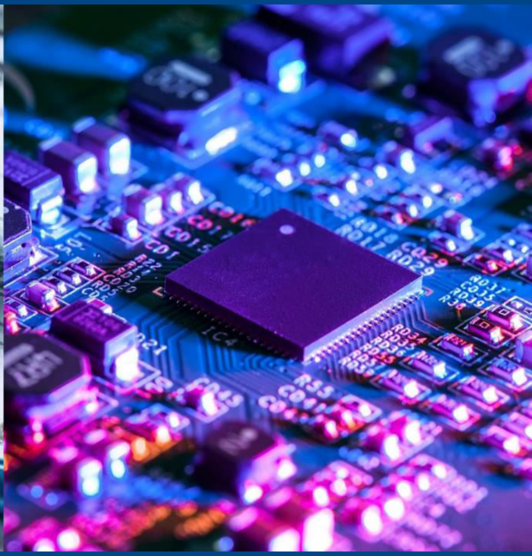
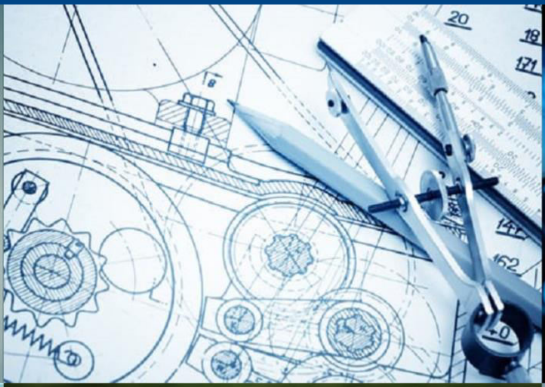
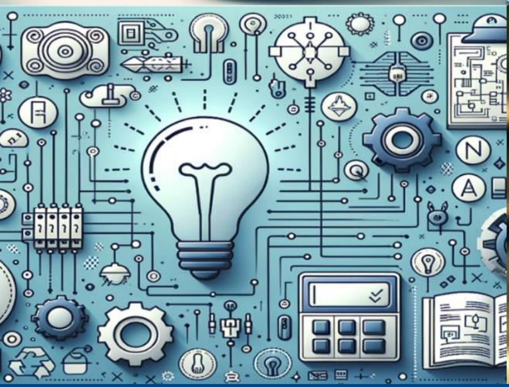
ISSN: 2582-7219



International Journal of Multidisciplinary Research in Science, Engineering and Technology

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)





Impact Factor: 7.521

Volume 8, Issue 1, January 2025

© 2025 IJMRSET | Volume 8, Issue 1, January 2025| DOI: 10.15680/IJMRSET.2025.0801049

ISSN: 2582-7219 | www.ijmrset.com | Impact Factor: 7.521| ESTD Year: 2018|

International Journal of Multidisciplinary Research in Science, Engineering and Technology (IJMRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Analysis on GenAI for Source Code Scanning and Automated Software Testing

Praveen Sivathapandi[1]*, **Girish Wali**[2]

Health Care Service Corporation, USA[1*]

CITI Bank, USA[2]

ABSTRACT: The fundamental purpose of software testing is to develop new test case sets that demonstrate the software product's deficiencies. Upon preparation of the test cases, the Test Oracle delineates the expected program behavior for each scenario. The application's correct functioning and its properties will be assessed by prioritizing test cases and running its components, which delineate inputs, actions, and outputs. The prioritization methods include initial ordering, random ordering, and reverse ranking based on fault detection capabilities. software application development often used a test suite, which was less well recognized as a suite for validating software correctness. Each test case set in the suite had distinct instructions and goals based on the system and its configuration that were evaluated. This article presents a Generative AI artificial neural network model for automated software testing based on particle swarm optimization. Generative AI is the method by which computers use existing data, including text, audio/video files, images, and code, to produce new material. An artificial neural network (ANN) is a complex adaptive system capable of altering its internal structure in response to the information it processes. Precisely manipulate the connection and its weight to achieve optimal accuracy. The PSO is a heuristic, population-based global optimization method.

KEYWORDS: Generative AI, Artificial Neural Network, Software Testing Automation, PSO,

I. INTRODUCTION

Generative AI is the method by which computers use existing data, including text, audio/video files, images, and code, to produce new material. This material may be classified as either unsupervised or semi-supervised. The primary aim is to create entirely original artifacts that replicate the look of genuine ones [1].

