International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 02, February 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

AN HYBRID MULTILAYER PERCEPTRON USING GSO-GA FOR SOFTWARE DEFECT PREDICTION

V.Jayaraj[1], N. Saravana Raman[2]

[1]*Bharathidasan University, Trichy, Tamil Nadu (India)*

[2]*Research Scholar, Bharathidasan University Trichy, Tamil Nadu (India)*

ABSTRACT

Software defect prediction has turned into an expected requirement for organizations to guarantee quality and reliability of software products. The early defect prediction can encourage managers to amend and improve reliability of product. Methodologies, for example, machine learning and neural network have ended up as eminent solution for training and classification of data and can be important for defect prediction. Though, these methodologies need optimizationfor weight update, parametric improvement while performing defect prediction. In this paper a hybrid Glowworm Swarm Optimization (GSO) - Genetic Algorithm (GA) to optimize the Multi-Layer Perceptron Neural Network (MLPNN) is proposed.

Keywords: Genetic Algorithm (GA),Glowworm Swarm Optimization (GSO), Multi-Layer Perceptron Neural Network (MLPNN),Software Defect Prediction (SDP)

I. INTRODUCTION

Software Engineering is a specialty that targets for creating high quality software through precise, well developed methodology of software development. To send high quality software, it is vital to add to acquire defect-freesoftware deliverable at every phase [1]. A defect is any flaw, imperfection, or undesired activities that happen either in the deliverable or in the product.

Software development organizations are under more pressure than any time in recent times. Development costs keep on rising. Minimizing defects is anefficient approach to hold development costs down, which is a main concern for any association. Also, the cost of repairing defects increments exponentially as software advances through the development lifecycle; it's important to catch defects as promptlyas conceivable [2].

To oversee software defects three levels are utilized defect discovery, defect analysis and defect prevention. At first level, test software work product until all the defects are found and repaired. Despite the fact that it is impractical to test the software hundred percent means totally. However at this level it is accepted to recognize numerous defects as could be expected under the circumstances this should be possible through static analysis and robotized testing instruments [3]. The second level is defect analysis in this level previously recognized defects are investigated and time is spent to look their underlying drivers and why they were not identified before. The Third level is defect prevention in this process particular strategies

Defects happen amid every one of the phases of the Software development life cycle. Thus defect prevention is exceptionally key some portion of Software development Life cycle for enhancing the Software Quality [4]. Defect Prediction recognize and prevent the defect bringing about failures before happening. This is finished by

first picking up the encounters from the prior Software Projects by Software Engineers and after that distinguishing the underlying driver for the defects and after that taking out the reasons.

Defect Analysis is utilizing defects as data for ceaseless quality change. Defect analysis by and large looks to classify defects into classifications and distinguish conceivable reasons so as to direct process change efforts. Main driver Analysis (RCA) has assumed valuable parts in the analysis of software defects. The objective of RCA is to distinguish the main driver of defects and start activities so that the source of defects is wiped out [5]. To do as such, defects are examined, each one in turn. The analysis is subjective and just constrained by the scope of human investigative abilities. The subjective analysis gives feedback to the designers that in the long run enhance both the quality and the profitability of the software association.

Defect prediction metrics assume the most essential part to fabricate a measurable prediction model. Most defect prediction metrics can be sorted into two sorts: code metrics and process metrics. To distinguish defect modules, the prediction models can then be utilized by the software associations, amid the early phases of software development. The software associations can utilize this subset of metrics amongst the accessible substantial arrangement of software metrics. These metrics can be utilized as a part of adding to the defect prediction models [6].

Software Defect Prediction (SDP) procedures are utilized either to classify which modules are defect-inclined or to predict the quantity of defects anticipated that would be found in a software module/venture. Various diverse systems have been utilized with the end goal of classification1/predicting defects; they can be comprehensively gathered into procedures utilized for predicting anticipated that number of defects would be found in a given software relic (Prediction) and methods that are utilized to predict if or not a given software ancient rarity is liable to contain a defect (Classification).

The prediction models might just utilize number of defects found amid development and testing without considering other attributes identified with the inward structure/design/implementation of the venture/item – these are assembled as black box defect prediction strategies [7]. Then again defect prediction strategies that utilization attributes identified with process and item e.g. size, complexity, changes are classified under white box systems

1.1 Software Defect Prediction Process

- Generate cases from software archives, for example, variant control frameworks, issue following frameworks, email archives, et cetera.
- After producing examples with metrics and names, we can apply preprocessing strategies, which are regular in machine learning.
- Preprocessing systems utilized as a part of defect prediction ponders incorporate feature selection, data normalization, and noise reduction.
- The prediction model can predict whether another example has a bug or not [8].

