International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Innovative Approaches for Machine-Driven Software Testing Using Neural Networks

*Amit Bhanushali[1], Karan Gupta[2], Manvendra Sharma[3]

Submitted: 17/09/2023 **Revised**: 17/11/2023 **Accepted**: 28/11/2023

Abstract: The primary objective of software testing is to create fresh sets of test cases that accurately represent software defects. Once these test cases are created, Test Oracle gives a procedure for how the program should respond to each test. Reducing the time and effort spent on finding and fixing bugs, while maintaining as much data as possible, is possible via careful evaluation of the application and the selection of an effective method for optimizing and prioritizing test cases. The suggested neural network outperforms ANN in terms of accurate output misclassification, at least on the synthetic dataset.

Keywords: *Neural Network, Software Testing, Test Oracle, Synthetic Dataset, Software Quality Optimization.*

1. Introduction

In the Software Development Life Cycle (SDLC), software testing is the process of running and analyzing the application to find bugs. In most cases, efforts are directed at gauging the effectiveness of some aspect of the program or system's ability to determine whether or not specified criteria have been fulfilled. The software system is quite similar to other sorts of physical systems in which inputs are taken in and outputs are generated. However, the program does not undergo any alterations of any kind. [1] Unless improvements or adjustments are made to accommodate user requirements, stability is maintained. It follows that design flaws in software may lie dormant until they are activated, even after the product has been made public. All of these estimates need to be double-checked and put through further tests. However, conducting thorough tests might be difficult. If an early check fails, the code is released, and the program passes the test case even if that was not the goal. Pre-debugging test cases may no longer be relied upon due to this behaviour.[2]

Various statistical programs have been undergoing testing by the software engineering community. The basic premise was that how software was used directly affected its dependability.[3] The possibility of encountering this failure has defined the real significance of its failure. Therefore, in addition to the many test cases that had to be run to statistically simulate software use, this program had also been evaluated in accordance with the model emphasizing crucial utilization.[4] Manually producing test case outputs and then comparing them to the AUT's outputs is possible, but it takes a lot of effort and is prone to mistake. In order to automatically create output, statistical software testing required the final findings of the automated test oracle. [5]

Once upon a time, testing was considered to constitute its own stage in the software development life cycle. Numerous scholars in the past have blamed a lack of testing for software failure.[6] It has been thoroughly tested at several tiers, including regression, system, module, integration, acceptance, and unit testing. There are two main schools of thought when it comes to testing at these tiers: white-box and black-box. In white-box testing, code is manually executed by a developer.[7]

One possible interpretation of the decision-making network is that optimization is a crucial quantitative tool. The selections to be made here will optimize some combination of stated goals over a finite set of circumstances. [8]When considering the usefulness of optimization, it is necessary to have a method for computing solutions to medium- and large-scale optimization problems using mathematical models of computers.[9] Researchers now have a plethora of bio-inspired methodologies and heuristics to choose from when attempting to find optimal solutions to a wide range of challenges.[10] For large-scale combinatorial optimization issues, such methods may not necessarily provide the best possible outcome. However, within the bounds of a reasonable amount of computing, similar methods may offer adequate answers to the other applicable issues.[11]

Many different types of issues are being solved using this method. The swarm is currently understood to be a collection of autonomous agents that coordinates its actions in accordance with a predetermined set of rules.

[1] *Independent Researcher, Quality Assurance Manager, West Virginia University, WV, USA*
ORCID ID: 0009-0005-3358-1299
[2] *Senior Data Scientist, SunPower Corporation, TX, USA*
ORCID ID : 0009-0007-6225-4056
[3] *Embedded Software Development Enineer, Amazon, USA*
ORCID ID : 0009-0008-2426-5325
***** *Corresponding Author Email: akbhanushali@mail.wvu.edu*

[12]Many different models of swarm intelligence have been created and studied in depth; some of these models are widely employed. They are the Particle Swarm Optimization (PSO), the Stochastic Diffusion Search (SDS), the swarming of honeybees, and the ant colony optimization (ACO). [13-14] All of these algorithms have shown their problem-solving prowess in a variety of contexts, including some very challenging ones. An method based on a combination of the PSO-SDS and the Harmony Search (HS)-SDS has been suggested for this study.[15]