Software testing, within the context of the Software Development Life Cycle (SDLC), involves executing and evaluating a program to identify its defects. Tasks that evaluate the program's or system's capacity to ascertain compliance with requirements sometimes assume primary importance. The software system closely resembles other types of physical systems that accept inputs or generate outputs. Nevertheless, no tangible alterations have been implemented in the software. Typically, no alterations occur unless the user solicits a modification or enhancement. Consequently, the design defects or vulnerabilities may persist in a latent state inside the deployed software until triggered. Further testing and validation of all this data is necessary. Nonetheless, conducting testing accurately may be challenging. While not the primary objective of the project, the code is made available in case of a failure during a preliminary phase, and the software operates well for the test case. This occurs just with the unreliable test cases addressed during pre-debugging [2].

Software testing is the primary procedure that evaluates and executes the software to identify defects. This procedure entails rigorously testing the system's components and then assessing them using automated tools to ascertain if the system fulfills the requirements and, if not, to identify any inconsistencies between the anticipated and actual outcomes. The technique has two components and is quite generic. The first section explores the steps of the testing life cycle in more depth and examines an ideal testing methodology. Subsequent are enumerations of testing methodologies. The first portion emphasizes the fundamental duties of Analysis (A), Planning and Preparation (P), Execution (E), and Closure (C). The closure encompasses the execution phase activities, which include tracking, problem resolution, release, and root cause analysis.

Developing software to fulfill particular criteria is a fundamental aspect of software development. Software testing is essential for verifying and certifying that the generated software meets the specified criteria. The client may seek alternatives if this is not the situation. To ensure the consumer receives the appropriate software solutions, testing is a viable choice. Testing is fundamentally concerned with ensuring the finished product is appropriate for building. We routinely identify system issues that may render the software inoperative. Moreover, this facilitates the prevention of system malfunctions.

Dynamic analysis refers to automated testing. The term often used in software testing delineates the testing procedure and its interaction with the code or program. It pertains to examining the bodily responses to systemic elements, which are dynamic and evolve over time. Executing and adhering to the program is essential for dynamic testing. This contrasts with statistical testing since it entails engaging with the program via input values and then confirming, using automated or manual methods, if the actual output aligns with the anticipated results. Dynamic testing is used in system, acceptance, integration, and unit testing [3] [4].

II. LITERATURE SURVEY

2.1 Generative AI Neural Networks

Monsefi et al. introduced a novel oracle using deep learning and the fuzzy inference method. The software result was developed with data and a deep neural network, employing the Takagi-Sugeno-Kang fuzzy inference methodology. The output was allocated to its ambiguous domain. In the last phase, the competitor stage of the input utilizes the remapped data in its original format. To validate the oracle and assess its performance, four models are chosen to evaluate the enforcement of the oracle. Oracle will manually alter the source codes upon completion of the training, using the appropriate applications and their outcomes. The subsequent stage is to evaluate the efficacy of the oracle. Various criteria were used to assess the accuracy of the test oracle. The oracle has been seen to accurately determine outcomes as true or false in most cases. The system architecture will be the primary element for the oracle design, necessitating numerous preceding tasks [5].

Utilizing the Adopted Autoregressive-System Identification (AR-SI), he and his colleagues presented an oracle. This effectively enabled the control of hitherto opaque physical systems. Furthermore, it was altered to detect the defective black-box Control-Cyber-Physical System (CPS). Subsequently, it proposes a methodology for producing the traces for Control-CPS debugging and serves as a diagnostic outcome to evaluate the behavior of the Control-CPS. The conventional Control-CPSs underwent extensive evaluations using either genuine or fake bugs. The results regarding latency, accuracy/recall, and false positive/negative rates indicated that the system surpassed a human oracle approach in the SFL. This approach facilitated the identification of an additional real-world problem associated with Control-CPS for customers.

Valueian et al. proposed an alternative black-box approach for constructing specialized automated oracles suitable for various low-observability software systems. This technique employs the ANN algorithm, using a training set comprised of various input values and their corresponding pass/fail outcomes for the program under examination. Comprehensive evaluations of several benchmarks were conducted to assess the proposed approach and its efficacy. Compared to other prevalent machine learning approaches, their results indicate that the methodology is more precise and suitable for software systems with limited observability [6].

