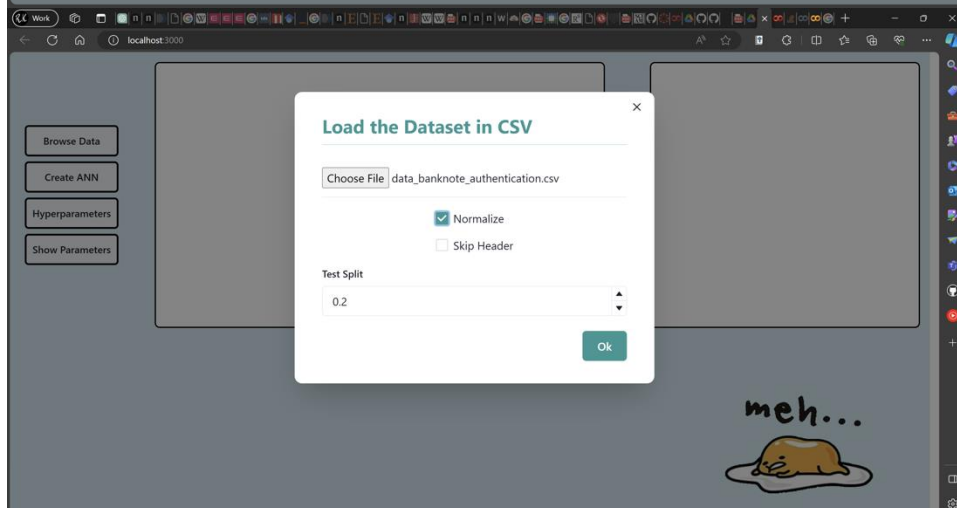
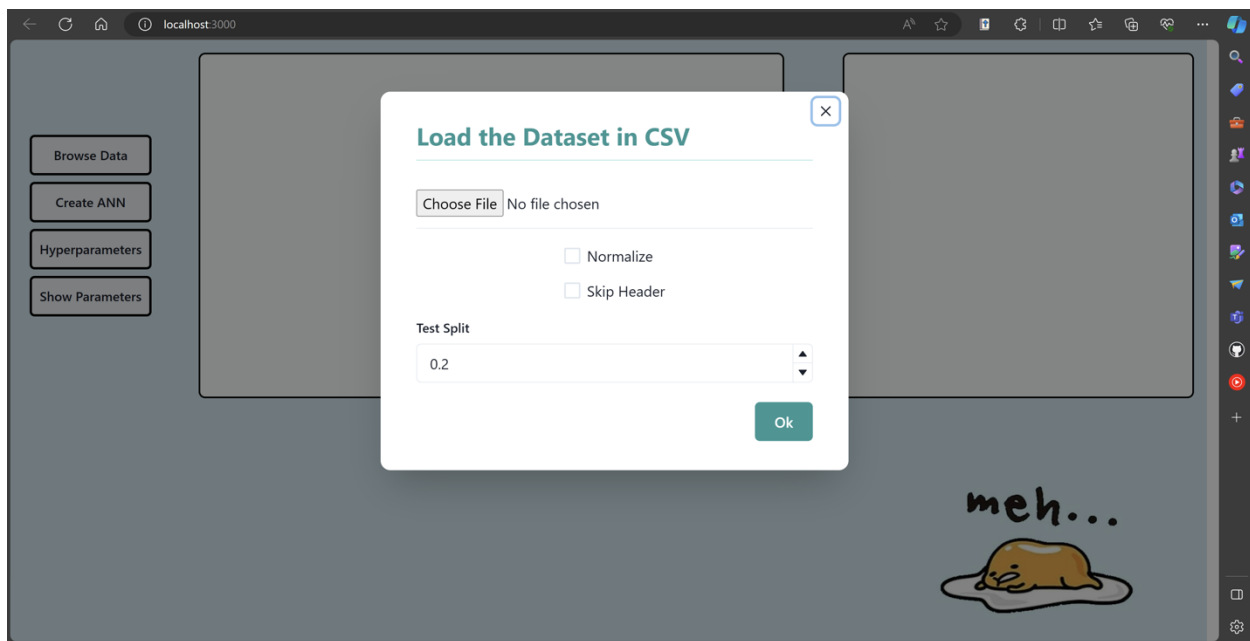
Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4. IEEE, pp. 1942-1948.