Amid the software defect prediction process, two sorts of misclassification errors can be experienced. The sort I misclassification happens when a not-fault-inclined module is predicted as fault-inclined one while a sort II misclassification is that a fault inclined module is classified as not-fault-inclined [9]. A sort I misclassification will bring about the exercise in futility and resources to survey a non-faulty module. A sort II misclassification

results in the missed chance to revise a faulty module that the faults might show up in the framework testing or even in the field.

Machine learning (ML) algorithms has exhibited awesome handy essentialness in determining an extensive variety of building issues including the prediction of failure, error, and defect-impulsions as the framework software develops to be more mind boggling. ML algorithms are extremely helpful where issue domains are not all around characterized, human information is constrained and dynamic adaption for changing condition is required, keeping in mind the end goal to create productive algorithms [10].

Machine learning includes distinctive sorts of learning, for example, artificial neural systems (ANN), concept learning (CL), Bayesian conviction systems (BBN), reinforcement learning (RL), hereditary algorithms (GA) and hereditary programming (GP), occurrence based learning (IBL), decision trees (DT), inductive logic programming (ILP), and analytical learning (AL).

Classification is a data mining and machine learning approach, valuable in software bug prediction. It includes order of software modules into defective or non-defective that is signified by an arrangement of software complexity metrics by using a classification model that is gotten from before development ventures data. The metrics for software complexity might comprise of code size, McCabe's cyclomatic complexity and Halstead's Complexity [11].

Clustering is a sort of non-hierarchal strategy that moves data focuses among an arrangement of groups until comparative thing bunches are framed or a sought set is gained. Clustering strategies make suspicions about the data set. In the event that that suspicion holds, then it results into a decent group. Be that as it may, it is an inconsequential errand to fulfill all suppositions. The blend of diverse varying so as to cluster systems and info parameters may be valuable. Affiliation standard mining is utilized for finding successive patterns of diverse attributes in a dataset. The affiliated classification the greater part of the times gives a higher classification when contrasted with other classification systems.

In this work, we utilized hybrid GSO-GA to prepare MLP neural system. Segment 2 manages writing work, area 3 uncovers the strategies utilized as a part of the work. Area 4 examines the investigation results lastly segment 5 finishes up the work.

II. RELATED WORK

Agarwal and Tomar [12] utilized the Twin Support Vector Machine (TSVM) for predicting the quantity of defects in another form of software item. This model gives an about immaculate efficiency which contrasted with other models is far superior. Twin Support Vector Machine based software defects prediction model utilizing Gaussian kernel capacity acquires better performance as contrast with prior proposed methodologies of software defect prediction. By predicting the defects in the new form, we thereby endeavor to step to take care of the issue of keeping up the high software quality. This proposed display specifically demonstrated its effect on the testing phase of the software item by essentially falling the general cost and efforts put in.

Malhotra et al., [13] investigated the predictive capacity of the transformative calculation and hybridized developmental calculation strategies for defect prediction. The proposed work adds to the examining so as to write the viability of the 15 developmental calculation and hybridized transformative calculation procedures to 5 datasets acquired from the Apache Software Foundation utilizing the Defect Collection and Reporting System.

The outcomes were assessed as far as the estimations of accuracy. The transformative calculation strategies thought about utilizing the Friedman ranking. The outcomes recommended that the defect prediction models manufactured utilizing the transformative calculation procedures performed well over all the datasets as far as prediction accuracy.

Okumoto et al., [14] presented a software defect prediction model utilizing defect data from soundness test. The test run span in hours is a superior measure than timetable time in days for predicting the quantity of defects in a software discharge. An exponential reliability development model is connected to the defect data regarding test run length of time. Creators tended to how to recognize whether evaluations of the model parameters are sufficiently steady to assure the prediction accuracy.

Shepperd et al., [15] directed a meta-analysis of all important, fantastic essential investigations of defect prediction to figure out what components impact predictive performance. To conquer the abnormal state of specialist inclination, defect prediction scientists if (i) lead blind analysis, (ii) enhance reporting protocols and (iii) direct more intergroup studies with a specific end goal to mitigate aptitude issues. Finally, research is required to figure out if this predisposition is prevalent in other applications domains.

Chen et al., [16] audited the condition of workmanship in the field of software defect management and prediction, and presented data mining innovation quickly. At long last, proposed a perfect software defect management and prediction framework, looked into and broke down a few software defect prediction strategies in light of data mining procedures and particular models (Bayesian Network and PRM). With the proposed framework, we shall design few prevention and arrangement scheme to control the development of new software.