Synthetic data refers to information that has been created in a lab. This is generated algorithmically and serves as a proxy for real-world production or operational information. [16]This is put to use in the processes of verifying mathematical models and instructing machine learning algorithms. For special purposes, it is possible to create synthetic data under circumstances that are not present in the original or genuine data. [17-18]Since synthetic data are often utilized for simulation or as a theoretical value or condition, this is of great help when developing any system. If the findings turn out to be less than ideal, we'll at least have the option of a simple fix. It is possible to develop synthetic data to stand in for real data, and thus allows for benchmarking purposes. Synthetic data also has the added benefit of shielding real information from prying eyes.[19]

2. Literature Review

He et al. (2019) [20] The Adopted Autoregressive-System Identification (AR-SI) was the basis for an oracle proposal. The black-box physical systems might be managed using this method. It was modified for use in tracking down the flawed black-box Control-Cyber-Physical System (CPS). It then presents a technique that produces the traces for the control-CPS debugging, and this is utilized as an identifying result when evaluating the Control-CPS's behavior. Several in-depth comparisons with real-world or simulated issues were performed on traditional Control-CPSs. This shown that the method had better accuracy, recall, latency, and false positive/negative rates than a human oracle method in the SFL. Another real-world problem in Control-CPS for home use was uncovered with the use of this method.

Jahan et al. (2019) [21] It was proposed to look into whether or not ANN-based techniques may enhance the method of version-specific prioritizing of test cases. In order to identify crucial flaws, this method use a mix of different test cases and the ANN. Priorities were recommended to be taught on three different elements. Two software programs were used to conduct an empirical evaluation of the strategy. Precision and recall, accuracy, and defect detection rate were also studied as indicators of efficacy. The results validated the practicability and efficacy of the approach.

Lachmann (2018). [22] Another method of test case prioritizing using supervised machine learning for system testing was suggested. After that, it analyzes natural language artifacts and meta-data in preparation for ranking additional manually conducted test cases. In addition to the novel ensemble learning strategy, various machine learning algorithms are used for the job at hand. Ensemble learning techniques are used in addition to other machine learning techniques for this job. Three more datasets from the automobile sector were used to further assess this method since they are representative of real-world datasets. The findings are assessed for their ability to detect problems. The results demonstrate that using these machine learning methods might enhance black-box testing.

Souza et al. (2022)[23] Another automated method for generating tests was developed which made use of Hill Climbing to get more robust mutation. This is gradual and has as its primary goal the eradication of mutations and the prevention of their further spread. Some empirical findings on the work's efficacy and price are also reported. This is calculated using a dataset of 18C software. In most cases, the tested programs' strong mutation scores were increased by the method compared to the average score from random testing (19,02%). On average, the spread of mutations was 7% slower using the older methods. Their findings demonstrated more efficiency than previous approaches.

3. Research Methodology

3.1 Weight Optimized

The Algorithm of Stochastic Diffusion Search Bishop describes SDS, a unique matching algorithm, as a population-based algorithm that leverages direct communication patterns like cooperative transport seen in social insects to assess the efficacy of a search based on an optimization hypothesis. The swarm agents that preserve the optimum and its hypothesis have been subjected to independent evaluation, and the method has been shown to be applicable to a wide range of search problems requiring optimization (Al-Rifaie and Bishop, 2013). Algorithm for Simplifying Difficult Sequences:

$$Initialising\ agents()$$

$$While\ (stopping\ condition\ is\ not\ met)$$

$$Testing\ hypothesis()$$

$$Diffusion\ hypothesis()$$

$$End$$

There are many steps to the SDS algorithm, and they include:

Initialization phase: Here, the agents will choose at random to determine which hypothesis to pursue, and the resulting subsets will be utilized in the subsequent search. After the setup phase is finished, the test phase begins by randomly assigning an objective function-based hypothesis to each agent. The agent is set to active if the hypothesis assessment yields a positive result, and to inactive otherwise. As a result, in the end, each agent takes on one Boolean attribute. Phase of diffusion: During the dissemination stage, the agents will share their knowledge of the hypothesis with one another. The key goal here was to guarantee that the proactive agents may spread the hypothesis to the passive ones. Various forms of interaction, or recruiting tactics, have been used. There are five types of recruiting strategies: active, passive, context-aware, context-free, and dual.

