Published in AJSE, Vol:21, Issue: 2
Received on 5th April 2022
Revised on 22th October 2022
Accepted on 23rd November 2022

AIUB JOURNAL OF SCIENCE AND ENGINEERING
ISSN: 1608 – 3679 (print) 2520 – 4890 (Online)

In-Depth Case Study on Artificial Neural Network Weights Optimization Using Meta-Heuristic and Heuristic Algorithmic Approach

Partha Sutradhar, Victor Stany Rozario

Abstract—The Meta-heuristic and Heuristic algorithms have been introduced for deep neural network optimization in this paper. Artificial Intelligence and the most used Deep Learning methods are getting popularity in these days, thus we need faster optimization strategies for finding more accurate results in the future. Neural Network Optimization with Particle Swarm Optimization, Backpropagation (BP), Resilient Propagation (Rprop), and Genetic Algorithm (GA) have been used for numerical analysis of different datasets and compared with each other to find out which algorithms work better for finding optimal solutions by reducing training loss. Meta-heuristic algorithms GA and PSO are higher-level formulas and problem-independent techniques that may be used for a diverse number of challenges. The characteristic of heuristic algorithms has extremely specific features that vary depending on the problem. The conventional backpropagation (BP) based optimization, genetic algorithm, particle swarm optimization, and resilient propagation (Rprop) are all fully presented, and how to apply these procedures in artificial deep neural networks optimization is also thoroughly described. Applied numerical simulation over several datasets shows that the Meta-heuristic algorithm particle swarm optimization and also the genetic algorithm performed better than the conventional heuristic algorithm like backpropagation and resilient propagation over these datasets. Evaluation of these algorithms was done based on training epoch and their error convergence.

***Index Terms*— Artificial Neural Networks, Resilient Propagation (Rprop), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Backpropagation (BP), Meta-Heuristic Algorithm, Heuristic Algorithm.**

I. Introduction

There has been a significant amount of growth in Artificial Intelligence, Machine Learning, and Reinforcement Learning, and Deep Neural Network have been seen during the earlier few decades. Therefore, we need faster learning and optimization algorithms for better and more efficient computational benefits. We have selected Particle Swarm Optimization, Genetic Algorithm (GA), and Backpropagation (BP) and, Resilient Propagation (Rprop) algorithms for research work.

Partha Sutradhar is a graduate student of the Department Computer Science, American International University-Bangladesh, Dhaka-1229, Bangladesh. Email: partharaj.dev@gmail.com.

Victor Stany Rozario is an Assistant Professor in the Department of Computer Science under the Faculty of Science and Technology at American International University-Bangladesh, Dhaka-1229, Bangladesh. Email: stany@aiub.edu.

Especially, Neural Network is also a bio-inspired based Artificial Neural Network which consists of many dense layers and each layer has a set of neurons called nodes, synapses, and, biases. Artificial Neural Network learns by minimizing the cost of the network predicted by the neural network. For this purpose, we need to alter the weights in such a manner that the weights in the neural network provide better and more accurate prediction, and to update these weights we can use meta-heuristic and heuristic algorithms to minimize the cost of the network. Most meta-heuristic systems are generic and can be functional to a diverse range of problem sets, whereas heuristic methods are often created for a specific array of problems. Here, we are going to compare generic bio-inspired algorithms known as Particle Swarm Optimization and Genetic Algorithm (GA) with application-specific algorithms like Backpropagation (BP) and Resilient Propagation (Rprop).

Neural Networks are prediction-based computing methods inspired by the nervous systems of our biological brain and are now applicable in most high-level intelligence system designing [1]. The neural network learns from numerous input patterns from the datasets and updates the synaptic connection weights for achieving the accurate and expected output [1]. Neural Network gets the input and then feeds it forward through neurons to the output layer. Then, the output layer produces the result, if the result is not expected then we need to update the synaptic weights vectors for the expected result. There are several algorithms used in the development of updating the weights for better performance. Traditionally, Backpropagation (BP) algorithm is used. This paper validates some of the bio-inspired algorithms named meta-heuristic algorithms that can achieve better performance than application-specific Backpropagation and Resilient Propagation which is a heuristic algorithm. There is some implication of a meta-heuristic algorithm. Implications of the meta-heuristic algorithm are, that it gets stuck in local minima and stays there for quite some time which makes the training period lengthy.

II. Literature Review

This is the era of Artificial Intelligence that is governing the world with the power of Neural Networks that has been reversed and engineered from our biological brain functionality. Therefore, it is very much crucial to optimize the network learning capacity. There are a lot of algorithms that can optimize neural network weights.

Jiri Stastny [9] et al proposed a scheme of a simpler network configuration for the learning process of neural networks. They have designed a Topology for Artificial Neural networks for effective learning using the Genetic Algorithm. The genetic Algorithm looks for precise settings of neural network weights and if fails it practices minimization of its given function [9]. A stochastic heuristic approach is recognized as the Genetic algorithm. Adaptive and evolutionary pathways in living organisms encourage genetic algorithms [9]. The major applications that use Genetic Algorithms are in analytical problem solutions domain which are unknown or extremely complicated. A genetic algorithm allows us to get to a solution to a numerical problem that is not known using evolutionary use cases [9].

Alan Mosca [11] et al proposed Adapting Resilient Propagation for Deep Learning by using an adjustment of the Resilient Propagation (Rprop) that included the standard Rprop steps with a special dropout technique. They have applied the methods of Deep Neural Networks as individual modules and ensembles in formulations [11]. Dropout [11, 12] is a regularization process in which just a random subset of nodes in the network are updated during each training iteration, but the entire network is used at the final evaluation. They have compared their Adapting techniques with Stochastic Gradient Descent, unmodified Resilient Propagation, and modified Resilient Propagation to speed up the training process [11].

Garro, Beatriz A et al [22] proposed a method that designs an Artificial Neural Network using PSO algorithms automatically. The goal of these strategies is to generate all three major components of an ANN simultaneously moment: synaptic weights, connections or architecture, and transfer operations for every neuron. To evaluate the efficiency of each approach and discover the optimum design, eight different fitness functions were offered. Furthermore, the proposed approach is associated with those created by manually utilizing the recognized Back-Propagation and Levenberg-Marquardt Learning Algorithms [22]. Finally, the method's accuracy is evaluated using a variety of nonlinear traditional classification challenges.

Researchers are trying to combine meta-heuristic and heuristic algorithms to predict some big events like earthquakes and natural disasters. PSO-BP combined structure has been introduced to predict big natural disasters like earthquakes [23].

III. Deep Neural Network Optimization

Fundamentally, Deep Neural Network is an Artificial Neural Network consisting of many dense layers and each layer contains a set of neurons, synapses, and biases. This allows computational models with several layers of nodes to learn multiple degrees of abstraction for data representations. A deep Neural Network is set with many layers in this way as shown in Fig. 1. There are a lot of algorithms that are capable of optimizing the weights of a simple neural network [16, 17].

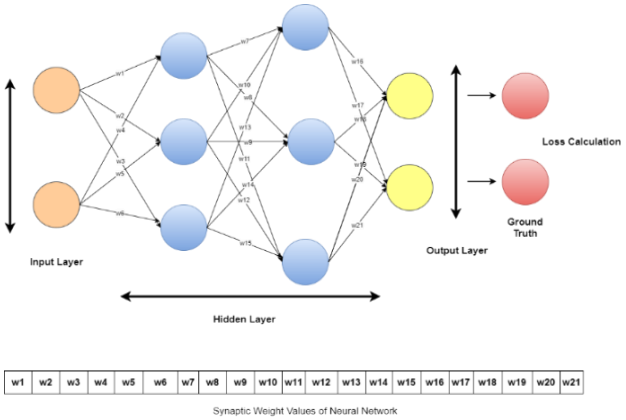

Fig. 1. Deep Neural Network

In the field of neural network optimization, there has already been a lot of improvement. Neural Networks have vastly enhanced the region in the detection of objects, pattern recognition, simulation, and predictions. In Neural Network we take inputs first, forward propagate the input values through the network and get outputs in the output layer which is the last layer of the Neural Network then we compare outcome values with target values and calculate the loss. So, the loss of the neural network determines how wrong a prediction was so it takes as input the predicted outputs and compares them with ground truth outputs. If those two things are extremely far away, the loss will be quite great. However, the nearer these two things are to each other than the lower the loss will be and the more accurate the model will be. Therefore, we should minimize the loss we want to incur if we want to predict something as close as possible to the ground truth. There are many loss functions like cross-entropy, log-loss, Mean Squared Error, Root Mean Squared Error (RMSE).