A proficient clustering approach, named as Package Based Clustering, proposed by Islam and Sakib [17] to assemble the software for predicting defects. To examine the proposed clustering algorithm, the linear regression model is utilized which gains from bunches of related and comparable classes. The test has been led on JEdit 3.2 and results demonstrated that the prediction model utilizing Package Based Clustering is 54%, 71%, 90% superior to the prediction models based on BorderFlow clustering, k-means clustering and the whole framework individually.

Mausa et al., [18] presented an apparatus exhibition that actualizes a precise data collection system for software defect prediction datasets from the open source bug following and the source code management storehouses. Fundamental testing issue that the instrument locations is connecting the data identified with the same element (e.g. class record) from these two sources. The instrument actualized interfaces to bug and source code vaults and even other devices for figuring the software metrics. At last, it offered the client to make software defect prediction datasets regardless of the fact that unconscious of the considerable number of subtle elements behind this mind boggling assignment.

Pushphavathi et al., [19] presented a novel hybrid technique for irregular woodland (RF) and Fuzzy C Means (FCM) clustering for building defect prediction model. At first, arbitrary backwoods algorithm is utilized to perform a preliminary screening of variables and to pick up a significance ranks. Consequently, the new dataset is information into the FCM procedure, which is in charge of building interpretable models for predicting defects. The capacity of this mix system is assessed utilizing essential performance estimations alongside a 10-fold cross approval. FCM and RF procedure is connected to software parts, for example, individuals, process,

which go about as significant decision making model for undertaking achievement. Exploratory results demonstrated that the proposed strategy gave a higher accuracy and a generally basic model empowering a superior prediction of software defects.

Lu et al., [20] proposed dynamic learning as an approach to mechanize the development of models which enhance the performance of defect prediction between progressive discharges. The outcomes demonstrated that the incorporation of dynamic learning with instability examining reliably outflanked the relating managed learning approach.

Going for the attributes of the metrics mined from the open source software, Wang et al., [21] proposed three new defect prediction models taking into account C4.5 model. The new models presented the Spearman's rank relationship coefficient to the premise of picking root hub of the decision tree which improves the models on defects prediction. Keeping in mind the end goal to confirm the adequacy of the enhanced models, a test scheme is designed. In the trial, the prediction exactnesses of the current models and the enhanced models were looked at and the outcome demonstrated that the enhanced models lessened the span of the decision tree by 49.91% all things considered and expanded the prediction accuracy by 4.58% and 4.87% on two modules utilized as a part of the investigation.

A model taking into account locally linear embedding and bolster vector machine (LLE-SVM) is proposed by Shan et al., [22]. The SVM is utilized as the essential classifier as a part of the model. What's more, the LLE algorithm is utilized to settle data excess because of its capacity of keeping up nearby geometry. The parameters in SVM are improved by the technique for ten-fold cross acceptance and network look. The correlation between LLE-SVM model and SVM model was tentatively checked on the same NASA defect data set. The outcomes demonstrated that the proposition LLE-SVM model performed superior to anything SVM model, and it is accessible to keep away from the accuracy diminish brought on by the data excess.

III. METHODOLOGY

In this work, hybrid Glow-worm Swarm Optimization (GSO) – Genetic Algorithm (GA) to prepare MLPNN is proposed.

3.1 NASA Dataset

The data utilized as a part of this study was taken from the NASA Metrics Data Program (MDP)- Metric Data Repository. The data repository contains software metrics and related error data at the capacity/system level. The data repository stores and sorts out the data which has been gathered and accepted by the Metrics Data Program. It at present contains 13 data sets proposed for software metrics research. Each of these data sets contains the static code metrics and comparing fault data for each involving module. Subsequent to preprocessing, modules that contain one or more defects were named as defective [23, 24 and 25]. Table 1 shows the NASA MDP dataset.

Table 1 NASA MDP Data Sets

Data Set	System	Language	Total Loc
CM1-5	Spacecraft Instrument	C	17K
KC3-4	Storage management for ground data	JAVA	8K and 25K
KC1-2	Storage management for ground data	C++	*
MW1	Database	C	8K
PC1,2,5	Flight Software for Earth orbiting Software	C	26K
PC3,4	Flight Software for Earth orbiting Software	C	30-36K

3.2 Neural Networks

Neural systems are prepared to perform complex capacities in different fields, including design acknowledgment, ID, classification, and vision, discourse, and control frameworks. Neural system depends on a machine learning approach. Neural systems can likewise be prepared to take care of issues that are troublesome for routine PCs or people [26]. Neural systems are additionally utilized as a part of different fields like data mining, picture processing, analytic frameworks and so on.