Artificial Neural Network (ANN) Classifier: The ANN is a complicated adaptive system that may modify its internal structure in response to the data it has been fed. Weight and connection adjustments may also accomplish this. This is how the neural network works:

· Nodes in the hidden, input, and output layers together constitute a structure.

• An algorithm is used to facilitate learning.

ALGORITHM

Input: Dataset D, the learning rate, the network

Output: A trained neurla network

Step 1: Receive Input

Step 2: Weight Input

Every input that is sent to the network has to be weighted i.e. multiplied using a random value falling between -1 and +1.

Step 3: Sum the weighted input

Step 4: Generate the output the actual network output will be produced by passing the sum through activation function.

Artificial neural networks (ANNs) were proposed, with its architecture derived from data that included real-world inputs, outputs, neurons, and connections between them. An intricate ANN framework may be designed using the lessons learned from solving similar problems in the past. Time complexity measurements made utilizing the SDS's system and dimensions revealed different efficient strategies compared to those used for other optimization challenges.

3.2 A Hybrid Optimization

An example of a population-based heuristic search technique, Particle Swarm Optimization (PSO) is used for global optimization problems. John Kennedy and Eberhart came up with this in 1995. A fitness function will be run on each particle, and its future fitness value will be assessed. lbest represents the best population achieved by any particle in its immediate vicinity since the PSO began. Every iteration's location and speed will be revised in accordance with the equation:

$$V_{pd}^{k+1} = wV_{pd}^{k} + c_1\, r_1\, (pbest^k - X_{pd}^{k}) + c_2 r_2 (gbest^k - X_{id}^{k})$$

$$X_{pd}^{k+1} = X_{id}^{k} + V_{pd}^{k+1}$$

Where is the inertia weight (between 0.2 and 0.9), k is the iteration number, k Vpd is the velocity in the pth particle's dth dimension, k Xid is the actual position of the pth particle in the dth dimension, pbest and gbest are the particle's best and worst memories, c1 and c2 are the cognitive and social factors, and r1 and r2 are uniform.

Harmony Search (HS) Algorithm:Based on how a musician might search for greater harmony on the fly during a performance (particularly in jazz improvisation), Geem (2010) developed the HS algorithm in 2001. This algorithm is able to identify a unique and aesthetically pleasing harmony via some kind of aesthetic decision-making process. Below is a breakdown of the optimization phases in the HS method :

Step 1: Parameter Initiation - The Optimization Issue Detected Is Depicted By Equation:

Minimize f(x)

Subject To

$$x_j \in X_j \;\; j = 1,2,\dots,N$$

Step 2: Harmony Memory's Initialization and Assessment: This is a Unique Random Initial Population Produced Using Equation:

$$x_{i,j}^{0} = x_j^{min} + r_j (x_j^{max} - x_j^{min})$$

Step 3: The new harmony vector was developed based on the following three rules: a random selection, pitch modification, and consideration of memory. “A design variable's value was picked from the set of HM values that did not exceed HMCR.”

Step 4: The term "Harmony Memory" is used to describe a New Harmony vector that was found by replacing the worst existing vector with a vector that violated some criteria.

Step 5: The stopping criteria (also referred to as the maximal improvisation) is checked, and the HS is

terminated if it is attained. If this is not the case, steps 3 and 4 are often repeated.