Jahan et al. advocated investigating the potential of ANN-based algorithms to improve the strategy of version-specific prioritization of test cases. This approach use a combination of many test cases with the ANN to detect significant faults. Priorities were assigned to three proposed training components. Two distinct software applications were used to do empirical assessments of the strategy. We also examined other effectiveness metrics such as accuracy, precision, recall, and the rate of issue detection. The findings demonstrated that the method was both feasible and successful [7].
Deep Evolution, proposed by Braiek and Khomh, is a novel approach for deep learning models that use metaheuristics to ensure maximum diversity in the generated test cases. A significant increase in neural coverage for the generated instances was seen while assessing Deep Evolution's effectiveness in evaluating computer vision deep learning models. Furthermore, several further corner case behaviors may be accurately discerned by Deep Evolution. Ultimately, in identifying various hidden defects arising from the quantization of these models, Deep Evolution outperformed

Tensorfuzz, a coverage-guided fuzzing tool developed by Google Brain. The findings illustrated the efficacy of these search-based approaches in creating testing instruments for deep learning systems [8].

Zhao and Gao introduced an alternative AI methodology for software testing using a scenario deductive approach. This model employs a stochastic method to generate test case inputs, integrating environmental factors, input/output parameters, and other elements. Subsequently, it utilizes a systematic approach to produce software testing assertions that autonomously verify test results. Following the exposition of the theory, the article employs an intelligent tracking vehicle to demonstrate the application of the approach to practical situations. Upon completion of the theory and description, an intelligent tracking vehicle is used to illustrate the technology, its applications, and the concerns that need attention. Finally, the article addresses the method's shortcomings and prospective avenues for further research [9].

Liu and Nakajima, in their continuation of the "Vibration-Method" or "V-Method," elucidated a further innovative strategy for the automated generation of test cases and test oracles derived from model-based requirements. The proposed methodology proved effective for evaluating data-intensive information systems. Methods for generating test cases derived from various functional situations are elucidated via the use of "divide and conquer" principles. A test oracle may also be produced by a well defined method. The methodology for the automated generation of test cases and the use of the Vibration technique were thoroughly examined. The effectiveness of the method was also evaluated in a separate controlled experiment. This facilitated the discussion on other relevant issues about the methodology, its effectiveness, and its feasibility [10].

Researchers in software testing are now concentrating on a novel approach known as Search-Based Software Testing (SBST), presented by Khari and Kumar in 2019. This is referred to as the optimization of software testing jobs by the development of test cases with metaheuristics. Heuristic search may facilitate the identification of optimal fitness results while seeking a more efficient and cost-effective method for testing and automating test case generation. Several unforeseen problems persisted despite the generation of test data based on search criteria. This research aimed to identify the most significant advancements and trending topics in search-based software testing by examining various methodologies and relevant literature in the field of software testing. A survey of studies on search-based software testing from 1996 to 2016 was conducted using metaheuristics [11].

Bodiwala and Jinwala proposed an alternate approach for generating test cases. This mostly concentrates on three specific bio-inspired techniques: the firefly algorithm, bacterial foraging, and genetic algorithms. These approaches facilitate code coverage via test data and the automated generation of test cases. In comparison to the Genetic Algorithm (GA), the findings of the test data generation indicated that the Bacterial Foraging Algorithm and the Firefly Algorithm attained enhanced coverage.

Nakajima (2018) proposed an alternative metamorphic testing methodology for neural network learning models. The dataset was the primary focus, apart from diversity and its behavioral oracle. Alongside enhancing the generation of follow-up test inputs, the dataset's variety considers the dependency of training results on the dataset. During the training process, the behavioral oracle monitors may modify statistical indicators that form the basis of the metamorphic relations requiring verification. Several instances of software testing for neural network algorithms that classify handwritten digits were used to illustrate the proposed methodology [12].