Neural systems comprise of numerous layers of computational units, generally interconnected in a food forward manner. Every neuron in one layer has guided associations with the neurons of the ensuing layer. In numerous applications the units of these systems apply a sigmoid capacity as an actuation capacity. The food forward neural system was the first and ostensibly most straightforward kind of artificial neural system concocted [27].

As the greater part of faults are found of its modules, there is a need to explore the modules that are influenced seriously when contrasted with other modules and appropriate support to be done on time particularly for the basic applications. Algorithms in light of neural systems have a considerable measure of utilization in knowledge sector.

3.3 Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron (MLP) is another sort of artificial neural system show that is prepared utilizing an administered learning method got back to propagation algorithm, which maps sets of information data onto an arrangement of proper yields. A MLP comprises of various layers of hubs in a coordinated chart: one information layer, one yield layer, and one or more hidden layers. The yield of a layer is utilized as the information of hubs in the resulting layer. MLP can recognize data that are not linearly distinguishable, which is superior to the standard linear perceptron [28].

MLPs endeavor to artificially imitate the working of a biological sensory system. Different hubs, or neurons, are associated in layers, with the yield of every hub being the thresholded weighted total of its inputs from the previous layer. It has been demonstrated that a different hidden layer neural system can estimated any capacity. In our study, the hiddenLayers parameter was changed to "3" to characterize a system with one hidden layer containing three hubs, and the validationSetSize parameter was changed to "10" to bring about the classifier to leave 10% of the preparation data aside to be utilized as an approval set to decide when to stop the iterative preparing process.

Assume the aggregate number of hidden layers is L. The info layer is considered as layer 0. Let the quantity of neurons in hidden layer l be Nl, l = 1,2,… ,L. Let w_{ij}^{l} represents the heaviness of the connection between the jth neuron of the l − 1th hidden layer and ith neuron of the lth hidden layer, and θ_i^l be the inclination parameter of ith neuron of the lth hidden layer. Let xi represent the ith data parameter to the MLPNN. Let y_i^{-1} be the yield of ith neuron of the lth hidden layer, which can be figured by standard MLPNN recipes as in equation (1):

$$\bar{y}_i^l = f\left(\sum_{j=1}^{N_{i-1}} w_{ij}^{l} . \bar{y}_j^{l-1} + \theta_i^l\right), i = 1,\ldots, N_l, l = 1,.., L$$
$$\bar{y}_i^0 = x_i, i = 1,\ldots, \mathrm{N}_x, N_x = N_0, \quad (1)$$

where f(.) is the actuation capacity. Let v_{ki} represent the heaviness of the connection between the ith neuron of the Lth hidden layer and the kth neuron of the yield layer, and β_k be the inclination parameter of the kth yield neuron. The yields of MLPNN can be processed as in equation (2):

$$y_k = \sum_{i=1}^{N_L} v_{ki} . y_i^{-L} + \beta_k$$
$$k = 1,\ldots, N_y \quad (2)$$

Bolster forward (feed forward) neural systems give a general structure to representing non-linear utilitarian mappings between an arrangement of data variables and an arrangement of yield variables. This is accomplished by representing the nonlinear capacity of numerous variables as far as creations of nonlinear elements of a solitary variable, which are called initiation capacities. The hidden layer of a MLP neural system commonly comprises of sigmoid capacity.

GA is a hunt heuristic that imitates the process of characteristic development [30]. This heuristic is routinely used to produce helpful answers for advancement and look issues. Singular arrangement is represented through a chromosome, which is only a rundown of representation. Along these lines, to locate the best arrangement, it is important to perform certain operations of every ideal arrangement.

Toward the starting it is creating a beginning population of chromosomes. This population must offer a wide differing qualities of hereditary materials and the quality pool ought to be as substantial as could reasonably be expected so that any arrangement of the inquiry space can be engendered. Then, the GA circles over a cycle process to make the population advance. Every emphasis comprises of selection, crossover, mutation and substitution. The quality of hereditary algorithms is that they quickly meet to close ideal arrangements.

The fitness of every chromosome is dictated by assessing it against a goal capacity. To mimic the characteristic survival of the fittest process, best chromosomes trade data to deliver offspring chromosomes. The offspring arrangements are then assessed, and used to develop the population in the event that they give preferred arrangements over powerless populations individuals [31]. As a rule, the process is proceeded for countless to get a best-fit arrangement. Table 2 shows the parameters of GA.

Table 2 GA Parameters

Parameter	Value
ρ	0.4
γ	0.6
β	0.08
n_t	5
s	0.03
l_0	5

To beat the lower union of GA, we hybridize the GA with the swarm intelligence strategy called Glowworm Swarm Optimization to enhance the performance.

