Hindawi
Scientific Programming
Volume 2022, Article ID 2117339, 10 pages
https://doi.org/10.1155/2022/2117339

Review Article

Software Defect Prediction Using Artificial Neural Networks: A Systematic Literature Review

Muhammad Adnan Khan,[1] Nouh Sabri Elmitwally,[2] Sagheer Abbas,[3] Shabib Aftab,[3,4] Munir Ahmad,[3] Muhammad Fayaz ,[5] and Faheem Khan[6]

[1]*Pattern Recognition and Machine Learning Lab, Department of Software, Gachon University, Seongnam, 13120, Republic of Korea*
[2]*Faculty of Computing, Engineering and the Built Environment at Birmingham City University, Birmingham, UK*
[3]*School of Computer Science, National College of Business Administration and Economics, Lahore 54000, Pakistan*
[4]*Department of Computer Science, Virtual University of Pakistan, Lahore 54000, Pakistan*
[5]*Department of Computer Science, University of Central Asia, Naryn 722918, Kyrgyzstan*
[6]*Department of Computer Engineering, Gachon University, Seongnam 13120, Republic of Korea*

Correspondence should be addressed to Muhammad Fayaz; muhammad.fayaz@ucentralasia.org

Received 18 October 2021; Revised 12 April 2022; Accepted 3 May 2022; Published 12 May 2022

Academic Editor: Shah Nazir

The demand for automated online software systems is increasing day by day, which triggered the need for high-quality and maintainable softwares at lower cost. Software defect prediction is one of the crucial tasks of the quality assurance process which improves the quality at lower cost by reducing the overall testing and maintenance efforts. Early detection of defects in the software development life cycle (SDLC) leads to the early corrections and ultimately timely delivery of maintainable software, which satisfies the customer and makes him confident towards the development team. In the last decade, many machine learning-based approaches for software defect prediction have been proposed to achieve the higher accuracy. Artificial Neural Network (ANN) is considered as one of the widely used machine learning techniques, which is included in most of the proposed defect prediction frameworks and models. This research provides a critical analysis of the latest literature, published from year 2015 to 2018 on the use of Artificial Neural Networks for software defect prediction. In this study, a systematic research process is followed to extract the literature from three widely used digital libraries including IEEE, Elsevier, and Springer, and then after following a thorough process, 8 most relevant research publications are selected for critical review. This study will serve the researchers by exploring the current trends in software defect prediction with the focus on ANNs and will also provide a baseline for future innovations, comparisons, and reviews.