Each PSO particle in a hybrid algorithm will store information about its past and future locations as well as its velocity (representing its best-ever finish) and its current position. However, each SDS agent operates on a supposition and a standing. When first activated, the SDS gives each agent a random set of weights and biases.

```
Initialise pAgents
While (stopping condition is not met)
For all pAgents
        Evaluate fitness value of each particle
                If (evaluation counter MOD n=0)
                //START SDS
                //TEST PHASE
                   for ag=1 to No_of_pAgents
                   r_ag=pick-random-pAgent ()
                        if (ag.pbestFitness () <= r_ag.pbestFitness ())
                           ag.setActivity (true)
                        else
                           ag.setActivity (false)
                        end if
                   end for
                //DIFFUSION PHASE
                   for ag=1 to No_of_pAgents
                        if (ag.activity () == false)
                        r_ag=pick-random-pAgent ()
                          if (r_ag.activity () == true)
                             ag.setHypo (r_ag.getHypo ())*
                           else
                             ag.setHypo (randomHypo ())
                           end if
                        end if
                    end for
                //End SDS
If (current.fitness<pbest)
        pbest=current fitness
           If (pbest<global (or local) best)
                global (or local) best=pbest
                Update particle velocity
                Update particle position
        End
End
```

Fig 1: Hybrid PSO-SDS Algorithm

High-Speed using SDS Proposed Algorithm To use the HS technique, it is customary to use Harmony Memory (HM) to keep track of a subset of the feasible vectors in a feasible space. The HM population is seeded at random and then tested for viability. Using a value from one of the solutions in the harmony memory (chosen uniformly at random), the probability HMCR uses that solution's value and then modifies it very little using the probability PAR. In order to acquire optimum values, the answers are updated and then used as the starting population of SDS. SDS-HS is a suggested method for.

```
Initialize the harmony memory with HMS randomly generated solutions
(encode ANN architecture, weights), initialize Agents
Repeat
        create a new solution in the following way
        for all decision variables do
                with probability HMCR use a value of one of the solutions in the
                harmony memory (selected uniformly at random) and
                additionally change this values lightly with probability PAR
                otherwise (with probability1-HMCR) use a random value for
                this decision variable
        end for
if the new solution is better than the worst solution in the harmony memory
then
        replace the worst solution by the new one
end if
Testing Hypothesis ()
Diffusion Hypothesis ()
Obtain solution of SDS
until the maximum number of iterations has been reached
return the best solution in the harmony memory with SDS solution
```

Fig 2: Pseudocode for Hybrid Harmony Search-SDS Algorithm

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). This applies to papers in data storage. For example, write “15 Gb/cm2 (100 Gb/in2).” An exception is when English units are used as identifiers in trade, such as “3½-in disk drive.” Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength H is A/m. However, if you wish to use units of T, either refer to magnetic flux density B or magnetic field strength symbolized as µ0H. Use the center dot to separate compound units, e.g., “A·m2.”

3.3 Synthetic Dataset for Machine Driven Software Test Oracle

This lets us model user and adversary behaviour accurately. The information is fed into a learning algorithm that fine-tunes the fraud detection system for usage in a given setting.

Below, we list some of synthetic data's inherent benefits:

- Once the synthetic environment is set up, it can rapidly and cheaply produce the necessary data;
- Unlike real data, synthetic data has accurate labels, which can be a significant cost savings when training a model;
- The synthetic environment can be tweaked to optimize model training; and
- Synthetic can be used to substitute real data segments and sensitive data.

This way of using synthetic data was proposed primarily for the applications of computer vision and, more particularly, in the detection of objects in which the synthetic environment will be a 3D model for the object that helps in navigating environments using certain visual information.

4. Results

Parameters for the neural network were already specified before

training began, since it was fed the processed data in lieu of a traditional training dataset. During network training, we presented the whole dataset in a single epoch, and we also stated the exact number of epochs utilized for training. Furthermore, the back-propagation training method terminated when either the maximum number of epochs was reached or the target error rate was reached. The network was then employed as an oracle to provide accurate predictions for future regression testing. Training utilizes 80% of the data, whereas testing uses 20%. In our studies, we used 10-way cross-validation. Misclassification as a percentage of accurate output, misclassification as a percentage of incorrect output, and convergence are all tabulated below. The fault detection threshold was set at 0.6.