Gholami et al. proposed an innovative black box method for developing various autonomous oracles in low-observability embedded software systems. A novel model was developed using the Artificial Neural Network (ANN) approach. This approach necessitated input values and yielded a pass or fail outcome. A multitude of benchmarks served as the foundation for several extensive tests. Compared to other prominent machine learning approaches, their results illustrate the proposed approach and its appropriateness for software systems [13]. Sathyavathy's proposed Software Development Life Cycle emphasizes the critical role of software testing in identifying defects and enhancing software performance. Automated testing employs several sophisticated approaches that eliminate the need for human involvement. Automated testing methods are used to enhance software quality [14].

Lachmann proposed a supervised machine learning approach for ranking test cases in system testing. Subsequently, the approach analyzes meta-data and natural language artifacts to select supplementary manually executed test cases. This research employs an innovative ensemble learning approach with conventional machine learning methods. This research employs ensemble learning approaches with other machine learning methodologies. more assessment of this approach was performed with three more datasets sourced from the car industry, therefore including real-world datasets. The results are assessed based on their capacity to detect faults. The findings indicate that the use of these machine learning methodologies may enhance black-box testing [15].

2.2 Optimization Techniques

Sharifipour et al. proposed an alternative memetic Ant Colony Optimization (ACO) approach for the structural generation of test data. Integrating (1+1)-Evolution Strategies (ES) into the methodology enhanced the ant's search efficacy during local migrations and exploitation. The authors proposed an alternative perspective on the role of pheromones, which inhibited the ants from using the obscured trails for their search engine optimization tactics. Because branch coverage is the coverage criterion, the approach employs two fitness functions. To optimize branch coverage, the initial criterion is established as a Boolean function. If the solution successfully traverses an exposed branch, it will yield one; otherwise, it will revert to zero. The complexity of the branches influenced the development of the later fitness function. Values corresponding to Boolean functions set to one were excluded. The ants' decision-making mechanism relied on its original fitness function for these alternatives. The experimental results demonstrated that our memetic ACO algorithm surpassed the test data generation techniques for convergence speed and coverage [16].

The Test Generator Flower Pollination approach (TGFP), first proposed by Alsewari et al. and originating from the Flower Pollination Algorithm (FPA), is a supplementary method focused on test case reduction. The analytical and experimental findings indicated the efficacy of the proposed strategy relative to current combinatorial testing techniques. The evaluation research outcomes indicated that the TGFP could successfully overcome the test situations. The research indicated that the TGFP might potentially enhance test cases for various t-way approaches [17].

Kalaee and Rafe (2016) proposed an additional effective strategy for generating a compact set of tests that comprehensively address a wide range of input parameters. To accomplish this, we used the Reduced Ordered Binary Decision Diagram (ROBDD) to ascertain the primary cause-and-effect graph influences. Furthermore, the ROBDD established a unique expressive representation by articulating the graph as a Boolean function. The use of ROBDD facilitates a reduction in the size of the generated test suite, hence enhancing testing speed. The proposed method employs Particle Swarm Optimization (PSO) to choose the optimal test suite, including all potential input parameters and pairwise combinations. The findings indicated that the strategy enhanced efficiency and reduced testing expenses for relevant test cases. Furthermore, it surpassed other advanced black-box testing approaches [18].

Aghdam and Arasteh used the Artificial Bee Colony (ABC) technique, using branch coverage requirements as a fitness function, to enhance their solutions for test data generation issues. Seven unique conventional programs used as benchmarks for these comparisons in the literature. The testing findings indicated that the approach surpassed Simulated Annealing, GA, PSO, and ACO, achieving a 99.94% success rate, an average convergence generation of 0.18 and 3.59, and 99.99% average branch coverage.

Souza et al. (2016) proposed an alternate automated technique for test creation using Hill Climbing to enhance mutation strength. This is a methodical strategy that focuses on mutations and their capacity for replication. The study also presents empirical facts linked to cost and efficacy. The assemblage of 18C programs is used to compute this. The technique surpassed random testing by 19.02% in strong mutation ratings for the majority of the programs analyzed. The prior approaches that ignored mutation spread had an average accuracy of 7.2%. In comparison to other methodologies, their results demonstrated a significant increase in efficiency [19].

A unique approach proposed by Huo et al. employs a Genetic Algorithm (GA) to generate multi-objective test data. The objective of comparing the two strategies was to use two distinct methodologies grounded on the Genetic Algorithm (GA). To further assess and examine the performance of the GP-based algorithms, three more multi-objective optimization frameworks were used. The study was based on two concerns related to test data production.