1. Introduction

Due to the exponent increase in the use of computer systems since the last two decades, almost everything around us has been changed and the world is getting digitalized even faster than we thought. More software systems are being developed nowadays to automate the processes and procedures which are related to our daily life. Due to these automate procedures, our lives are now more comfortable than before. A software with compromised quality can produce wrong and unexpected results [1]. Nowadays, due to growing complexities in modern software systems, it is becoming difficult to produce high-quality software product at lower cost. This issue can be solved by using the techniques of software defect prediction. Software defect prediction is one of the famous and dynamic research areas in the software engineering domain [2] and is also considered as a key activity in the quality assurance process. The defect prediction approaches have accomplished the significant acceptance in the software industry in the last two decades. This activity can improve the software quality by indicating the software modules in advance where faults are more likely to occur [3, 4]. Prediction and prevention of software defects at the initial stages of software development can reduce the overall development time and cost by limiting the testing efforts [5, 6]. Artificial Neural Network (ANN) is a widely accepted supervised learning approach to deal with the prediction problems in multiple domains of software engineering such as effort estimation, cost estimation, and defect prediction [7]. The structure of ANN is divided into three layers: (1) input layer, (2) output layer, and (3) hidden layer(s). A connection exists from the nodes in the input layer with the nodes in the hidden layer and then from the nodes of the hidden layer with the nodes of the output layer. The input data are entered into the neural network through the input layer [8, 9]. The classifiers which belong to the supervised machine learning group first need a data set (training data) with predefined output class for training. The data set includes various features which are categorized as dependent and independent features. The dependent feature is one which is going to be predicted, also known as the output class. Other features except the output class are known as independent features. During training, the supervised classifier extracts the hidden patterns and relations among the dependent and independent features and develops a classification model. After training, a data with unknown output (test data) class is given to the classifier which is then predicted by the classification model on behalf of extracted patterns and rules from training data [10–14]. The task of software defect prediction is achieved by classifying a particular software instance (method, class, module, file, and package) as defective or nondefective. The data set used for software defect prediction includes the historical software defect data collected from previous releases of same project (or in some cases from other projects), which consists of a number of features or attributes, called quality metrics. The defect prediction approach can help the development team to effectively utilize its testing effort to save the time and cost. Suppose there are 50 modules in a software from which 30 are already developed and tested whereas the remaining 20 are in developing stage. Now, defect prediction techniques can be used by using the data of 30 tested modules to predict the defects in the remaining 20 modules. After the identification of possible defective modules, testing practices can only be performed on those modules which are predicted as defective. In other case, the defective modules can be focused with multiple testing iterations as compared to those which are predicted as nondefective. ANN is one of the widely used machine learning techniques for software defect prediction. The nonlinear nature of ANN is capable to effectively extract the hidden patterns in the historical defect data set [8, 15]. ANN with its multilayer optimizable structure can provide the promising results on the detection of defect-prone software modules as compared to other techniques [16, 17]. This study aims to provide a critical review of the latest literature published from 2015 till 2018 on software defect prediction with the focus on ANNs. Three online libraries are used to extract the literature: IEEE, Elsevier, and Springer. The extracted papers with particular search query are 3937, 2214, and 1280, respectively. However, by following the complete systematic process, 8 most relevant papers were finally selected for in-depth critical review.