GSO is a novel method of swarm intelligence based algorithm for optimizing numerous capacities. This algorithm has gotten to be one of the dynamic examination zones of swarm intelligence. GSO algorithm is ordinarily utilized for capacity advancement issues. While utilizing the GSO to take care of the capacity advancement issues, a swarm of glowworms are arbitrarily disseminated in the pursuit space of article capacities. As needs be, these glowworms convey a luminescent amount called luciferin alongside them and they have their own decision domain. A glowworm *i* considers another glowworm *j* as its neighbor if *j* is inside of the neighborhood scope of *i* and the luciferin level of *j* is higher than that of *i*. Specifically, the neighborhood is characterized as a nearby decision domain that has a variable neighborhood range r_d^i limited by a spiral sensor range $r_s (0 < r_d^i \le r_s)$

Every glowworm chooses, utilizing a probabilistic instrument, a neighbor that has a luciferin esteem higher than its own particular and moves toward it. That is, glowworms are pulled in to neighbors that gleam brighter [32]. The glowworms discharge a light whose power is corresponding to the related luciferin and associate with other glowworms inside of a variable neighborhood. The glowworms' luciferin power is identified with the fitness of their present areas. The higher the force of luciferin, the better the area of glowworm, in other words, the glowworm represents a decent target esteem. Otherwise, the objective worth is poor. GSO algorithm to advance the multi-modular capacity incorporate five noteworthy steps [33]:

1) Each sparkle worm i encodes target capacity esteem J (xi(t)) at its present area xi(t) into a luciferin esteem li(t)

2) Constructing neighborhood set (t) ;

3) Each glowworm *i* compute moves toward *j* probability (t) ;

4) Select the moving articles $j*$, and ascertain the new area (t+1), *s* is the moving step;

5) Update the sweep of the dynamic decision domain as in equations (3 to 6):

$$l_i(t) = (1-\rho) l_i(t-1) + \gamma J(x_i(t)) \quad (3)$$

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)} \quad (4)$$

$$x_i(t+1) = x_i(t) + s * \left(\frac{x_j(t) - x_i(t)}{\| x_j(t) - x_i(t) \|} \right) \quad (5)$$

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta - | N_i(t) |)\}\} \quad (6)$$

IV. RESULTS AND DISCUSSION

The KC1 Dataset is used for the performance evaluation of the proposed technique; 2107 samples was used of which 1391 samples are used as training set and 716 samples are used for testing. The precision, recall and f measure are calculated with defect and defect free using three techniques and the results compared. The table 3 shows the results of classification accuracy. Table 3 to 6 and Fig. 1 to 4 shows the results of classification accuracy, precision, recall and F measure respectively.

Table 3 Classification Accuracy

Techniques	Classification Accuracy
MLP with back propagation training	0.925
MLP with proposed GSO training	0.9364
MLP with GSO-GA training	0.9734

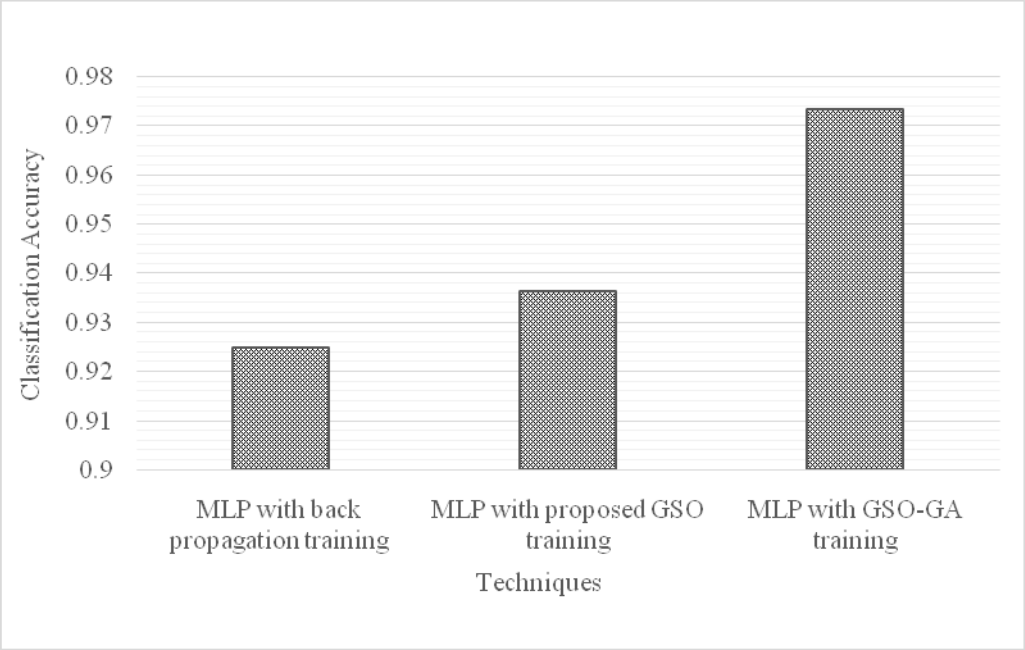

Figure 1 Classification Accuracy

It is observed from table 3 and fig. 1, that the proposed MLP with hybrid GSO-GA training increased classification accuracy by 3.87% when compared with MLP with proposed GSO training.