Number of erroneously labelled features divided by total features is the misclassification rate (in percent). Metaheuristic algorithms use the term "convergence" to describe the rate at which they find optimum (or good enough) solutions.

Table1: Rate of incorrect output due to misclassification

Number of injected faults	Artificial Neural Network	Proposed Neural Network
1	7.5	3.13
2	8.13	3.13
3	8.75	5.63
4	9.38	6.25
5	10.63	6.88

The proposed neural network outperforms ANN at iterations 1, 2, 3, 4, and 5 with regard to the percentage of incorrect output classifications by 82.22%, 88.81%, 43.44%, 40.11%, and 42.83%, respectively.

Table2: Incorrect output misclassification as a percent

Number of injected faults	Artificial Neural Network	Proposed Neural Network
1	5.83	6.67
2	8.33	7.5
3	11.67	8.33
4	12.5	9.17
5	13.33	9.17

The proposed neural network outperforms ANN at iterations 1, 2, 3, 4, and 5 with regard to the percentage of incorrect output classifications reduced by 13.44%, 10.5%, 33.4%, 30.73%, and 36.97%, respectively. At iteration number 550, ANN converged at a value of 0.36. At the 500th iteration, the suggested ANN converged to an accuracy of 0.23.

The findings showed that the proposed neural network outperformed the ANN at iterations 1, 2, 3, 4, and 5 by 82.22 percent, 88.81 percent, 43.4 percent, 40.1 percent, and 42.83 percent for misclassifying the correct output. The proposed neural network outperforms ANN at iterations 1, 2, 3, 4, and 5 by 13.44%, 10.5%, 33.4%, 30.73%, and 36.97%, respectively, in terms of incorrect output categorization.

The usage of software oracles as a supplementary tool for test automation has become widespread. The popular open-source tools Selenium and Katalon Studio do not enable test oracles, despite their widespread usage. Methods for implementing test oracles were suggested in this paper. Matlab and a modified version of the WEKA program were used for the simulations. WEKA is a cost-free open-source software application. In this work, we show and train the neural network with errors that are inserted dynamically. The misclassification rate was calculated by comparing the testing data with the injected fault whenever new faults were introduced (both singly and in groups). The introduced problems are detailed in Table.

Table 3: Introductory Errors

Originalstatement	Injectedfaults
To determine whether either of two options is true, the OR operator is used.	Taken out so that just one Operator is visible

The Use of an OR Operator	Swapped out OR Operator with AND Operator
In excess of the Operator Used	Greater Than Operator Substituted
In excess of the Operator Used	Equal Operator has been substituted.
In excess of the Operator Used	Less-Than operator has been substituted.
The AND operator is used to verify both conditions.	Substituted OR for AND operator
Discourse Shift	Quantity modified

Misclassified correct output, misclassified incorrect output, and convergence data are shown in tables and.

Table 4: The proposed Harmony Search-SDS neural network's misclassification error as a percentage of its accurate predictions

Number of injected faults	Proposed PSO-SDS Neural Network	Proposed Harmony Search- SDS Neural Network
1	3.03	2.91
2	3.02	2.92
3	5.4	5.22
4	5.99	5.75
5	6.63	6.37

Misclassification of the right output is improved by 4.04%, 3.37%, 3.39%, 4.09%, and 4% for the Proposed Harmony Search-SDS Neural Network compared to the Proposed PSO-SDS Neural Network at iterations 1, 2, 3, 4, and 5.

Table 5: The percentage of incorrect predictions made by the proposed Harmony Search-SDS neural network.

Number of injected faults	Proposed Harmony Search-SDS Neural Network	Proposed PSO-SDS Neural Network
1	5.93	6.12
2	6.54	6.8
3	7.47	7.73
4	8.08	8.36
5	8.14	8.43

"Misclassification of incorrect output is improved by 3.15%, 3.9%, 3.42%, 3.41%, and 3.5% for iterations 1, 2, 3, 4, and 5 for the Proposed Harmony Search-SDS Neural Network compared to the Proposed PSO-SDS Neural Network. At iteration 500, with a value of 0.23, the proposed Harmony Search-SDS Neural Network converged." Iteration 550, with a value of 0.36, also marked the point at which the planned PSO-SDS Neural Network converged.