2. Related Work

Use of machine learning techniques to solve the prediction problems has been the focus of many researchers since last decade. Most of the researchers have been working to tune the performance of ANN classifier for software defect prediction. They have proposed various models and frameworks either by optimizing the structure of ANN or by integrating it with other preprocessing, machine learning, or statistical techniques. This section highlights some of the related studies which used ANN and other classifiers for software defect prediction. Researchers in [18] proposed a defect prediction model based on an improved ANN. They compared the performance of proposed technique by using the data sets from MDP (Metrics Data Program). The proposed improved ANN model showed better results while comparing with other techniques such as Bayesian Logistic Regression, Random Tree, and Classification and Regression Trees (CART). In [19], researchers presented an ANN-based tool with Levenberg–Marquardt algorithm to predict the software defects at the early stage of SDLC. The used data sets in experiments are collected from PROMISE repository and consisted of CKOO (Chidamber and Kemerer Object-Oriented) metrics. This study reflected that Levenberg–Marquardt-based ANN provides better results as compared to other techniques such as Polynomial Function-Based Neural Networks, Linear Function-Based Neural Network, and Quadratic Function-Based Neural Network. Researchers in [20] discussed that a software module which is in developing stage and have same metrics like any other module which is defective and developed in the same environment might have same levels of defects. To predict the defects, the authors have developed an Adaptive Resonance Neural Network having 29 input nodes (metrics) and two output nodes (defective or nondefective). The developed model is trained with data set obtained from PROMISE repository. The results reflected the improved Recall measure. Researchers in [21] compared various classification models such as Naïve Bayes, Decision Tree, and Random Forest to detect the defects in latest version of software by using the data of seven old versions of same software. According to the results, Random Forest outperformed others and showed highest predictive power while using the AUC curve as performance measure. In [22], researchers explored that the ANN reflects the more accurate results as compared to Fuzzy logic based model. Moreover according to these researchers, ANNs have the tendency to be used effectively in hybrid approach for large data set. Performance analysis of these models is performed by Magnitude of Relative Error (MMRE) and Balanced Mean Magnitude of Relative Error (BMMRE). Researchers in [23] presented a novel technique named P-SVM to detect the bugs from software at early stages of software development. P-SVM is the combination of PSO and SVM. The proposed technique outperformed Backpropagation-based Neural Network, SVM Model, and GA-SVM model. The researchers also claimed this mode to be the more robust. Researchers in [24] explored that the Multi-Variant Gauss Naïve Bayes (MVGNB) technique has tendency to outperform all other kind of classifiers. They experimented with J48 to analyze the performance of MVGNB and found that it is more effective technique to predict the defects at an early stage of software development. In [25], researchers discussed that the hybrid model of classifiers always reflects more accurate results than any of the single classifier. They also elaborated that the use of hybrid technique to select the attributes can further improve the accuracy. In this research, five classifiers are compared: k Nearest Neighbors (KNN), KStar (k ∗), Locally Weighted Learning (LWL), Random Forest (RF), and Random Tree (RT). According to results, LWL outperformed the others with 92.23% accuracy. Researchers in [26] implemented the multilevel preprocessing and performed attribute selection twice followed by instance filtering thrice. Preprocessing is performed with four versions of KNN classifiers: KNN-LWL, K ∗ r, KNN, and IB1 classifier. The results are compared with RT, RF, and nonnested generalized classifier. Performance is analyzed by means of Accuracy, Recall, Area Under Curve (AUC), and precision. Results reflected that performance of RF outperformed the others after double preprocessing. In [27], researchers integrated the SVM and Auto Regression Integrated Moving Average (ARIMA) for software defect prediction. They analyzed that proposed technique can perform better with lower error rate in comparison with other conventional models. In [28], researchers combined three classification algorithms, NB, Voting Feature Interval, and ANN, for software defect prediction and used five data sets for experiment. They concluded that the integration of these classifiers showed better performance during the prediction of software defects especially for embedded system. Researchers in [29] performed an analysis on software defect prediction by using various machine learning techniques including Logistic Regression (LR), RF, Decision Tree (DT), Association Rule Mining, Naïve Bayes (NB), ANN, SVM, Genetic Algorithm, and Fuzzy Programming. They discussed that machine learning techniques can be helpful for the detection and removal of minor defects. In [30], researchers performed a comparative analysis of 13 machine learning algorithms and found that B, ANN, and Instance-based learning showed better results than others. According to [31], various studies on software defect prediction represent the data sets which are used for experiments, methods which are used for prediction, and frameworks which are the collection of multiple methods. The aim of research on all of these elements related to development activities is to improve the quality of end product. The literature review presented in this study analyzes and presents the data sets, research trends, methods, and frameworks used in the research of software defect prediction between year 2000 and 2013. Researchers in [32] highlighted that different software metrics are in use for defect prediction modeling; however, the small subset from all of the available metrics would not only increase the accuracy of prediction system but also reduce the cost. The researchers have used the Bayesian networks to select probabilistic relationships between metrics and the defect prone modules. Moreover, they have extracted two additional metrics including "number of developers (NOD) and lack of coding quality (LOCQ)" which contributed towards the detection of defect-prone software modules.

3. Research Protocol

This study focuses on the latest research trends regarding the use of ANNs for software defect prediction. The purpose of the systematic literature review is to deeply investigate the selected research articles, spanned over a specific time period as explained by [33]. A research protocol has to be followed to formalize the systematic review process. This process includes various steps to be followed in a particular sequence. In this study, the latest research articles published in four years (2015 till 2018) are focused. To select the high-quality research papers, a systematic research process is followed as explained by [34, 35]. Moreover, the detailed guidelines are also taken from [36–42]. The systematic research process followed by this study consists of 9 steps as shown in Figure 1.

3.1. Research Questions. Research questions reflect the true objectives of a systematic literature review SLR. During the critical review of the most relevant extracted articles, we will try to find the answers to these questions.