Table 4 Precision

Techniques	Precision for defect	Precision for no defect
MLP with back propagation training	0.835	0.9344
MLP with proposed GSO training	0.8692	0.944
MLP with GSO-GA training	0.8933	0.9867

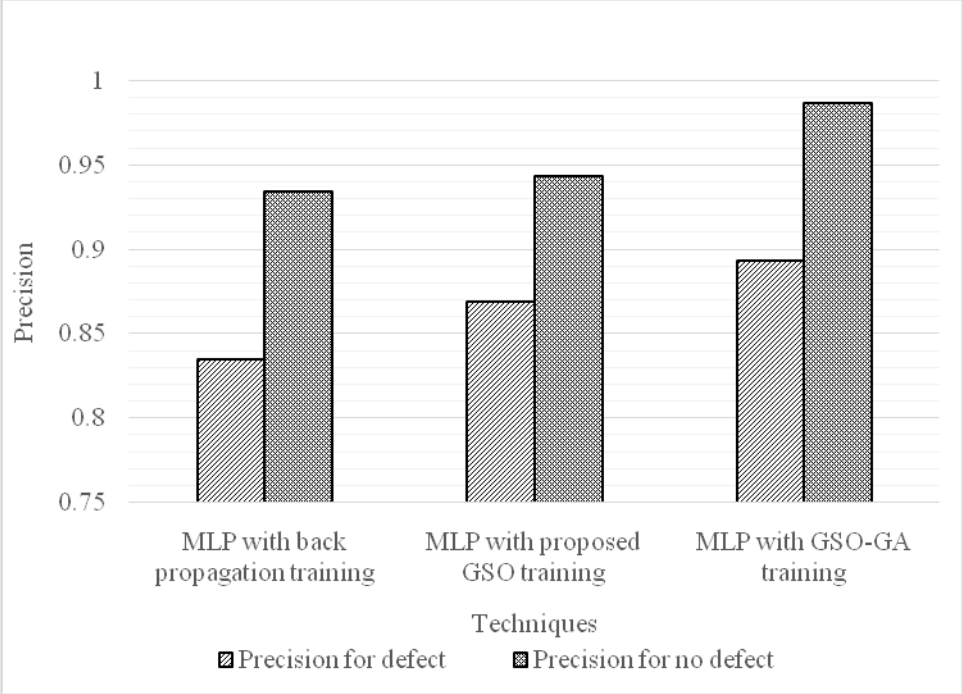

Figure 2 Precision

It is observed from the table 4 and fig. 2, that the proposed MLP with hybrid GSO-GA training increased precision by 2.73% and 4.42% with defect and no defect respectively when compared with MLP with proposed GSO training.

Table 5 Recall

Techniques	Recall for defect	Recall for no defect
MLP with back propagation training	0.5719	0.9818
MLP with proposed GSO training	0.637	0.9846
MLP with GSO-GA training	0.9178	0.9824

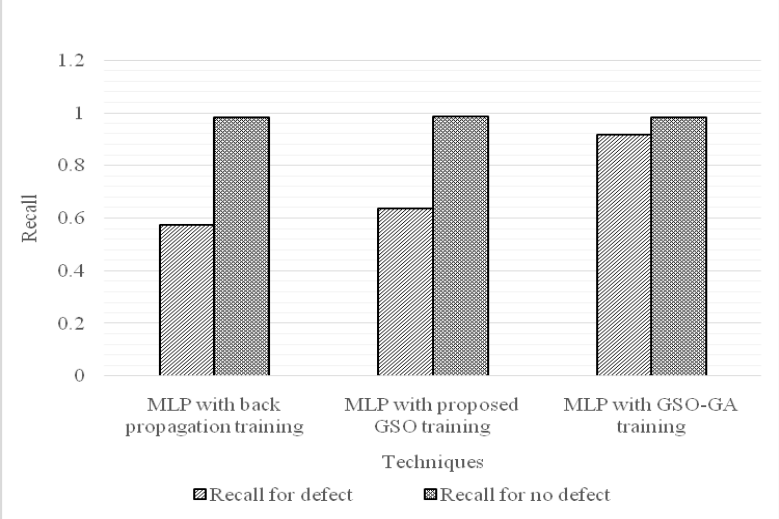

Figure 3 Recall

It is observed from the table 5 and fig. 3, that the proposed MLP with hybrid GSO-GA training increased recall by 36.12% and 0.22% with defect and no defect respectively when compared with MLP with proposed GSO training.