The experimental results demonstrated that the Proposed Harmony Search-SDS Neural Network outperforms the PSO-SDS Neural Network by approximately 4.04% at iteration 1, approximately 3.37% at iteration 2, approximately 3.39% at iteration 3, approximately 4.09% at iteration 4, and finally by 4% at iteration 5. However, in the case of real datasets, there may be misclassification of the right output.

5- Synthetic data is information that has been created in a lab rather than gathered from the actual world. Oftentimes, machine learning models are trained using synthetic data, which is constructed algorithmically and utilized as a stand-in for test datasets of production or operational data. Synthetic data may be used to generate real-world results and establish norms. Synthetic information may also be used to conceal the origin of real information. Misclassified valid output and incorrect output are shown in separate tables. convergence, incorrect output misclassification, and incorrect output misclassification outcomes.

Table 6: Synthetic Data Percentage of Accurate Results for Misclassification

Number of injected faults-Eligibility output	Proposed ANN	ANN
1	3.32	7.92
2	3.3	8.43
3	5.83	9.24
4	6.61	9.87
5	7.2	11.1

The suggested NN outperforms the ANN in terms of accurate output misclassification at iterations 1, 2, 3, 4, and 5 by 81.85%, 87.46%, 45.25 %, 39.56 %, and 42.62 %, respectively.

Table 7: Synthetic Data's Percentage of Incorrect Classification Output

Number of injected faults-Eligibility output	Proposed ANN	ANN
1	6.92	6.15
2	7.85	8.73
3	8.65	12.34
4	9.62	12.92
5	9.53	13.81

The suggested NN outperforms the ANN in terms of incorrect output misclassification by 11.78%, 10.61%, 35.15%, 29.28%, and 36.67% on iterations 1, 2, 3, 4, and 5, respectively. It took 650 iterations for ANN to converge, at which point the value was 0.37. 500th iteration saw convergence for the proposed ANN, with a final value of 0.24.

Table 8: Accuracy of the proposed Harmony Search-SDS neural network in correctly classifying synthetic data, expressed as a percentage

Output based on the number of errors introduced	ProposedPSO-SDSNeural Network	ProposedHarmony Search-SDSNeuralNetwork
1	3.2	3.09
2	3.18	3.05
3	5.62	5.42
4	6.36	6.11
5	6.96	6.71

Proposed Harmony Search-SDS Neural Network outperforms Proposed PSO-SDS Neural Network in terms of accurate output misclassification at iterations 1, 2, 3, 4, and 5 by 3.5%, 4.17 %, 3.62%, 4%, and 3.66%, respectively.

Table 9: Incorrect output percentage of a proposed neural network using the harmony search-synthetic data set

Numberofinjectedfaults-Eligibility output	ProposedPSO-SDSNeural Network	ProposedHarmonySearch-SDSNeural Network
1	6.67	6.46
2	7.59	7.35
3	8.35	8.08
4	9.29	8.97
5	9.2	8.89

demonstrates that for each iteration, the Proposed Harmony Search-SDS Neural Network outperforms the Proposed PSO-SDS Neural Network by a margin of 3.19%, 3.2%, 3.29%, 3.5%, and 3.43% with regard to the misclassification of the incorrect output.

At the 500th iteration, the Proposed Harmony Search-SDS Neural Network converged at a value of 0.23. Iteration 650, with a value of 0.35, also marked the point at which the Proposed PSO-SDS Neural Network converged.

The experimental findings further show that for Synthetic Data, the proposed NN outperforms the ANN at iterations 1, 2, 3, 4, and 5 with respect to the percentage of correctly classified inputs by 81.85%, 87.46%, 45.25 %, 39.56 %, and 42.62 %, respectively.