Research questions of this SLR are as follows:

RQ1: which technique of ANN is proposed/used for software defect prediction?

RQ2: how the performance of the proposed/used technique of ANN is evaluated?

RQ3: with which technique(s) the proposed/used technique is compared?

RQ4: which data set is used for the performance evaluation of used technique?

RQ5: what is the source of the used data set?

RQ6: which tool is used for the experiment and simulation of results?

3.2. Selection of the Keywords. Extraction of keywords from the research questions is considered a basic building block of systematic research process. These keywords will be arranged systematically to create a query string, which will extract the articles.

Keywords extracted from research questions are as follows: Software, Application, System, Fault, Defect, Error, Prediction, Prone, Probability, Assessment, Detection, Estimation, Classification, Artificial Neural Network, Neural Network, and ANN.

3.3. Formation of Query String. Query string is formed by using various combinations of selected keywords and will be used to extract the relevant research articles from the selected libraries.

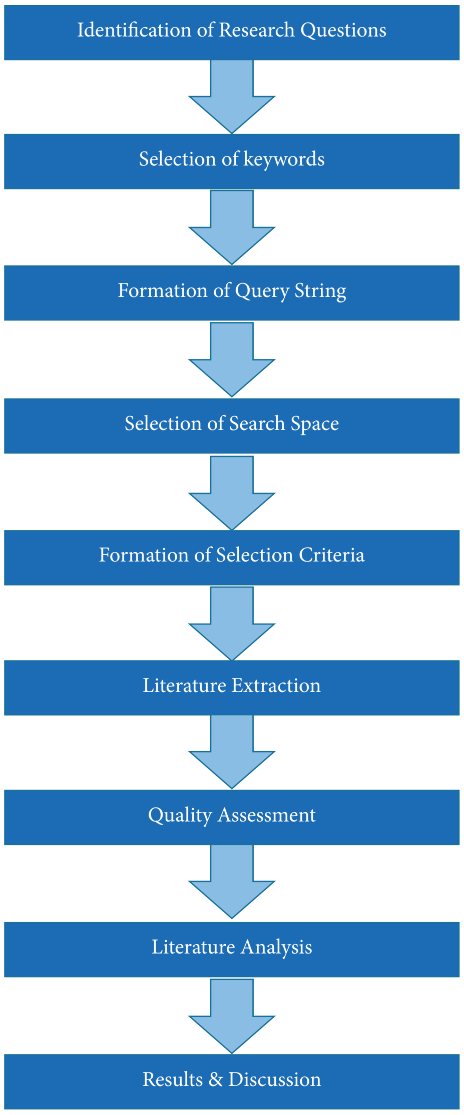
Identification of Research Questions
Selection of keywords
Formation of Query String
Selection of Search Space
Formation of Selection Criteria
Literature Extraction
Quality Assessment
Literature Analysis
Results & Discussion

Figure 1: Steps of SLR.

The following search query is finalized with the extracted keywords.

((“Software” OR “Application” OR “Systems”) AND (“Fault” OR “Defect” OR “Error”) AND (“Prediction” OR “Prone” OR “Probability” OR “Assessment” OR “Detection” OR “Estimation” OR “Classification”) AND (“Artificial Neural Networks” OR “Neural Network” OR “ANN”))

3.4. Selection of Search Space. Three well-known online search libraries are selected to extract the relevant research literature: IEEE, Elsevier, and Springer. All three selected libraries have different characteristics and options to search the relevant material. Therefore, few adjustments are made in the query string to extract more relevant and appropriate literature from these libraries. Moreover, the query had to be searched for multiple times with different combinations of selected keywords.

3.5. Formation of Selection Criteria. This section of systematic research process deals with the selection of relevant articles from the extracted literature. For this purpose, a particular selection criterion is formed which consists of IC (inclusion criteria) and EC (exclusion criteria).