A. Loss optimization: To use this loss function to train the weights of our neural network such that it can learn that problem well what we want to do is to find the weights of the neural network that will minimize the loss of our dataset.

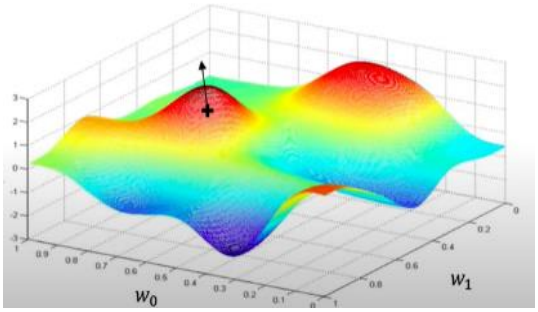

Fig. 2. Problem Search Space

A deep Neural Network needs to update the weights of its network to reduce the cost or loss according to its loss function. Fig.2 represents that we can go through the search space to find out the optimal solution.

$$MSE = \frac{1}{N}\sum_{i-1}^{N}(Y_i - \hat{Y}_1)^2 \qquad (1)$$

Here, MSE represents Mean Squared Error and we are using this loss function to optimize the Artificial Neural Network using the preferred algorithms. In this research context, we have described the BP, PSO, GA, and Rprop algorithm for optimizing the neural network weights. This algorithmic approach could lead us to find which one is better for Neural Network optimization.

IV. Particle Swarm Optimization

Algorithm Particle Swarm Optimization is inspired by the social behavior of biological environmental life form process. It is an evolutionary algorithm invented by Eberhart and Kennedy in 1995. It is an iterative method that optimizes a problem by improving its particle position. It has several particles as a population and every individual particle in the population has the neural network synaptic weight values. The algorithm's work is to move particles around the problem search space using the position and velocity of the particles and find the optimal solution in the iterative method [15].

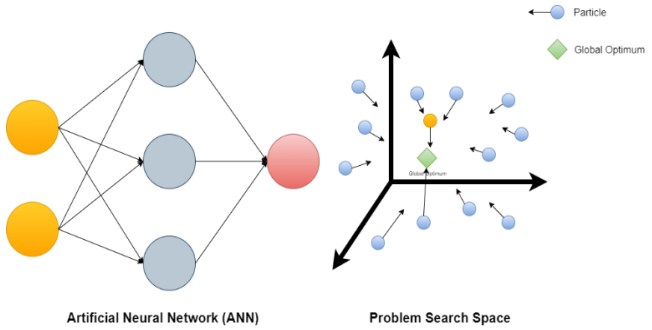

Fig. 3. Neural Network & PSO Search Space

Here, each particle in Particle Swarm Optimization holds the synaptic weight values of a neural network as position in search space and uses the position and velocity of the PSO algorithm's hyperparameter to alter the position and velocity of each particle to exploration for the global optimum of the artificial neural network. This is an iterative method and after each iteration, the particle gets closer to the optimum of the neural network. Here, x is the position vector $x_i^n = x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in},)$ and velocity v vector $v_i^n = v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in})$. Looking at the position vector and velocity vector n is the iteration for each of the particles and i is the responsible particle working on that n-th iteration. Below is the equation of particle swarm optimization search:

$$V_{id}^{n+1} = wV_{id}^{n} + c_1 r_1 \left(p_{id}^{n} - x_{id}^{n}\right) + c_2 r_2 (p_{gd}^{n} - x_{id}^{n}) \quad (2)$$

$$X_{id}^{n+1} = X_{id}^{n} + V_{id}^{n+1} \quad (3)$$

In equation 2. *w* is the inertia weight and c_1 cognitive weight and c_2 is social weight in this algorithm and r_1 r_2 is a random value between 0 < r < 1 [4]. Local Best and Global Best particles take place to find the optimal solution for a neural network. The global best is where the best fitness is found and the local best is the particle amongst all the particle's positions [10]. The neural network optimization happens and learns the best weights from the position of the neural network. In Fig. 3, we see that particle looks for the global optimum of the neural network and calls other particles to come to (Global Best) GBest particle because GBest holds the current best position of neural network weights and all other (Particle Best) PBest particles moves to GBest particle according to new velocity and position [15]. Again, in the next iteration, the Particle position changes due to velocity and randomness [15]. Therefore, the GBest Particle as known as Global Best in search space will eventually find the best global optimum position of the neural network. PSO algorithm is also an exploration algorithm because it searches around all the search space. We can integrate this algorithm for Neural Networks like the ANN-PSO model [15].

V. Backpropagation

A basic Neural Network is made up of the input neurons, hidden units neurons, and output neurons. Neural Network is really good for non-linear predictions and generally used for the problem-specific dataset. Neural Network has two parts for the training phase, first Forward Propagation and then Backward Propagation. In Forward Propagation, the values go to input vector values and go through the network and each activation function produces an output for all the hidden layers. Then, the output layer gives an estimated result. If the result is not satisfied then we use backward propagation using loss calculated from the last layer and update the weights according to gradient-based optimization. We also use the momentum factor for faster optimization of neural networks in Backpropagation (BP).

1. Forward Propagation

First, the Phase of the Neural Network is to forward propagate the input vector to the hidden layer and generate outputs in the output layer. Each layer has neurons and every neuron has synaptic weight connected to the previous layer. Neuron work is to calculate the weighted sum from all the connected synaptic weights, plus adds bias 1, and then applies an activation function to generate output from that individual neuron. Below eq. 4 is the logistic function also known as the sigmoid function and eq. 5. is a derivative of a logistic function of eq. 4.

$$f(x) = \frac{1}{1+e^{-x}} \quad (4)$$

$$f'(x) = f(x) * (1 - f(x)) \quad (5)$$

The whole idea of forwarding propagation is to feed forward the input vectors from the datasets then generate outputs and calculate the loss at the output layer. Then Backpropagation takes place to update the weights for optimizing the network to generate satisfied predicted outcomes and reduce the loss.

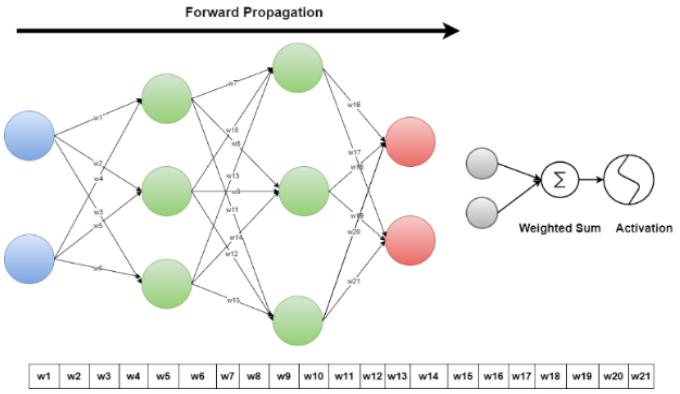

Fig. 4. Forward Propagation

2. Backward Propagation

The second, Phase is to update weights according to the calculated loss from the last layer of the artificial neural network using the gradient descent approach. Gradient Descent Optimization is also known as Backpropagation (BP) the main usage of the Backpropagation (BP) algorithm is to update weights to train the artificial neural network to learn from the dataset and make accurate predictions after training. Now, the main work is done in the last layer of the neural network which calculates the total loss of the neural network using a simple error function [18].

$$E_{total} = \sum \frac{1}{2} (target - output)^2 \qquad (6)$$

After calculating loss, synaptic weights are updated regarding the weighted error found from the last layer to the first layer.