Both the integer and the double were identical. There are 160 benchmark programs, each exhibiting differing levels of nesting. The new GP surpassed rival GA-based techniques using a random search baseline approach, as shown by the findings [20].

III. METHODS

This section presents particle swarm optimization based Generative AI Artificial Neural Network model for automated software testing.

Particle Swarm Optimization (PSO), a robust meta-heuristic optimization technique, is inspired by the collective behavior seen in natural swarms, such as those of fish and birds. PSO is fundamentally concerned with simulating a user-friendly framework. The primary objective of the PSO algorithm was to graphically replicate the fluid but transient movement of a bird in flight. The bird's observable range is limited in its natural habitat. A flock of birds may explore a broader expanse of a fitness function than an individual bird could alone. To train the swarm to identify the global minima of a fitness function, it is necessary to statistically replicate the specified criteria.

Particle Swarm Optimization (PSO) is a population-based heuristic search method used for global optimization. John Kennedy and Eberhart conceived this in 1995. The domains of computer science, engineering, social psychology, and artificial life were the fundamental origins of the PSO. It traverses hyperspace using specified velocities, aided by particles and their quantities. The velocity of each particle. A user-defined fitness function may be used to optimize outcomes for both the particle and the neighborhood. The motion of each particle evolves naturally within the solution, which may be optimal or nearly so. This disordered assemblage of particles is referred to as a swarm, exhibiting behavior akin to that of mosquitoes rather than a school of fish or a flock of birds. Computational intelligence can converge on optimal solutions to problems, regardless of size or non-linearity [21]. Subsequent assessments will rely on the fitness values derived by executing a fitness function on each particle. Within the vicinity of the PSO implementation, lbest represents the optimal population so far.

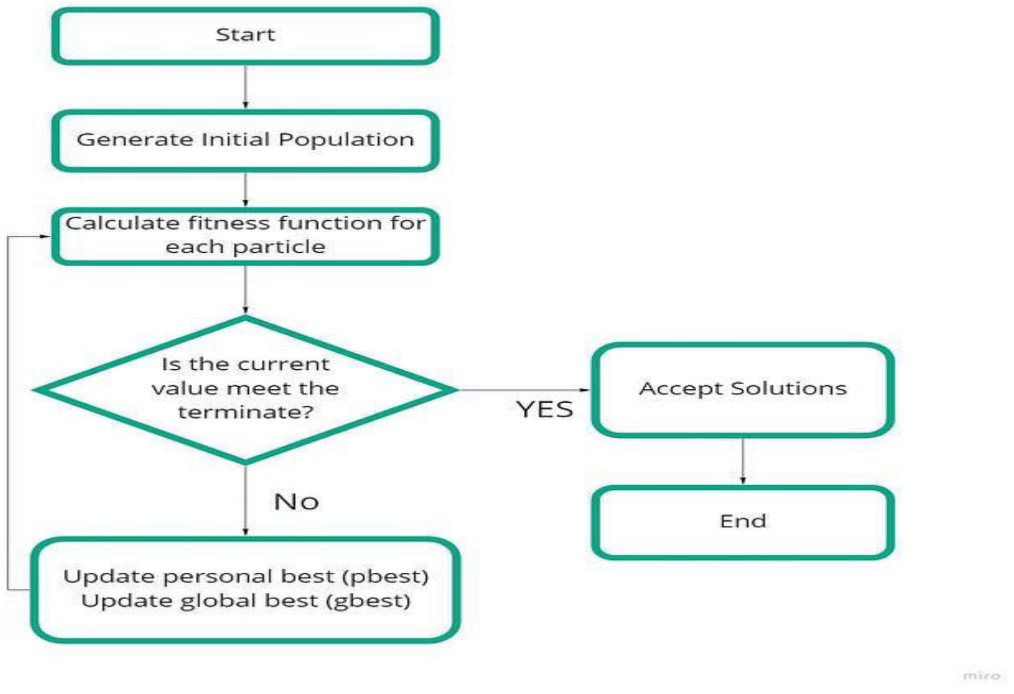

Start
Generate Initial Population
Calculate fitness function for each particle
Is the current value meet the terminate?
YES
Accept Solutions
No
End
Update personal best (pbest)
Update global best (gbest)
miro