5. Discussion

The SDS was a population-based multi-agent global search, while the optimization method was a simple-agent-interaction-based distributed model of computing. In its most effective version, the ANN functioned as a device for fixing a number of decision modelling issues. It did not rely on guesses since it was non-parametric. Therefore, the evidence supported the conclusion, which is always a plus.

In software, there are several opportunities for mistakes to occur. The selection of test cases for the goal of finding these kinds of mistakes was accomplished with the help of a regression test suite. This increased the effectiveness of the test case and its goals, which reduced the resources needed for keeping service-oriented business applications up and running. Here, the HS-SDS algorithm is presented with the hybrid PSO-SDS. This method may successfully compensate for the drawbacks of PSOs like partial optimize, which include the need to control the algorithm's pace and course. This PSO-SDS method may also enhance the performance scalability, solution diversity, and efficiency of the solution's process.

In terms of correct misclassification, the Proposed Harmony Search-SDS Neural Network has outperformed the Proposed PSO-SDS Neural Network. 0.2 0.25 0.3 0.35 0.4 0.45 0.5 0.55 0.6 0.65 Fitness Value Neural Network Iteration Proposed PSO-SDS Output on Eligibility Eligibility output 129 from the proposed Harmony Search-SDS neural network Network by roughly 3.5% during iteration 1, 4.1% during iteration 2, 3.62 % during iteration 3, 4% during iteration 4, and 3.66 % during iteration 5.

Similar research includes There were previously two additional test case creation strategies that have materialized. [24] The method of multi-objective optimization, on which the mentioned algorithms are based, is used to achieve the desired outcomes. The Cuckoo search method, named after the bird's mating strategy, has been used to generate test cases. Developed a novel pairwise test suite generation technique based on a hybridization of the Artificial Bee Colony; they called it Pairwise Hybrid Artificial Bee Colony (PhABC). PhABC's result was a collection of optimum permutations of the test sets. The experiments demonstrated that the PhABC performed better than competing methods and produced more reliable test data than other research tactics that did not have the same limitations.[25]

6. Conclusion

Program testing is necessary since these mistakes might cause the program to crash. Regression test suites are used to select test cases and optimize goals that cut down on the amount of time and money required to maintain service-oriented business systems. The three phases of software testing are data production, data application, and data analysis. Classifiers have been used to produce and provide these findings. Priorities were used to categorize them into the most appropriate groupings, and items with the same priority were placed together.

References

[1] Williams, H & Bishop, M 2018, _Stochastic diffusion search: a comparison of swarm intelligence parameter estimation algorithms with ransac‘, Algorithms, vol. 7, no. 2, pp. 206-228.

[2] Aghdam, ZK & Arasteh, B 2017, _An efficient method to generate test data for software structural testing using artificial bee colony optimization algorithm‘, International Journal of Software Engineering and Knowledge Engineering, vol. 27, no. 06, pp. 951-966.

[3] Wu, H, Li, HR & Wan, JL 2016, _Improved genetic algorithm used in test cases‘, Computer Systems & Applications, vol. 8, no. 35.

[4] Yang, XS 2019, _Harmony search as a metaheuristic algorithm‘, In Music-inspired harmony search algorithm, Springer, Berlin, Heidelberg, pp. 1-14.

[5] Yao, Y & Liu, J 2018, _Metamorphic Testing for Oracle Problem in Integer Bug Detection‘, International Journal of Performability Engineering, vol. 14, no. 7.

[6] Yogi, MK & Yamuna, L 2017, _Robust Fault-Tolerant Training Strategy Using Neural Network to Perform Functional Testing of Software‘, International Journal of Advanced Networking and Applications, vol. 9, no. 3, pp. 3455-3460.

[7] Sharma, R & Saha, A 2020, _Identification of critical test paths using firefly algorithm for object oriented software‘, Journal of Interdisciplinary Mathematics, vol. 23, no. 1, pp. 191-203.

[8] Singh, A, Garg, N & Saini, T 20194, _A hybrid approach of genetic algorithm and particle swarm technique to software test case generation‘, International Journal of Innovations in Engineering and Technology, vol. 3, no. 4, pp. 208-214.