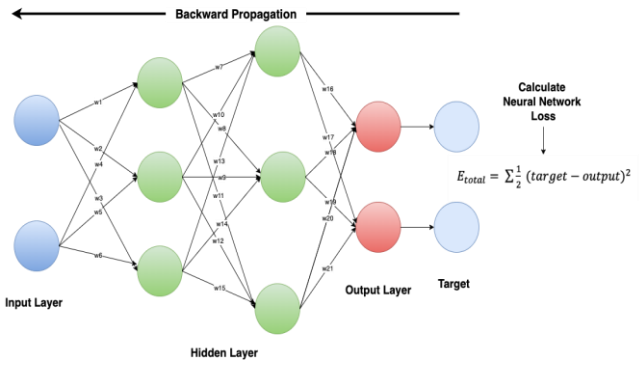

Fig. 5. Backward Propagation (Backprop)

Here, looking at Fig. 5, we see that the total loss of the error is calculated at the last layer and then synaptic weights are updated according to a weighted error we have found using Parts of the gradient computation from one layer are reprocessed in the gradient computation for the preceding layer. Iteratively, the neural network becomes trained sufficiently to forecast accurately after a period of 10000 to 100000 iterations for forwarding and backward pass.

$$\Delta w_{ij} = \left(n * \left(\frac{\partial E}{\partial w_{ij}}\right)\right) + \left(mu * \Delta w_{ij}^{t-1}\right) \qquad (7)$$

Here, eq. 8 shows weights updates with momentum. Momentum increases the speed of finding the optimal result for the neural network [18].

VI. Resilient Propagation

Resilient Propagation is also known as the Rprop algorithm, an efficient new form of the Backpropagation (BP) algorithm. This algorithm is a direct alteration of the synaptic weight step based on the knowledge of the local gradient found while training. Rprop is a powerful gradient descent approach that approximates updates only depending on gradient signs. It stands for Resilient Propagation and is useful in a wide range of contexts since it dynamically optimizes the step-wise size for each weight independently. Many different types of algorithms depend on the magnitude of the gradient and signs to find the global minima [20]. It often works fine but it is not always a good option, some of the time works badly and does not contain the valuable information it needs. Using gradient magnitude to optimize weights becomes questionable [20]. Also using a settled learning rate to find the global minima fails. That's why Resilient Propagation (Rprop) ignores gradient magnitude for better performance. Modern gradient descent variants use dynamically adapting step sizes for overcoming this problem. Resilient Propagation uses the positive and negative signs of the gradient to find the global optimal solution. Resilient Propagation algorithm, synaptic weights are updated by eq. 5 [21].

$$w_i^t = w_i^{t-1} - n_i^{t-1} * sgn\left(\frac{\partial E^{t-1}}{\partial w_i^{t-1}}\right) \qquad (8)$$

Here, n_i^t is the step size for the t-th iteration of the gradient descent and i-th weight. Where the sign of the partial derivative of the error concerning the corresponding weight is determined in the last step Using the given step size, we advance in the direction of descent. For adapting the step size for each iteration Resilient Propagation (Rprop) follows the formula for updating the scheme [6, 21], eq. 9.

$$n_i^t = \begin{cases} \min(\eta_i^{t-1} * \alpha, \eta_{max}) \ \ if \ \frac{\partial E^t}{\partial w_i^t} * \frac{\partial E^t}{\partial w_i^{t-1}} > 0 \\ \min(\eta_i^{t-1} * \ \beta, \eta_{\min}) \ \ if \ \frac{\partial E^t}{\partial w_i^t} * \frac{\partial E^t}{\partial w_i^{t-1}} < 0 \\ n_i^{t-1} \quad otherwise \end{cases} \qquad (9)$$

Here, $\alpha > 1 > \beta$ is the step size of the algorithm, and depending on α, β algorithm decides whether speed should be increased or decreased. The most accepted values of α and β are 1.2 and 0.5. Though Resilient Propagation (Rprop) works well in many scenarios, it is not quite good yet in finding solutions for more complex problems.

VII. Genetic Algorithm

A Genetic algorithm is an algorithm inspired by the course of natural selection. It is also known as a Meta-heuristic algorithm, above and beyond it is an evolutionary algorithm that is collected from natural behavior[19]. A genetic algorithm is used to solve high-quality problems solutions that are based on search and optimization of problems. Genetic Algorithm (GA) uses population, mutation, crossover, and selection for optimization, and is also used in numerous search problems. We can use a Genetic Algorithm to optimize neural network synaptic weights to find the problem's best weights and reduce the total datasets' loss. Looking at Fig. 1 we see that, the artificial neural network has w1 to w21 synaptic weights, if we can find the best weights for any specified dataset then network prediction will be accurate.

A genetic algorithm finds the best individual neural network from its population pool then it copies the same characteristic or trained weights to other networks. Here these individuals have been taken for getting more outcome variance so that the best individual can be different from the parent weights vector and still produce close to accurate of its ground truth value.

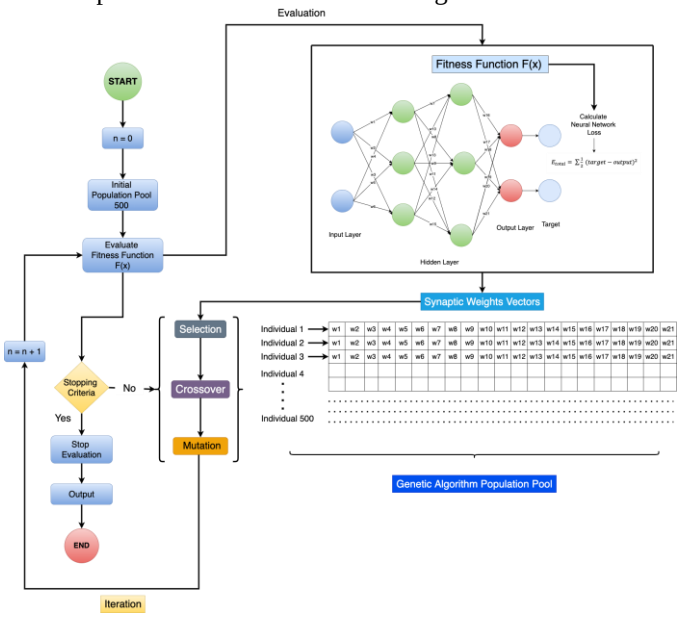

Fig. 6. Genetic Algorithm and Neural Network

The Genetic Algorithm (GA) algorithm has a population pool where a number of a chromosome are there and each chromosome has a gene. Chromosomes consist of genes in a genetic algorithm inspired by nature. Here, for the optimization of neural networks, we can use a genetic algorithm [19]. Each chromosome represents a neural network and every gene in chromosomes is denoted as weight.

Fig. 7. Selection

The genetic Algorithm (GA) selects the best two individual chromosomes according to the best fitness from the population pool for the crossover process. Parent 1 and Parent 2 chromosomes have synaptic weights as genes in Fig. 7.

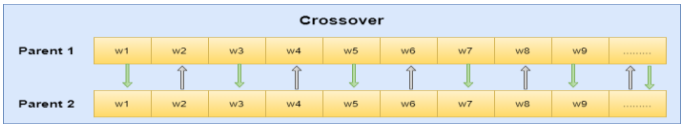

Fig. 8. Crossover

After that crossover happens, Parent 1 and Parent 2 exchange their genes and create new offspring from the two individual parents. Here, flip the coin happens when genes from chromosomes crossover to make a new individual [19].

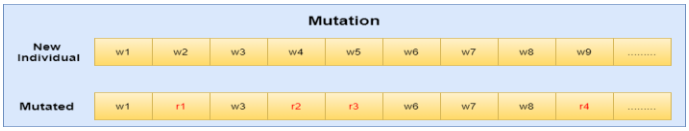

Fig. 9. Mutation

In the mutation part, synaptic weights values randomly change from some of the genes created from new offspring. Here, r1, r2, r3, and r4 are random values produced from a threshold for mutation. Now, the new offspring has its natural and individual character. Each chromosome in the population has gene values, we can set the values of synaptic weights in the chromosome as genes. Then, we calculate the fitness function of the neural network to check the total loss of the network. If the criteria of the fitness function meet essential requirements then we stop the iterative training of the genetic algorithm. When criteria do not meet the requirements of the fitness function then we select the two best individuals as parents from the population pool according to the best fitness and we use the crossover technique or mating to create new individuals from the parents by exchanging their genes [19]. To do that we can use the flip the coin technique as randomness. After that, we need to make the new individual different from the parents. For that, we can use mutation to change some of the synaptic weights as genes randomly from the chromosome. Thus, a new offspring is created. Then, iteratively we insert the offspring into the population pool for again evaluating the fitness function. In this iterative way, we can find the best synaptic weights from the genetic algorithms by finding the lowest F(x) error meaning the mean squared error for neural network training. Each iteration is also known as generation in a Genetic Algorithm [19].