Figure 1: Flow diagram of PSO optimization algorithm

The ANN may alter its internal architecture based on the data processed; it is a complex and flexible system. Altering its weight and connection may also achieve this. The weight of a signal is the actual number that governs its influence across several neurons. Subsequently, these weights are adjusted to improve the results [22]. An exemplary representation of a supervised learning paradigm is an Artificial Neural Network (ANN). It may acquire knowledge via interactions with other nodes within a network. The derivation and classification of data mining rules have been necessitated by the difficulty in acquiring such information. Subsequently, we start with a dataset that can be partitioned into two equal halves. These people function as both the training and testing samples. The former is used to train the network, whilst the latter is employed to assess the classifier's effectiveness. Prevalent methods for data partitioning including random sampling, the hold-out approach, and cross-validation.

The steps involved in the neural network are:
- A structure is defined using nodes in hidden, input, and output layers.
- An algorithm is employed to enable the learning process.

To enhance its use for artificial intelligence, the neural network must ensure that the architecture is modified and that the weights are often tweaked. The artificial neural networks were trained using a dataset including diverse inputs and corresponding results generated based on logical principles. The last phase was using it to validate the case studies. To facilitate the artificial neural network's (ANN) learning from a diverse array of numerical values, all non-numerical inputs and outputs were normalized prior to the commencement of training. The artificial neural network was built with a mix of training samples and several input/output datasets. Prior to transitioning to the new training phase, expert domain knowledge and the verified software requirements from other reliable approaches were used to provide training examples. The dataset's correctness is essential, since the artificial neural network (ANN) depends on this data to replicate the system under test (SUT) and its behaviors.

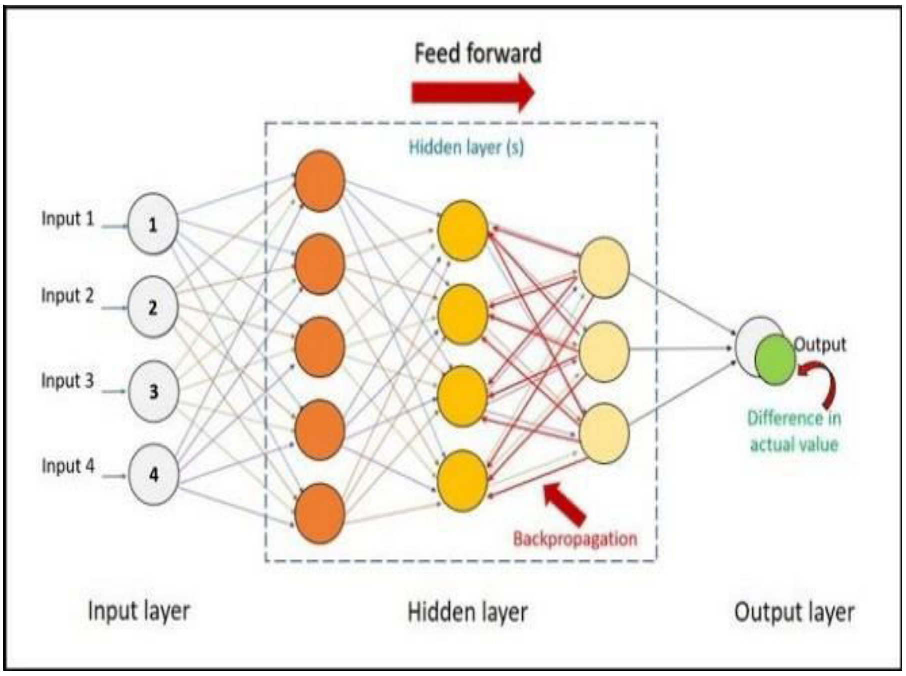


Figure 2: ANN Neural Network

To guarantee the accuracy of the test oracle, it is essential to use each record of the training samples throughout the training phase. The ANN's output was evaluated against the right output to improve the network's quality. We computed the discrepancy between the projected outcomes and the actual results. The network parameters, including the weights of the biases and neurons, were optimized via back-propagated error data to achieve an acceptable error rate. The whole method was repeated until an acceptable error rate was attained. The network is now ready for deployment after the conclusion of the new adjustment cycles. The ANN promptly generated the expected outputs for the input at the completion of training, using a training dataset. The network design was suitably modified using the trial-and-error method due to the erroneous results of the ANN.