3.5.1. Inclusion Criteria (IC). Inclusion criteria is formed with the following rules.

IC1: papers published from year 2015 till 2018 date

IC2: papers that proposed/used any technique(s) of ANN for Software Defect Prediction

IC3: papers that used hybrid model for software defect prediction, which includes ANN

IC4: papers that used any data set to implement the proposed technique/model

IC5: papers that used other technique(s) in comparison with proposed/used technique to evaluate the performance

3.5.2. Exclusion Criteria (EC). Exclusion criteria are formed with the following rules:

EC1: papers which are not in English

EC2: papers published before 2015 or after 2018

EC3: papers which did not target the software defect prediction

EC4: papers which did not use ANN in proposed/used technique

EC5: papers that used Hybrid Model, which does not include ANN

EC6: papers which did not evaluate the performance of proposed/used techniques

3.6. Literature Extraction. Literature extraction is also considered as one of the crucial stages of systematic research process as it deals with the step-by-step process of short-listing the most appropriate research articles for critical review as shown in Figure 2.

After applying the search process, 8 most relevant research articles were short-listed as provided in Table 1 where C.P stands for Conference Paper and J.P stands for Journal paper.

3.7. Quality Assessment. In this research, quality standards were maintained throughout the systematic research process. The crucial parameters for quality assessment which are followed during the selection of relevant literature make this study a true reflection of current research.

Following parameters are considered for the selection of literature:

(i) The scientific libraries with highest ratings are selected to extract the appropriate research material

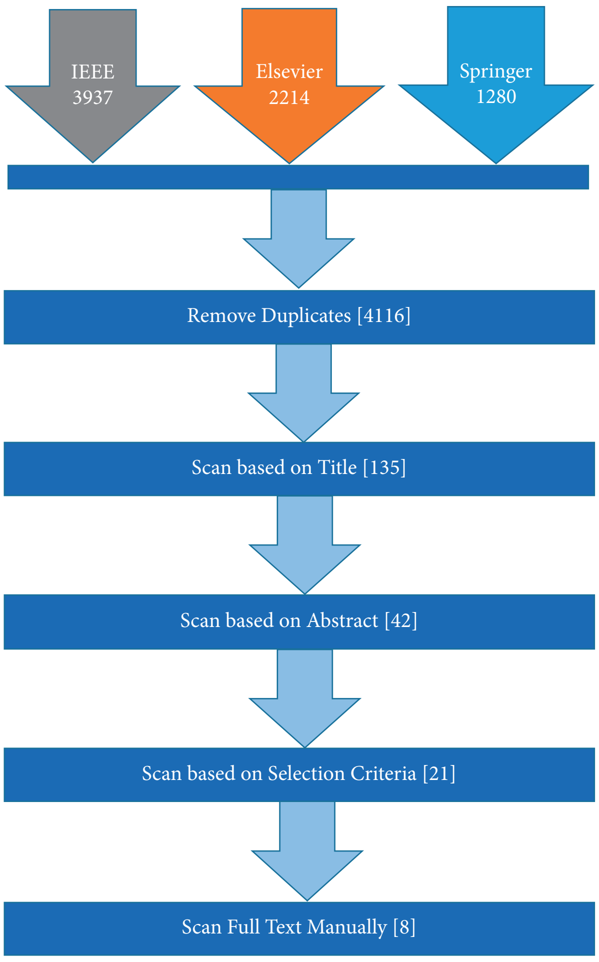
IEEE
3937
Elsevier
2214
Springer
1280
Remove Duplicates [4116]
Scan based on Title [135]
Scan based on Abstract [42]
Scan based on Selection Criteria [21]
Scan Full Text Manually [8]

Figure 2: Search process.

(ii) Selection process is unbiased

(iii) All the steps of SLR are followed in the true sense

3.8. Literature Analysis. This stage deals with the critical review of the shortlisted papers from the stage "Literature extraction." This stage will be discussed in detail in Section 4.

3.9. Results and Discussion. This stage deals with the answers to the identified research questions. This stage will be discussed in detail in Section 5 of this study.