VIII. Research Methodology

The Research Methodology has two distinct types of points of view. 1) Data Acquisition and 2) Applying Algorithms. Using the encog machine learning library [13], we see the results of the algorithms that are described in the paper which include Resilient Propagation, Particle Swarm Optimization, and Backpropagation also a customized version of the Genetic Algorithm which was capable of optimizing Neural Networks weights and also see the induced effects. Here, we have used the epoch and saw how the resource utilization comes into

functioning and gives us the training loss results. It is also possible to calculate training time by averaging each iteration but we are going to use the epoch as an outcome. We have used the Graph Tool known as matplotlib and NumPy to see the effects in the classification process.

1. Dataset Acquisition

Datasets were acquired from the Kaggle. These datasets are open source and good for the classification process. Besides, we can make use of these datasets to make a strong understanding of the neural network optimization process. There is a total of 713 observations for this research that is included here to see the induced effects.

A. Xor gate

Xor or exclusive OR gate is a simple logic gate that only produces true results when its inputs are different. It is the standard approach for an artificial neural network to see if it can draw the separation line to classify the logical gate and is capable of doing other classification tasks. This has two input variables with one targeting output variable with four observations.

B. Iris dataset

The Iris Flowers Dataset is a simple dataset containing 150 observations of flower dimensions. It has been used to identify flower species based on their dimensions. The dataset has four features containing the width and length of petals, sepals, and three classes Iris Setosa, Iris Virginica, and Iris Versicolor.

C. Sonar dataset

The Sonar Dataset is concerned with determining if the entity is a Mine (M) or a Rock (R) based on the strength of sonar returns at multiple viewpoints. This dataset contains 208 observations with 1 output variable and 60 input variables. The output variable determines if the object is rock or mine.

D. Ionosphere dataset

The Radar System yields pointing free electrons found in the ionosphere, and the Ionosphere Dataset demand the forecasting of weather behavior in the atmosphere. It's a two-class binary classification problem. There are 34 input variables and 1 outcome variable and found 351 observations were. Besides, the total of observations per class is not evenly distributed. The output variable determines if the atmosphere is Good (G) or Bad (B).

2. Applying Algorithms

We have used 1 hidden layer with 10 Neurons in the XOR dataset and Iris dataset and 2 hidden layers with 10 neurons for the Sonar dataset and Ionosphere dataset because these two datasets have a huge amount of input in the input layer of the Artificial Neural Network. Hidden Layer uses Sigmoid Activation Function. We have used Encog Machine Learning [13] to produce the training losses and also used JFreeChart for implementing the Graph. For the Genetic Algorithm, we have used 500 population sizes as a default population pool of neural networks. In Fig.10, we are seeing the whole process of that we are applying a different algorithm for Artificial Neural Network weights optimization. Here, we have used Particle Swarm Optimization, Resilient Propagation, Genetic Algorithm, and Backpropagation algorithm in this paper.

In this research paper, We have used the sigmoidal function for the neural network activation function. In Fig. 10. We have prepared the process in phases. In the first phase, we have prepared the data for neural network input and output so that it can fit easily in the neural network input layer and output layer for each dataset because each dataset's input and output look different from the other dataset. After the data preparation phase, the algorithm selection phase is started. Here, we have to select an algorithm to train the neural network to get the training outcome. Next, the training phase started and after 1000 epochs, we stop the training to get to the output phase. In the output phase, we store the neural network loss. This process is again repeated until all of the algorithms are used on the provided datasets. In this way, we can find the results of the neural network losses. The process shows that we can optimize Neural Networks with different algorithms like Bio-Inspired PSO and GA and application-specific algorithms BP and Rprop. We have used four datasets to see the results of these algorithms that can reduce the training loss of neural networks and provide an optimal solution.

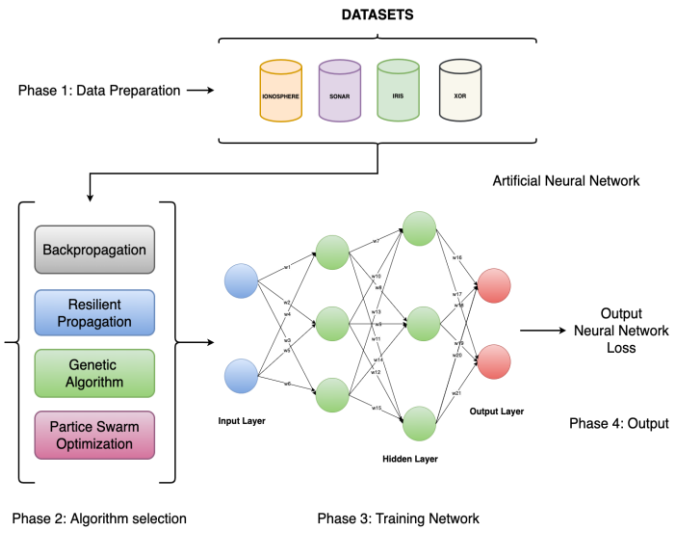

Fig.10. Training Neural Network with Algorithms

Besides, we have used the matplotlib library to implement multiple training losses in one graph. This type of work gives us a visual idea. Nowadays, we are seeing a lot of growth using artificial intelligence that uses a neural network. Therefore it is very important for us to provide a more sophisticated algorithm or optimization technique that leads us to accurate results.

IX. Results and Analysis

In this section, the findings and performed simulations are analyzed by training losses or errors of the Artificial Neural Network by using meta-heuristic and heuristic algorithms that are described in the paper. Observing the training error reductions while training Neural Network will give us the best

optimal solution for each dataset we are using here. The lowest the training loss value higher the accuracy will be. This case study provides a comparison between the meta-heuristic algorithm and heuristic algorithm using the xor gate, iris dataset, sonar dataset, and ionosphere dataset. In this study, we have found the particle swarm algorithm was better in all the cases and also produced much better results than other algorithms. The results that are found while performing the algorithmic approach for Artificial Neural Network optimization are given below:

1) Here, we going to see the effects of the training error *results* of each algorithm described in the paper applied in simple the XOR dataset. Each of the algorithms will try to adapt the weight to minimize the loss function of the neural network.

Here, we have used the Backpropagation algorithm in the XOR dataset to reduce the loss or training error of the artificial neural network. Fig. 11 shows the convergences of neural networks for the xor gate.

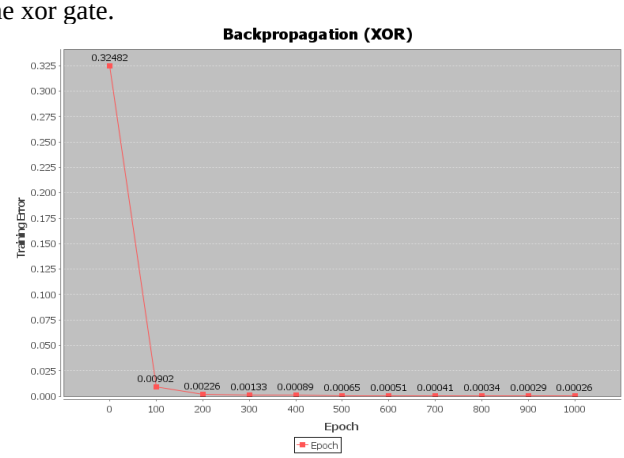

Fig.11 Applied Backpropagation (BP) on XOR Gate

The Resilient technique was applied to the XOR dataset to reduce the artificial neural network's loss or training error. Fig.12 illustrates the convergences.