Back Propagation (BP) was identified as a prevalent technique for training Artificial Neural Networks (ANNs) in optimization challenges, akin to Gradient Descent. The algorithm computed the gradients of all loss functions related to the network weights. The total number of input instances was established by using the appropriate quantity of input nodes, encompassing the hidden layers as well as the output layers n, k, and m.

IV. RESULTS

We trained the neural network using processed data rather than a conventional training dataset, and we were aware of their parameters prior to initiating the training. During the network training, the whole dataset was shown for a single epoch, and the precise number of epochs was specified. The back-propagation training procedure concluded when the maximum number of epochs was attained or the minimum error rate was achieved. Subsequently, the network functioned as an oracle, predicting the accurate outcomes for the subsequent regression testing. Training utilizes 80% of the data, and testing use the remaining 20%. During the experiments, we used a 10-fold cross-validation methodology. The misclassification rate, expressed as a percentage, is calculated using the formula: number of features incorrectly categorized divided by the total number of features. In the realm of metaheuristic algorithms, "convergence" refers to the rate at which optimal or sufficiently satisfactory solutions are achieved. Results are shown in table 1 and figure 3.

Table 1: Misclassification of Correct Output

No of Software faults	ANN results (%)	PSO ANN results (%)
1	6.25	3.15
2	7.10	3.65
3	7.92	4.26
4	8.62	5.10
5	10.89	6.38

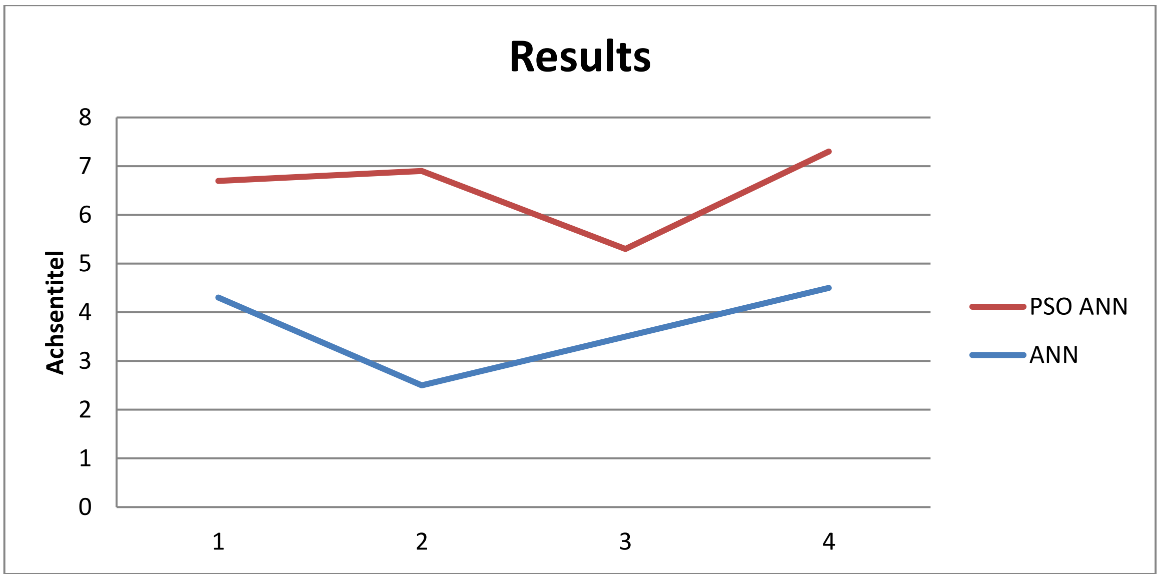


Figure 3: Misclassification of Correct Output

V. CONCLUSION

The application's correct operation and attributes will be evaluated by prioritizing test cases and executing its components, which define inputs, actions, and outputs. Prioritization strategies include initial ordering, random ordering, and reverse ranking based on defect detection skills. program application development often used a test suite, which was less widely known as a suite for evaluating program correctness. Each test case set in the suite had specific instructions and objectives depending on the system and configuration being examined. This article introduces a Generative AI artificial neural network model for automated software testing that uses particle swarm optimization. Generative AI is the process by which computers create new content from existing data, such as text, audio/video files, photos, and code. An artificial neural network (ANN) is a sophisticated adaptive system that may change its internal structure in response to the data it processes. To reach the highest level of precision, precisely control the connection and its weight. The PSO is a heuristic, population-based global optimization technique.