4. Literature Analysis

4.1. Software Defect Prediction via Convolutional Neural Network. Researchers in [43] presented a framework for software defect prediction called DP-CNN (Defect Prediction via Convolutional Neural Network) which used Convolutional Neural Network (CNN) to generate discriminative features from the programs' Abstract Syntax Trees (ASTs), which preserves semantic and structural information of the source code. Moreover, to further tune the performance, the proposed technique used word embedding to encode the tokens extracted from ASTs which aids CNN to learn the semantics of source code. Finally, the CNN-learned features are integrated with traditional defect prediction features to take the advantages of both nonlinear features and hand-crafted features. The proposed DP-CNN framework is evaluated on seven open-source Java projects from Tera-PROMISE Repository. From the selected projects: version numbers, class name of each file, and the defect label for each source file are provided. The source code of each file can be extracted from GitHub by using the version numbers and class names. Performance of proposed framework is measured in terms of F-measure. Performance is compared with other baseline methods such as Deep Belief Network (DBN), Traditional Logistic Regression (TLR) classifier, DBN+, and CNN. According to overall results, the proposed DP-CNN outperformed all other baseline methods especially state-of-the-art DBN-based methods and traditional features-based methods by 12% and 16%, respectively.

4.2. Comparison of Backpropagation Training Algorithms for Software Defect Prediction. Researchers in [44] developed Multilayer Feed Forward Neural Network using three standard backpropagation training algorithms: Levenberg–Marquardt (LM), Resilient Backpropagation (RP), and Bayesian Regularization (BR) Backpropagation in MATLAB. The network architecture consisted of single hidden layer with 10 neurons. The constructed models were compared on seven data sets from the PROMISE repository to predict the software defects. Performance evaluation is performed by using MSE, RMSE R2, and other parameters of the confusion matrix. The results showed that the BR training function performed well on six out of seven data sets with a minimum MSE of 4.91; however, RP performed well only for one data set PC2 with an MSE of 0.02. The LM function did not achieve minimum MSE in any of the data sets. Moreover, according to overall results from other performance-related parameters such as R2, Accuracy, Recall, and False Negative Rate, the BR outperformed the LM and RP. This research has highlighted the application of search-based optimization algorithm which is inspired by the nature, in the domain of software engineering. Moreover, according to the authors, the project managers can prioritize the performance measures on the basis of context and criticality of software, and then, according to goals and available resources, the appropriate training function can be selected.

4.3. Software Defect Prediction via Transfer Learning-Based Neural Network. In [45], researchers presented Transfer Component Analysis Neural Network (TCANN) for effective software defect prediction. The proposed approach targeted to solve three problems in software defect prediction domain: (1) Noisy Data, Class Imbalance, and Transfer Learning among Cross-Projects. TCANN consisted of three modules for the mentioned three issues: interquartile range (IQR)-based method is used to remove the noise, Dynamic Sampling Neural Network (DSNN) is used to take care of the class imbalance issue of training data, and Transfer Component Analysis (TCA) technique is used to reduce the differences of feature distribution among

Table 1: Selected studies for review.

Ref	Type	Name	Publisher	Year	Cited by
1.	C.P	International Conference on Software Quality, Reliability & Security	IEEE	2017	225
2.	C.P	2nd International Conference on Contemporary Computing and Informatics	IEEE	2016	18
3.	C.P	1st International Conference on Reliability Systems Engineering	IEEE	2015	19
4.	J.P	Cognitive Systems Research	Elsevier	2018	15
5.	J.P	Applied Soft Computing	Elsevier	2015	183
6.	J.P	Applied Soft Computing	Elsevier	2015	69
7.	J.P	Cluster Computing	Springer	2018	44
8.	J.P	Cluster Computing	Springer	2018	80

source and target data. Performance of the proposed method is measured in terms of Precision, Recall, F-measure, and AUC are used. Performance analysis is performed from two aspects of defect prediction, within-project and cross-project. Within-project evaluation for each data set, 70% data are selected for training, and the remaining 30% are used for test purpose, and finally, the results are compared with the real value. Along with ANN, noise reduction technique is used; however, TCA is not used as training and testing both instances are from the same project. In cross-project validation, all 26 combinations of AEEEM and ReLink data sets are used along with TCA. This experiment included the comparison with other modern techniques such as TCA and TCA + etc. The proposed approach reflected the good performance for software defect prediction in both within-project and cross-projects.