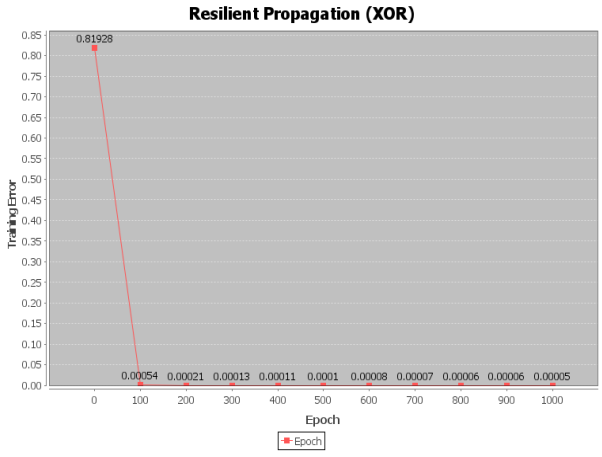

Fig.12 Applied Resilient Propagation (Rprop) on xor gate

In Fig.12, we have applied the Particle Swarm Optimization algorithm which is also known as PSO an evolutionary algorithm to reduce the neural network error function loss. After applying the algorithm network compellingly reduces in less than a few epochs.

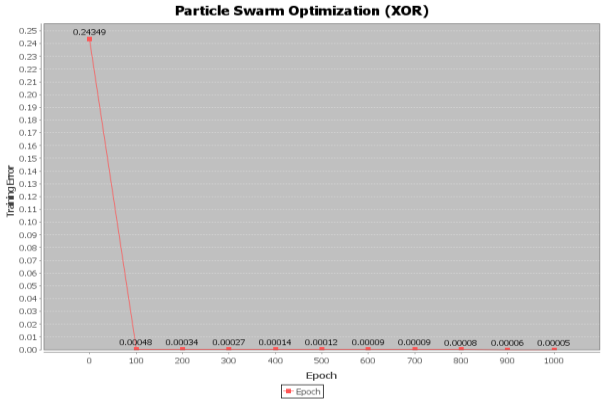

Fig.13 Applied Particle Swarm Optimization on xor gate

Genetic Algorithm (GA) with Artificial Neural Network combined structure shows that it has learned the problem set very quickly and found the optimal solution. Thus, we are seeing a close to zero figures graph in Fig.14.

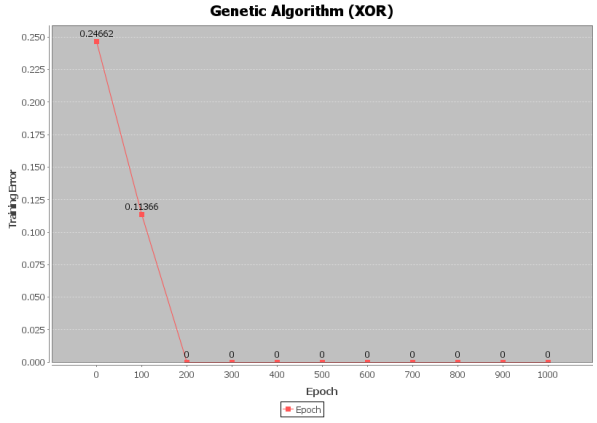

Fig.14 Applied Genetic Algorithm (GA) on xor gate

Table. 1

Epoch	Training Loss			
	BP	Rprop	GA	PSO
0	0.32482	0.81928	0.24622	0.24349
100	0.00902	0.00054	0.11366	0.00048
200	0.00226	0.00021	5.0E-41	0.00034
300	0.00133	0.00013	5.0E-41	0.00027
400	0.00089	0.00011	5.0E-41	0.00014
500	0.00065	0.00001	5.0E-41	0.00012
600	0.00051	0.00008	5.0E-41	0.00009
700	0.00041	0.00007	5.0E-41	0.00009
800	0.00034	0.00006	5.0E-41	0.00009
900	0.00029	0.00006	5.0E-41	0.00006
1000	0.00026	0.00005	5.0E-41	0.00005

Also, Table 1 contains all the training losses found while we are training the neural network for the xor gate.

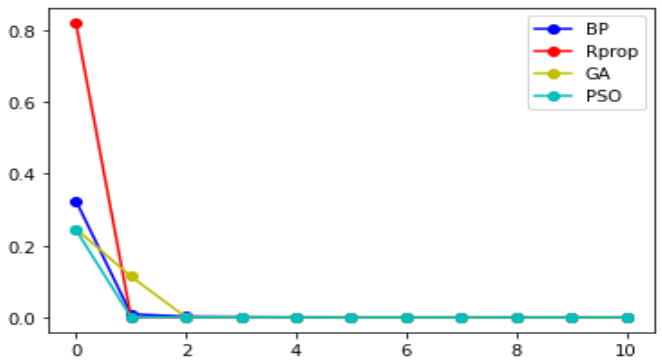

Fig.15. Training Losses of xor gate

By looking at Table.1, we have found that the Evolutionary algorithms Genetic Algorithm and Particle Swarm Optimization algorithm performed better than the rest of the algorithms. The meta-Heuristic algorithm achieved faster convergence in the XOR gate classification problem while BP and Rprop algorithms took more epochs in the process of reducing the neural network loss. Noticeably, we are seeing that GA and PSO in Fig.15 have the lowest error rate in fewer amount epochs.

2) Now, we are going to use the Iris dataset to see the convergence speed of each algorithm that is provided to see and analyze the finding results. Also, closely look at the graphs to then compare them with other algorithms. The lowest the value of loss higher the accuracy will be.

Here, we have again applied Backpropagation (BP) to the standard Iris dataset to see the induced effects of the artificial neural network loss. In Fig.16 we are seeing that the error of the network is slowly decreasing in curve shape which is a good indication of this algorithm.

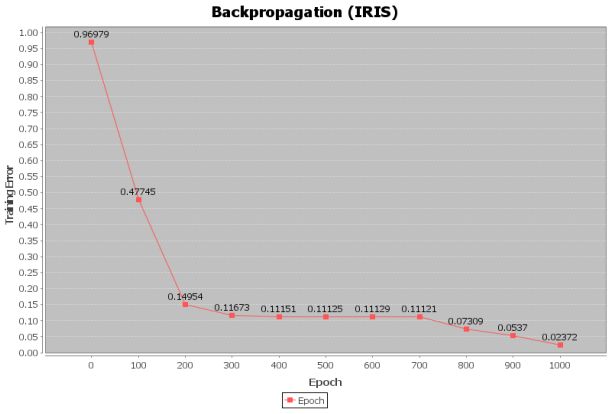

Fig.16 Applied Backpropagation (BP) on the iris dataset

After applying Resilient Propagation (Rprop) in Fig.17 algorithm we have found that the Iris dataset was affectedly learned by the neural network by reducing error quickly. Thus, it has been shown that this algorithm is better than Backpropagation (BP) for this dataset.

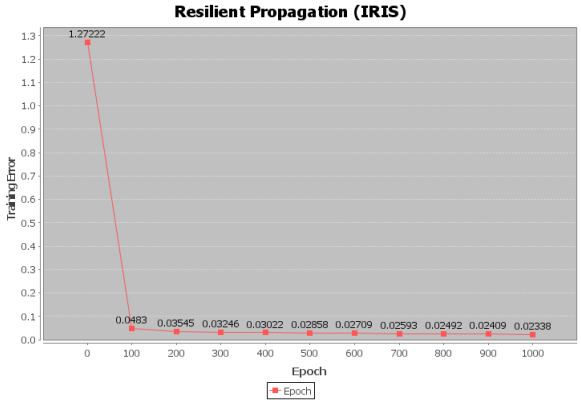

Fig.17 Applied Resilient Propagation (Rprop) on Iris

Besides, we have also used the Particle Swarm Optimization algorithm for faster convergence of neural networks to learn the Iris dataset. The results provided in Fig. 18 show that the algorithm found the optimal solution for this dataset faster than other algorithms. Looking at Table 2, we are seeing that in the first to last epoch, this Particle Swarm Optimization (PSO) bio-inspired algorithm found the optimal solution while the other algorithm was still in the learning process. In Fig. 19, we have shown that PSO is giving more promising results than the other algorithms.