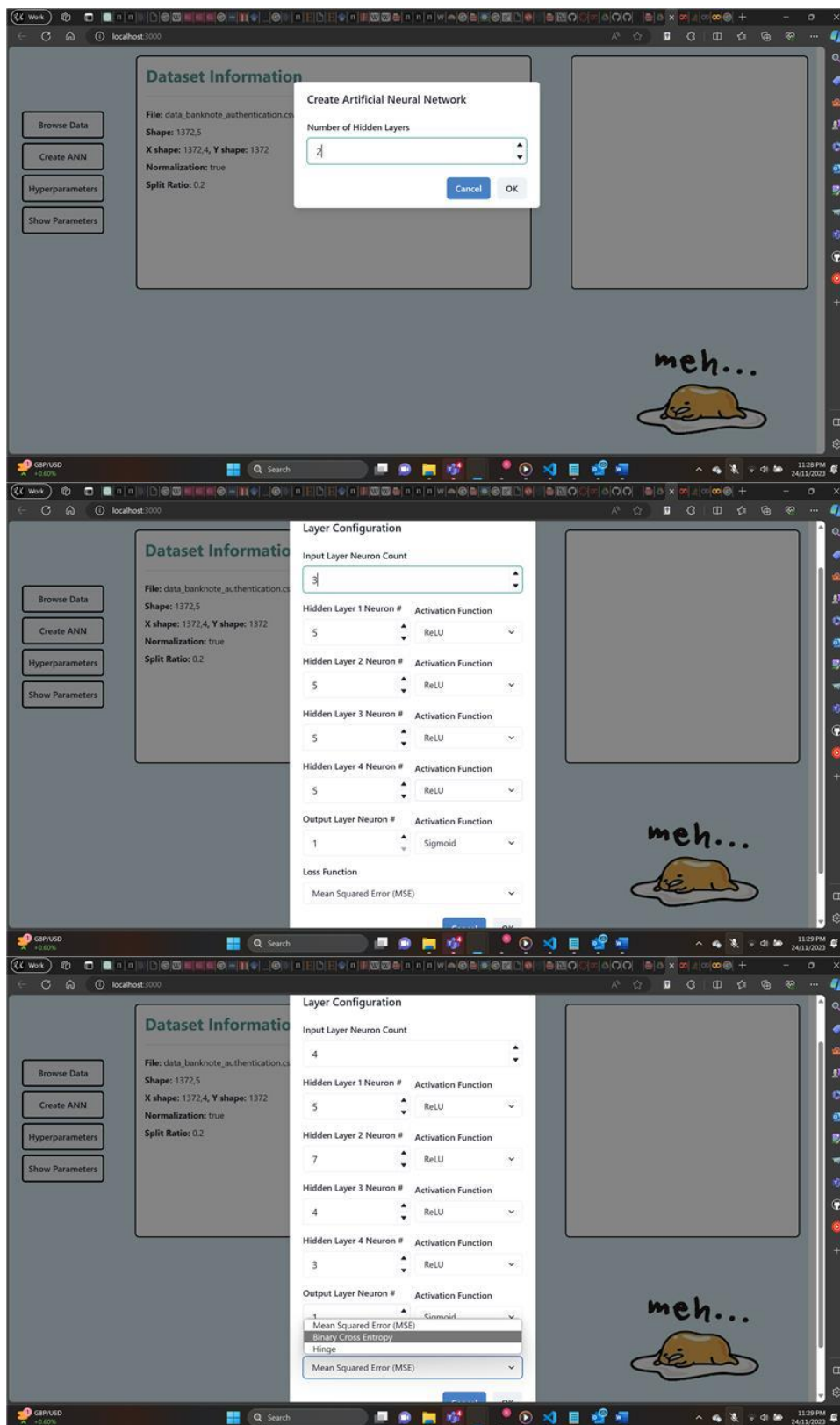
@article{mazaheri\_shahryar\_rahnamayan\_azam\_asilian\_bidgoli\_2023, title={Designing Artificial Neural Network Using Particle Swarm Optimization: A Survey}, url={https://www.intechopen.com/chapters/82833},

DOI={https://doi.org/10.5772/intechopen.106139}, journal={Artificial intelligence}, author={Mazaheri, Pooria and Shahryar Rahnamayan and Azam Asilian Bidgoli}, year={2023}, month={Feb} }

## 5.4 Screenshots of GUI







Browse Data

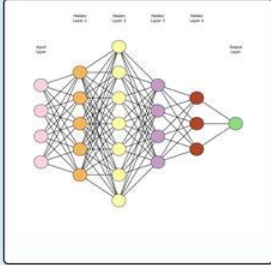
Create ANN

Hyperparameters


Show Parameters

### Artificial Neural Network (ANN) Configuration

Input Layer: 4 nodes  
Hidden Layer 1: 5 neurons, Activation Function: relu  
Hidden Layer 2: 7 neurons, Activation Function: relu  
Hidden Layer 3: 4 neurons, Activation Function: relu  
Hidden Layer 4: 3 neurons, Activation Function: relu  
Output Layer: 1 nodes, Activation Function: sigmoid  
Loss Function: binaryCrossEntropy



meh...



Browse Data

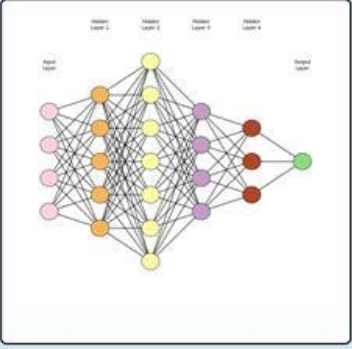
Create ANN

Hyperparameters

Show Parameters

### Artificial Neural Network (ANN) Configuration

Input Layer: 4 nodes  
Hidden Layer 1: 5 neurons, Activation Function: relu  
Hidden Layer 2: 7 neurons, Activation Function: relu  
Hidden Layer 3: 4 neurons, Activation Function: relu  
Hidden Layer 4: 3 neurons, Activation Function: relu  
Output Layer: 1 nodes, Activation Function: sigmoid  
Loss Function: binaryCrossEntropy



meh...