REFERENCES

1. M. Jovanovic and M. Campbell, "Generative artificial intelligence: Trends and prospects," Computer, vol. 55, no. 10, pp. 107–112, Oct. 2022, doi: 10.1109/MC.2022.3192720.
2. H.-Y. Lin, "Large-scale artificial intelligence models," Computer, vol. 55, no. 5, pp. 76–80, May 2022, doi: 10.1109/MC.2022.3151419.
3. Z. Akata et al., "A research agenda for hybrid intelligence: Augmenting human intellect with collaborative, adaptive, responsible, and explainable artificial intelligence," Computer, vol. 53, no. 8, pp. 18–28, Aug. 2020, doi: 10.1109/MC.2020.2996587.
4. I. Ozkaya, "Application of large language models to software engineering tasks: Opportunities, risks, and implications," IEEE Softw., vol. 40, no. 3, pp. 4–8, May 2023, doi: 10.1109/ MS.2023.3248401
5. 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.
6. Valueian, M, Attar, N, Haghighi, H & Vahidi-Asl, M 2020, Constructing automated test oracle for low observable software', Scientia Iranica, vol. 27, no. 3, pp. 1333-1351.
7. 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.
8. Braiek, HB & Khomh, F 2019, DeepEvolution: A Search-Based Testing Approach for Deep Neural Networks', In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, pp. 454-458.
9. Zhao, X & Gao, X 2018, An ai software test method based on scene deductive approach'. In 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), IEEE, pp. 14-20.
10. Liu, S & Nakajima, S 2020, Automatic Test Case and Test Oracle Generation based on Functional Scenarios in Formal Specifications for Conformance Testing', IEEE Transactions on Software Engineering.
11. Khari, M & Kumar, P 2019, An extensive evaluation of search-based software testing: a review', Soft Computing, vol. 23, no. 6, pp. 1933-1946.
12. Bodiwala, SS & Jinwala, DC 2016, Optimizing test case generation in glass box testing using Bio-inspired Algorithms', In 2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS), IEEE, pp. 40-44.
13. Gholami, F, Attar, N, Haghighi, H, Asl, MV, Valueian, M & Mohamadyari, S 2018, A classifier-based test oracle for embedded software', In 2018 Real-Time and Embedded Systems and Technologies (RTEST), IEEE, pp. 104-111.
14. Sathyavathy, V 2017, Evaluation of software testing techniques using artificial neural network', Int. J. Electr. Comput. Sci, vol. 6, no. 3, pp. 20617-20620.
15. Lachmann, R 2018, Machine learning-driven test case prioritizationapproaches for black-box software testing', In The European Test and Telemetry Conference, Nuremberg, Germany.
16. Sharifipour, H, Shakeri, M & Haghighi, H 2018, Structural test data generation using a memetic ant colony optimization based on evolution strategies', Swarm and Evolutionary Computation, vol. 40, pp. 76-91.
17. Alsewari, AA, Har, HC, Homaid, AAB, Nasser, AB, Zamli, KZ & Tairan, NM 2017, Test cases minimization strategy based on flower pollination algorithm', In International conference of reliable information and communication technology, Springer, Cham, pp. 505-512.

18. Kalaee, A & Rafe, V 2016, An optimal solution for test case generation using ROBDD graph and PSO algorithm‘, Quality and Reliability Engineering International, vol. 32, no. 7, pp. 2263-2279.
19. El-henawy, IM & Ismail, MM 2014, A hybrid swarm intelligence technique for solving integer multi-objective problems‘, International Journal of Computer Applications, vol. 87, no. 3, pp. 45-50.
20. 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.
21. Du, KL & Swamy, MNS 2016, _Particle swarm optimization‘, In Search and optimization by metaheuristics, , Birkhäuser, Cham, pp. 153-173.
22. Dwivedi, AK 2018, _Artificial neural network model for effective cancer classification using microarray gene expression data‘, Neural Computing and Applications, vol. 29, no. 12, pp. 1545-1554.