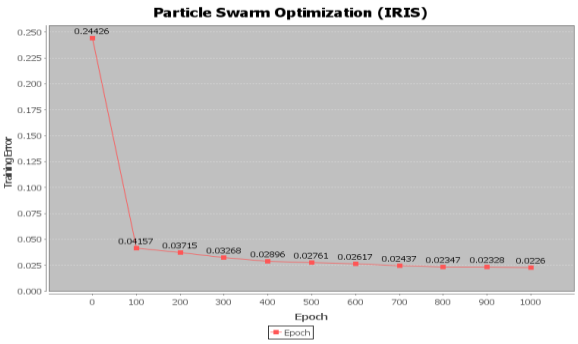

Fig.18 Applied Particle Swarm Optimization on iris dataset

Below is Fig.19, which is containing the losses of neural networks that were applied to the Iris dataset using the Genetic Algorithm (GA).

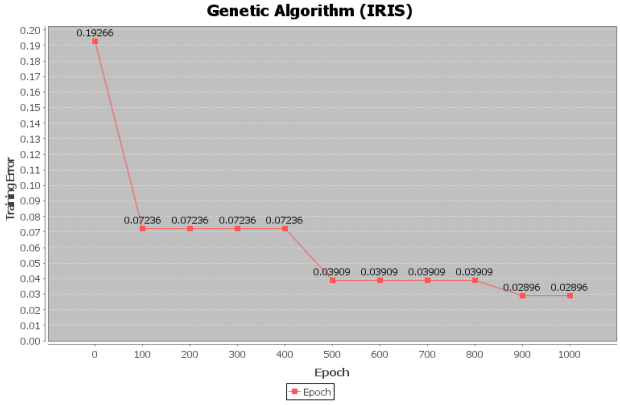

Fig.19 Applied Genetic Algorithm (GA) on iris dataset

Table.2

Epoch	Training Loss			
	BP	Rprop	GA	PSO
0	0.96979	1.27222	0.19266	0.24426
100	0.47745	0.04830	0.07236	0.04157
200	0.14954	0.03545	0.07236	0.03715
300	0.11673	0.03246	0.07236	0.03268
400	0.11151	0.03022	0.07236	0.02896
500	0.11125	0.02858	0.03909	0.02761
600	0.11129	0.02709	0.03909	0.02617
700	0.11121	0.02593	0.03909	0.02437
800	0.07309	0.02492	0.03909	0.02347
900	0.05370	0.02409	0.02896	0.02328
1000	0.02372	0.02338	0.02896	0.02260

Here in Table 2, also includes all of the training losses discovered while training the artificial neural network for the iris dataset for finding the optimal solution.

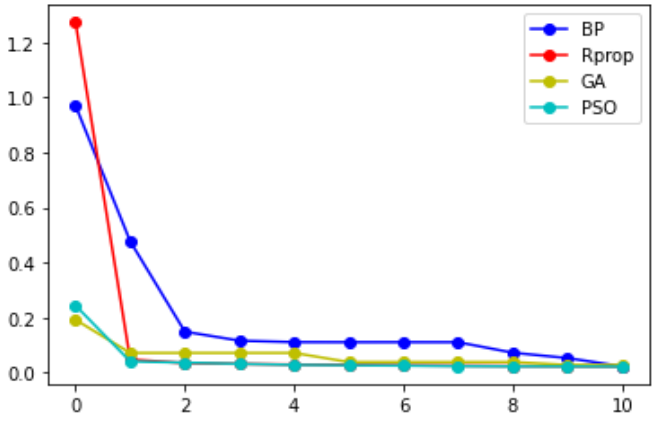

Fig.20. Training Losses of the iris dataset

In the training process of the Artificial Neural Network of Iris dataset Fig.20 shows the promising results that the evolutionary algorithms are better at finding the optimal solution for this dataset. Nevertheless, evolutionary or meta-heuristic algorithms are bound to give sufficiently better results to an optimization problem where the information is not clean and incomplete.

3) At this time, we will utilize the Sonar dataset to examine the convergence speed of each of the algorithms that have been presented, as well as to examine and analyze the findings.

For this Sonar dataset, we have used the Backpropagation (BP) algorithm and in Fig. 21 we are seeing the training error is reducing correspondingly.

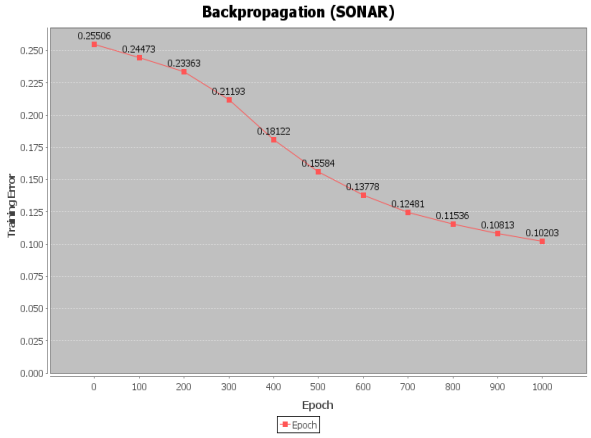

Fig.21. Applied Backpropagation (BP) on sonar dataset

In Fig. 22, the algorithm Resilient Propagation was used to optimize the loss function on the Sonar dataset. Besides, we are seeing that it has reduced the loss much faster than the Backpropagation (BP) algorithm.

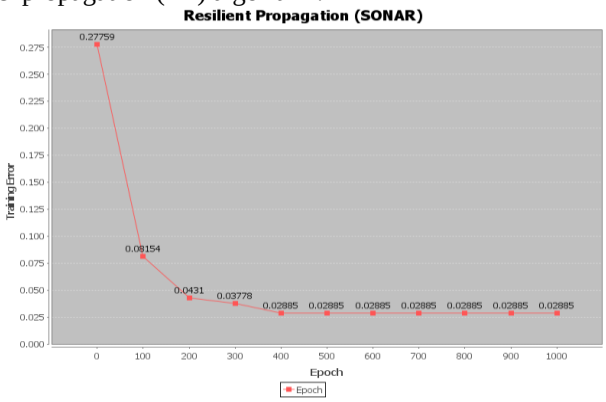

Fig.22. Applied Resilient Propagation (BP) on sonar dataset

Besides, we are also getting efficient results after applying Particle Swarm Optimization for the Sonar dataset. Here, the Meta-Heuristic algorithm Particle Swarm Optimization (PSO) got very prominent results where over the Rprop algorithm we used earlier. In Fig. 23 we are seeing the training loss which was conducted during the training period. Also, this algorithm provided a much quicker training error than the other algorithm.

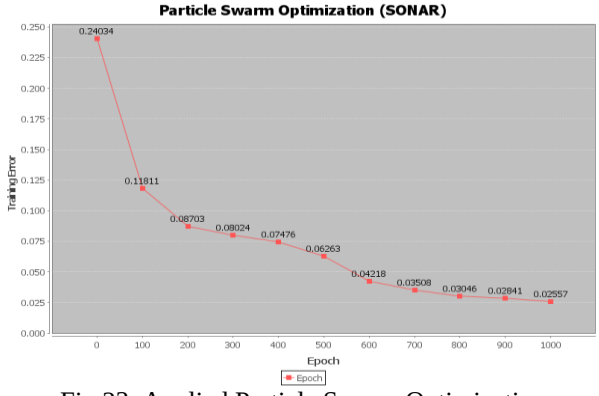

Fig.23. Applied Particle Swarm Optimization on sonar dataset

In Fig.24, we have again used the Genetic Algorithm to see the induced effects after applying the training to the Sonar dataset. Here, we can see the training loss.