Table 6 F Measure

Techniques	F measure for defect	F measure for no defect
MLP with back propagation training	0.9575	0.6788
MLP with proposed GSO training	0.9639	0.7352
MLP with GSO-GA training	0.9845	0.9054

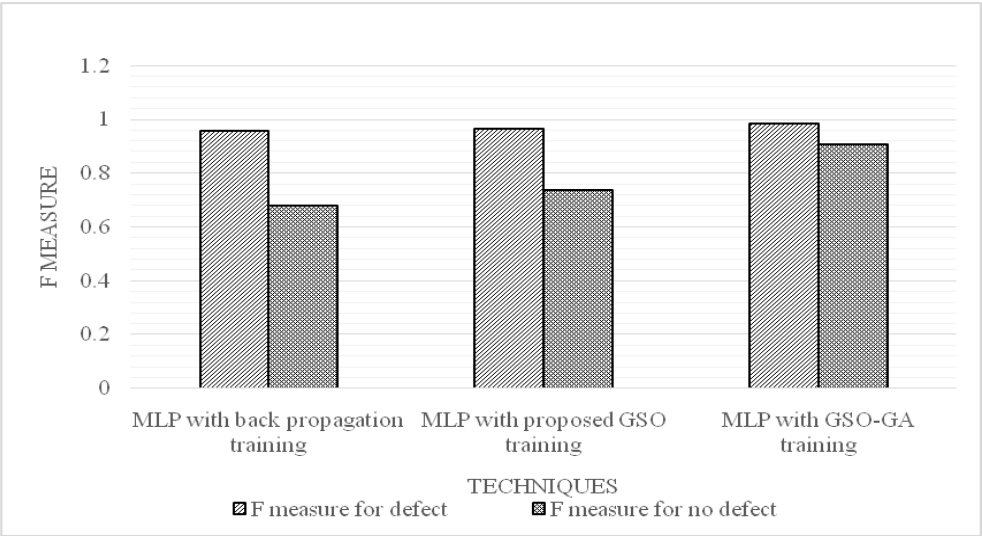

Figure4 F Measure

It is observed from the table 6 and fig. 4, that the proposed MLP with hybrid GSO-GA training increased f measure by 2.11% and 20.75% with defect and no defect respectively when compared with MLP with proposed GSO training.

V. CONCLUSION

In software development, parcel of defects would rise amid the development process. It is a false notion to trust that defects get infused to start with of the cycle and are evacuated through whatever remains of the development process. In previous work, we utilized MLP NN with GA for the defect prediction. To defeat the poor union of GA, an enhanced Glow Swarm Optimization (GSO) algorithm to prepare the MLP NN was proposed. The hybrid algorithm can beat the issue that it's troublesome for conventional algorithms. The outcomes demonstrated that the new hybrid GSO-GA enhanced the classification accuracy, precision, recall and f measure when compared to other methods.

REFERENCES

[1] Suma, V., & Nair, T. R,Defect Management Strategies in Software Development. arXiv preprint arXiv:1209.5573. 2012, 379-404.

[2] Ahmad, K., &Varshney, N. On minimizing software defects during new product development using enhanced preventive approach. International Journal of Soft Computing and Engineering, 2(5), 2012, 9-12.

[3] Khan, H. A,Establishing a defect management process model for software quality improvement. International Journal of Future Computer and Communication, 2(6), 2013, 585.

[4] Mittal, A., &kumar Dubey, S,Defect Handling in Software Metrics.International Journal of Advanced Research in Computer and Communication Engineering, 1(3), 2012, 167-170.

[5] Kumaresh, S., &Baskaran, R,Defect analysis and prevention for software process quality improvement. International Journal of Computer Applications, 8(7), 2010, 1-5.

[6] Paramshetti, P., &Phalke, D. A,Survey On Software Defect Prediction Using Machine Learning Techniques. International Journal Of Science And Research, 3(12), 2014, 1394-1397.

[7] Rana, R, Software defect prediction techniques in automotive domain: evaluation, selection and adoption (Doctoral dissertation, University of Gothenburg), 2015.

[8] Nam, J. **Survey on software defect prediction**. Department of Compter Science and Engineerning, The Hong Kong University of Science and Technology, Tech. Rep, 2014.