[9] Singh, K, Mishra, SK & Shrivastava, GA 2018, _Strategic Approach to Software Testing‘, International Journal of Information & Computation Technology, ISSN 0974-2239, vol. 4, no. 14, pp. 1387-1394

[10] Ragunath, PK, Velmourougan, S, Davachelvan, P, Kayalvizhi, S &Ravimohan, R 2020, _Evolving a new model (SDLC Model-2010) for software development life cycle (SDLC)‘, International Journal of Computer Science and Network Security, vol. 10, no. 1, pp. 112-119

[11] Rani, SBASU 2017, _A detailed study of Software Development Life Cycle (SDLC) models‘, International Journal of Engineering and Computer Science, vol. 6, no. 7.

[12] Rastogi, V 2018, _Software development life cycle models-comparison, consequences‘, International Journal of Computer Science and Information Technologies, vol. 6, no. 1, pp. 168-172. 89.

[13] Rhmann, W & Saxena, V 2016, _Optimized and prioritized test paths generation from UML activity diagram using firefly algorithm‘, International Journal of Computer Applications, vol. 145, no. 6, pp. 16- 22.

[14] Monsefi, AK, Zakeri, B, Samsam, S &Khashehchi, M 2019, _Performing software test oracle based on deep neural network with fuzzy inference system‘. In International Congress on High-Performance Computing and Big Data Analysis, Springer, Cham, pp. 406-417.

[15] Nakajima, S 2018, _Dataset diversity for metamorphic testing of machine learning software‘, In International Workshop on Structured Object-Oriented Formal Language and Method, Springer, Cham, pp. 21-38.

[16] Nascimento, AM, Vismari, LF, Cugnasca, PS, Júnior, JBC & de Almeira Júnior, JR 2019 _A Cost-Sensitive Approach to Enhance the use of ML Classifiers in Software Testing Efforts‘, In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE, pp. 1806-1813.

[17] Nayak, N & Mohapatra, DP 2020, _Automatic test data generation for data flow testing using particle swarm optimization‘, In International conference on contemporary computing, Springer, Berlin, Heidelberg, pp. 1-12.

[18] Le Thi My Hanh, KT & Tung, NTB 2018, _Mutation-based test data generation for simulink models using genetic algorithm and simulated annealing‘, International Journal of Computer and Information Technology, vol. 3, no. 04, pp. 763-771.

[19] Kale, S & Murthy, YS 2017, _Hybrid firefly algorithm based regression testcase prioritisation‘, International Journal of Business Intelligence and Data Mining, vol. 12, no. 4, pp. 340-357.

[20] He, Z, Chen, Y, Huang, E, Wang, Q, Pei, Y & Yuan, H 2019, _A system identification based Oracle for control-CPS software fault localization‘, In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, pp. 116-127.

[21] Jahan, H, Feng, Z, Mahmud, SM & Dong, P 2019, _Version specific test case prioritization approach based on artificial neural network‘, Journal of Intelligent & Fuzzy Systems, vol. 36, no. 6, pp. 6181-6194.

[22] Lachmann, R 2018, _Machine learning-driven test case prioritization approaches for black-box software testing‘, In The European Test and Telemetry Conference, Nuremberg, Germany.

[23] Souza, FCM, Papadakis, M, Le Traon, Y &Delamaro, ME 2022, _Strong mutation-based test data generation using hill climbing‘, In Proceedings of the 9th International Workshop on Search-Based Software Testing, pp. 45-54.

[24] Choudhary, K, Gigras, Y & Rani, P 2020, _Cuckoo Search in Test Case Generation and Conforming Optimality using Firefly Algorithm‘, In Proceedings of the Second International Conference on Computer and Communication Technologies, Springer, New Delhi, pp. 781-791.

[25] Alazzawi, AK, Rais, HM & Basri, S 2018, _Hybrid Artificial Bee Colony Algorithm for t-Way Interaction Test Suite Generation‘, In Computer Science On-line Conference, Springer, Cham, pp. 192-199.