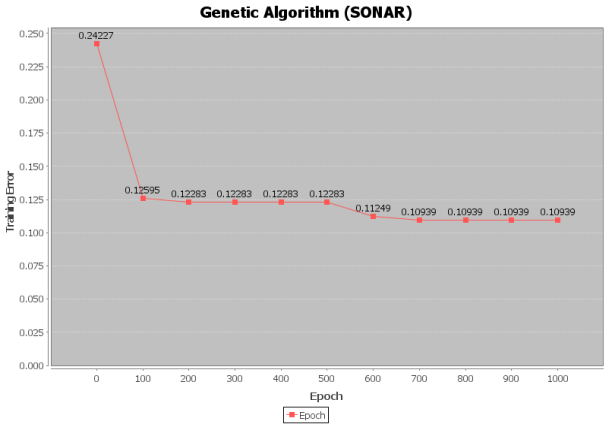

Fig.24. Applied Genetic Algorithm (GA) on sonar dataset

Table. 3

Epoch	Training Loss			
	BP	Rprop	GA	PSO
0	0.25506	0.27759	0.24227	0.24034
100	0.24473	0.08154	0.12595	0.11811
200	0.23363	0.04310	0.12283	0.08703
300	0.21193	0.03778	0.12283	0.08024
400	0.18122	0.02885	0.12283	0.07476
500	0.15584	0.02885	0.12283	0.06263
600	0.13774	0.02885	0.11249	0.04180
700	0.12481	0.02885	0.10939	0.03508
800	0.11536	0.02885	0.10939	0.03046
900	0.10813	0.02885	0.10939	0.02841
1000	0.10203	0.02885	0.10939	0.02557

Nevertheless, all of the training losses discovered during training the artificial neural network for the Sonar dataset for finding the optimal solution are included in Table 3 is shown.

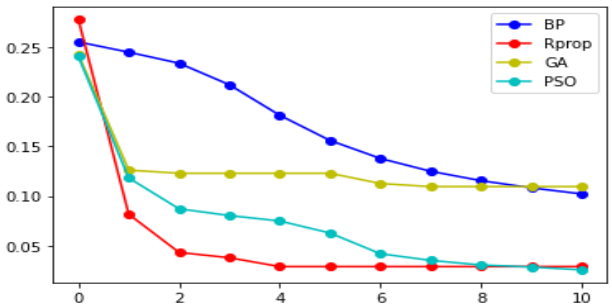

Fig. 25. Training Losses of sonar dataset

Fig.25 displays hopeful results from the training phase of the Artificial Neural Network of Ionosphere dataset, indicating that evolutionary algorithms are better at finding the ideal solution for this dataset.

4) Besides, we will look at the implications of the training error findings of each approach given in the study on the Ionosphere dataset in this section. Each method will attempt to adjust the weight to reduce the neural network's loss function.

In this context, we have applied the Backpropagation (BP) algorithm to the Ionosphere dataset. Below Fig.26 contains all the training errors. This algorithm training loss has a curved shape in the Graph.

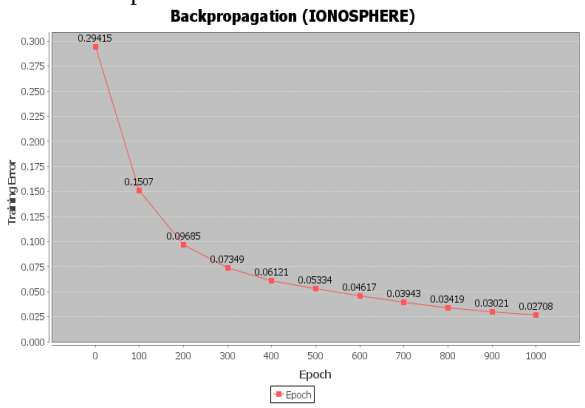

Fig.26. Applied BP on the ionosphere

Moreover, we have again used the Resilient Propagation (Rprop) algorithm on the Ionosphere dataset. In Fig. 27, we have shown the training losses found while training Artificial Neural Network for the Ionosphere dataset.

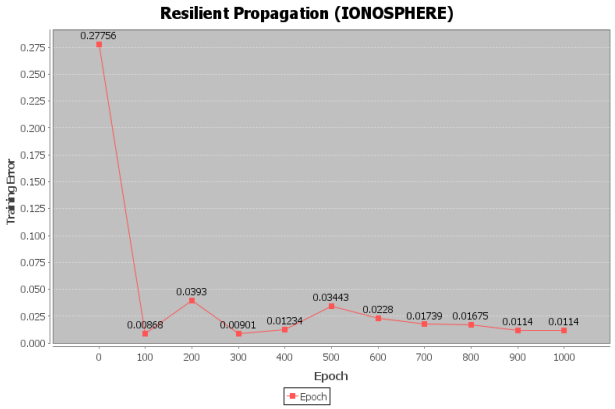

Fig.27. Applied Resilient Propagation (Rprop) on ionosphere dataset

In Fig.28 after applying the Particle Swarm Optimization algorithm for the Ionosphere dataset the training loss graph has been shown.

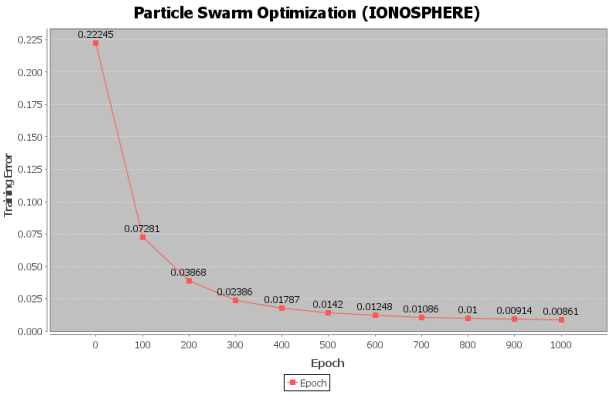

Fig.28. Applied Particle Swarm Optimization on ionosphere dataset

The training loss graph for the Ionosphere dataset is given in Fig.28 after using the Genetic Algorithm for Neural Network. The Graph shows that the loss of the training got stuck for some time in epoch 100-700 and again regain reducing the training losses.

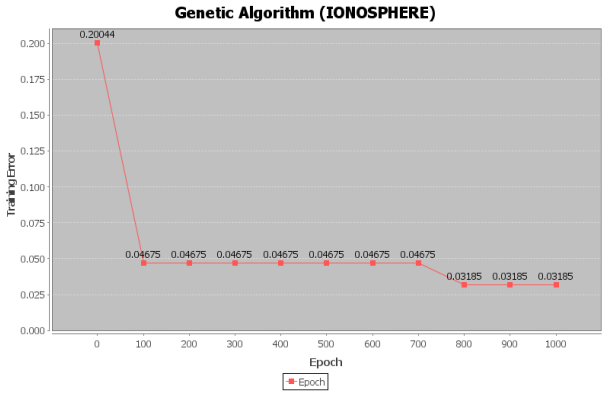

Fig.29. Applied Genetic Algorithm (GA) on ionosphere dataset

Table. 4

Epoch	Training Loss			
	BP	Rprop	GA	PSO
0	0.29415	0.27756	0.20044	0.22245
100	0.15070	0.00868	0.04675	0.07281
200	0.09685	0.03930	0.04675	0.03868
300	0.07349	0.00901	0.04675	0.02386
400	0.06121	0.01234	0.04675	0.01787
500	0.05336	0.03443	0.04675	0.01420
600	0.04617	0.02280	0.04675	0.01248
700	0.03943	0.01739	0.04675	0.01086
800	0.03419	0.01675	0.03185	0.01000
900	0.03021	0.01140	0.03185	0.00914
1000	0.02708	0.01140	0.03185	0.00861

In Table. 4, there are a lot of training losses we are seeing but the table shows that Particle Swarm Optimization is better at reducing the training losses. Hence, Particle Swarm Optimization provided the best optimal solution in this context.

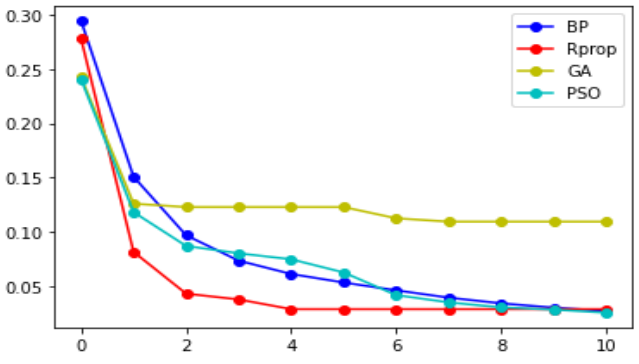

Fig.30. Training Losses of Ionosphere dataset

The results are good when applied to bio-inspired algorithms like Particle Swarm Optimization. But Genetic algorithm did not produce good results on the Ionosphere dataset.