[9] Zheng, J,Cost-sensitive boosting neural networks for software defect prediction. Expert Systems with Applications, 37(6), 2010, 4537-4543.

[10] Rawat, M. S., & Dubey, S. K,Software defect prediction models for quality improvement: a literature study. IJCSI International Journal of Computer Science Issues, 9(5), 2012, 288-296.

[11] Aleem, S., Capretz, L. F., & Ahmed, F,Benchmarking Machine Learning Technologies for Software Defect Detection. arXiv preprint arXiv:1506.07563, 2015.

[12] Agarwal, S., &Tomar, D, Prediction of Software Defects using Twin Support Vector Machine. In Information Systems and Computer Networks (ISCON), 2014, 128-132). IEEE.

[13] Malhotra, R., Pritam, N., & Singh, Y,On the applicability of evolutionary computation for software defect prediction. InAdvances in Computing, Communications and Informatics (ICACCI), 2014, 2249-2257.

[14] Okumoto, K, Software defect prediction based on stability test data. In Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2011,385-387.

[15] Shepperd, M., Bowes, D., & Hall, T,Researcher bias: The use of machine learning in software defect prediction. Software Engineering, IEEE Transactions on, 40(6), 2014, 603-616.

[16] Chen, Y., Shen, X. H., Du, P., & Ge, B, Research on software defect prediction based on data mining. In Computer and Automation Engineering (ICCAE), 2010 563-567.

[17] Islam, R., &Sakib, K. A Package Based Clustering for enhancing software defect prediction accuracy. In Computer and Information Technology (ICCIT), 2014,81-86.

[18] Mausa, G., Grbac, T. G., & Basic, B. D. Software defect prediction with bug-code analyzer-a data collection tool demo. InSoftware, Telecommunications and Computer Networks (SoftCOM), 2014,425-426.

[19] Pushphavathi, T. P., Suma, V., &Ramaswamy, V. A novel method for software defect prediction: Hybrid of FCM and random forest. In Electronics and Communication Systems (ICECS), 2014,1-5.

[20] Lu, H., Kocaguneli, E., &Cukic, B. Defect prediction between software versions with active learning and dimensionality reduction. In Software Reliability Engineering (ISSRE), 2014,312-322.

[21] Wang, J., Shen, B., & Chen, Y. Compressed c4. 5 models for software defect prediction. In Quality Software (QSIC), 2012,13-16.

[22] Shan, C., Chen, B., Hu, C., Xue, J., & Li, N. Software defect prediction model based on LLE and SVM. In Communications Security Conference (CSC 2014), 2014, 1-5.

[23] Gray, D., Bowes, D., Davey, N., Sun, Y., & Christianson, B. Software defect prediction using static code metrics underestimates defect-proneness. In Neural Networks (IJCNN), 2010,1-7.

[24] Kutlubay, O., &Bener, A. A Machine Learning Based Model for Software Defect Prediction. workingpaer, Boaziçi University, Computer Engineering Department, 2005.

[25] Sahana, D. C. Software Defect Prediction Based on Classication Rule Mining (Doctoral dissertation), 2013.

[26] Singh, M., &Salaria, D. S. Software defect prediction tool based on Neural Network. International Journal of Computer Applications, 70(22), 2013, 22-31.

[27] M.V.P Chandra Sekhara Rao, Dr. B RavendraBabu, Aparna Chaparala, Dr. A Damodaram. An improved Multiperceptron Neural network model to classify Software defects. International journal of computer science and information security, 9(2), 2011, 124-128.

[28] Zhang, Y., Lo, D., Xia, X., & Sun, J. An empirical study of classifier combination for cross-project defect prediction. In Computer Software and Applications Conference (COMPSAC), 2015,264-269.

[29] Gao, K., Khoshgoftaar, T. M., Wang, H., &Seliya, N. Choosing software metrics for defect prediction: an investigation on feature selection techniques. Software: Practice and Experience, 41(5), 2011, 579-606.

[30] Terfaloaga, I. M. Solving Systems of Equations with Techniques from Artificial Intelligence. AnaleleUniversitatii'EftimieMurgu', 22(1) 2015, 324-340.

[31] Kaswan, K. S., Choudhary, S., & Sharma, K. Software Reliability Modeling using Soft Computing Techniques: Critical Review. Journal of Information Technology & Software Engineering, 2015.

[32] Zhao, G., Zhou, Y., & Wang, Y. The glowworm swarm optimization algorithm with local search operator. Journal of Information & Computational Science, 9(5), 2012, 1299-1308.

[33] Sathishkumar, K., Thiagarasu, V., &Balamurugan, E. A study on microarray gene expression data and clustering analysis. International journal of engineering sciences & research technology 4(1), 2013, 188-196.