Here, we have used four datasets that include the XOR gate, Iris dataset, Sonar dataset, and Ionosphere dataset. Particle Swarm Optimization was better in three datasets named Iris, Sonar, and Ionosphere. Besides, the Genetic Algorithm has gained a lot of training loss in the XOR gate. Both algorithms are bio-inspired whereas the other algorithms were application-specific like Backpropagation (BP) and Resilient Propagation (Rprop). Hence, the Particle Swarm Optimization and also Genetic Algorithm were not application-specific but this algorithm produced good results in optimizing the weights of the Neural Network. In the research paper, after getting training loss results, we find that the evolutionary bio-inspired algorithm Particle Swarm Optimization was better and it is also a meta-heuristic algorithm.

X. Discussion

In this case study research paper, we have discussed Deep Neural Network Optimization with meta-heuristic and heuristic algorithms. We have also used bio-inspired algorithms Genetic algorithm which works by the course of natural selection and also Particle Swarm Optimization. Besides, there were also application-specific algorithms like Backpropagation and Resilient Propagation respectively. We have compared four datasets applying these application-specific and Bio-Inspired algorithms by looking at the convergence meaning reducing the loss of the neural network. The Particle Swarm Optimization algorithm produced the lowest training loss meaning it has the highest probability of getting the highest accuracy respectively. Lower training loss means higher accuracy. Therefore, we have seen that bio-inspired algorithm particle swarm optimization is producing the optimal solution faster and more accurately. In this paper, we have included all the documents that relate to Meta-Heuristic and Heuristic algorithms.

XI. Conclusion

The motivation of this research is to highlight bio-inspired and application-specific algorithm performance. To optimize a loss function these meta-heuristic algorithms like particle swarm optimization can be used because of their computational efficiency also particle swarm optimization and genetic algorithms are evolutionary algorithm which is inspired by living organisms that reflects natural behavior. Neural Network is a bio-inspired algorithm that is collected from the behavior of nature and now it is used by almost every field. Therefore it is crucial to use an optimization algorithm that is reversed engineered from nature and also has computational efficiency. If we can utilize the nature-inspired algorithm more, we will have a more robust structured algorithm that will provide a decent amount of performance. We found that the meta-heuristic algorithm PSO was better in this case study. We used the Genetic algorithm and the Particle Swarm Optimization algorithm as bio-inspired algorithms and on the other hand, we have used the application-specific algorithms Backpropagation and Resilient Propagation to find the optimal solution for the dataset that we have compared here. Nevertheless, we have also described how this algorithm works with Artificial Neural Network Optimization techniques. In conclusion, we have come to a solution that the bio-inspired algorithm Particle Swarm Optimization was better in all of the case studies.

XII. REFERENCES

[1] Abhijit Suresha, K.V Harisha, N. Radhikaa, "Particle swarm optimization over back propagation neural network for length of stay prediction", 2015.

[2] M. Carvalho and T.B. Ludermir, "Hybrid Training of Feed-Forward Neural Networks with Particle Swarm Optimization", Springer-Verlag Berlin Heidalberg, 2006.

[3] Venu G. Gudise and Ganesh K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training", 2003.

[4] Xiao-Lin Li, Roger Serra, Olivier Julien. "Effects of the Particle Swarm Optimization parameters for structural dynamic monitoring of cantilever beam". Surveillance, Vishno and AVE conferences, INSA-Lyon, Université de Lyon, Jul 2019, Lyon, France.

[5] Venu G. Gudise and Ganesh K. Venayagamoorthy, "Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks", 05 June 2003, DOI: 10.1109/SIS.2003.1202255

[6] Martin Riedmiller, Heinrich Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", 06 August 2002, DOI: 10.1109/ICNN.1993.298623

[7] Hong Cai, Yanda Li, "Fuzzy neural network optimization method based on Hopfield networks", January 1998.

[8] Seba Susan, Rohit Ranjan, Udyant Taluja, Shivang Rai, Pranav Agarwal "Global-best optimization of ANN trained by PSO using the non-extensive cross-entropy with Gaussian gain", Soft Computing 2020

[9] Jiri Stastny, Vladislav Skorpil, "Designing Neural Networks using Genetic Algorithms." August 2007.

[10] Santosha Rathod, Amit Saha, Kanchan Sinha, "Particle Swarm Optimization and its applications in agricultural research" April 2020.

[11] A. Mosca and G. D. Magoulas, "Adapting resilient propagation for deep learning," CoRR, vol. abs/1509.04612, 2015.

[12] W. D, "Stacked generalization," Neural Networks, vol. 5, pp. 241–259, 1992.

[13] J. Heaton, "Encog java and dotnet neural network framework", Heaton Research, Inc., Retrieved on July 20 (2010) 2010.

[14] Sadegh Mirshekarian, Dusan Sormaz, "Machine Learning Approaches to Learning Heuristics for Combinatorial Optimization Problems",https://doi.org/10.1016/j.promfg.2018.10.019.

[15] Fadlallah, S.O., Anderson, T.N. & Nates, R.J. "Artificial Neural Network–Particle Swarm Optimization (ANN-PSO) Approach for Behaviour Prediction and Structural Optimization of Lightweight Sandwich Composite Heliostats". Arab J Sci Eng 46, 12721–12742 (2021). https://doi.org/10.1007/s13369-021-06126-0

[16] Agnes Lydia, Sagayaraj Francis, "A Survey of Optimization Techniques for Deep Learning Networks", May 2019, DOI: 10.35291/2454-9150.2019.0100

[17] Thomas Ragg, Heinrich Braun, Heiko L, "A Comparative Study of Neural Network Optimization Techniques", February 1997, DOI: 10.1007/978-3-7091-6492-1_75

[18] Trần Ngọc Hà, Nguyễn Thanh Thủy, "The backpropagation neural network for modelling", March 2016, DOI: 10.15625/1813-9663/14/1/7879

[19] Yang Meng, Hosahalli S. Ramaswamy, "Neural Networks and Genetic Algorithms", December 2008, DOI:10.1201/9781420061420.ch10

[20] Gursel Serpen, Joel Corra, "Training Simultaneous Recurrent Neural Network with Resilient Propagation for Static Optimization" June 2022, International Journal of Neural Systems 12(3-4):203-18, DOI: 10.1142/S0129065702001199.

[21] Riedmiller, M. and Braun, H., 1993. "A direct adaptive method for faster backpropagation learning": The RPROP algorithm. In Neural Networks, 1993., IEEE International Conference on (pp. 586-591). IEEE

[22] Garro, Beatriz A, Vázquez, Roberto, "Designing Artificial Neural Networks Using Particle Swarm Optimization Algorithms", 2015/06/29 https://doi.org/10.1155/2015/369298.

[23] Cao Li; Xiaoyu Liu , "An improved PSO-BP neural network and its application to earthquake prediction" 28-30 May 2016, DOI: 10.1109/CCDC.2016.7531576

Partha Sutradhar[1] is a graduate student from American International University - Bangladesh. Currently, He is working as a Full-Time Software Engineer in a R & D Sector. He is also looking for a post-graduation degree in Software Engineering. He is specialized in System Architecture, Deep Learning and Embedded System. His region of interest includes Artificial Intelligence, Computer Vision, Automation, IoT, Neural Network Optimization, Image Processing and, Embedded Systems.

Victor Stany Rozario[2] completed B.Sc. in Computer Science & Engineering and M.Sc. in Computer Science from American International University-Bangladesh, Dhaka Bangladesh. Currently he is working as an Assistant Professor in the Department of Computer Science under the Faculty of Science and Technology, AIUB. His current research interest include Data Science, Data Mining, Intelligent Systems, Machine Learning, Deep learning, Web Mining and Human Computer Interaction.