4.4. Cognitive Deep Neural Networks Prediction Method for Software Fault Tendency Module Based on Bound Particle Swarm Optimization. Researchers in [46] presented a Deep Neural Network (DNN)-based BPSO (Bound Particle Swarm Optimization) dimensionality reduction technique to predict the software defects. In the proposed approach, BPSO is used to reduce the dimensionality of measurement space, whereas DNN predicted the potential defective software modules. Traditional particle swarm algorithm is optimized by introducing a field theory for effective searching to ensure the global convergence. Preprocessing method is used to keep the values of used software parameters in a specific limit for better training and prediction results. Data sets used to evaluate the performance of the proposed approach are taken from the software programs of NASA, and the performance is analyzed in terms of AUC and AUC mean value, no of selected software indexes, and Mean Calculation Time. The accuracy is compared with various conventional and modern proposed methods, which showed that the proposed approach performed well and brought the impressive results with less software indexes in short time. The authors concluded that the proposed feature reduction technique can directly guide the developers to focus on such software modules which are more likely to have defects.

4.5. Software Defect Prediction Using Cost-Sensitive Neural Network. Researchers in [16] proposed a novel approach for software defect prediction in the form of an integrated technique which includes traditional ANN and the Artificial Bee Colony (ABC) algorithm. To extract the optimal weights, ABC algorithm is used for training the ANN. In this technique, the False Positive Rate (FPR) and False Negative Rate (FNR) are multiplied by parametric cost coefficients, and this process is governed by the ABC algorithm. Software defect data contain the class imbalance issue due to the skewed distribution of defective and nondefective parts of software; therefore, the conventional error functions bring unbalanced FPR and FNR output. The proposed approach was evaluated by applying on five publicly available data sets from NASA Metrics Data Program repository. Performance is measured in terms of Accuracy, Probability of Detection, Probability of False Alarm, Balance, AUC, and Normalized Expected Cost of Misclassification (NECM). In this research, Correlation-based Feature Selection (CFS) method is used in WEKA tool to extract the subset of available features instead of considering all quality attributes as input. To remove the unbiasedness from the results, the data set was shuffled and the algorithm was executed 10 times with the use of n-fold cross-validation in each iteration. Performance of the proposed approach is compared in terms of AUC with other classifiers such as NB, RF, DT, Immunes, and AIRS.

4.6. Prediction Approach of Software Fault-Proneness Based on Hybrid Artificial Neural Network and Quantum Particle Swarm Optimization. Researchers in [47] presented an approach which integrates Hybrid ANN and Quantum Particle Swarm Optimization (QPSO) to detect the software modules in which defects are likely to occur. QPSO is used to reduce the dimensionality of software quality features, and ANN classified the modules into defective or nondefective. The proposed technique has highlighted the correlation among defective/nondefective software modules and software metrics. Moreover, the implementation of QPSO is simple than PSO as it has only one controlling parameter as compared to three in PSO. The experiment results reflected that QPSO due to its feature reduction technique can simplify the structure of ANN and can perform well than other traditional and proposed prediction techniques. The selected metrics with the proposed approach also direct the developers to focus in such modules which are more likely to be defective, which can ultimately reduce the development cost of the software. Data sets used to evaluate the performance of the proposed approach are taken from the software programs of NASA, and the performance is analyzed in terms of AUC and AUC mean value, no of selected software indexes, and Mean Calculation Time. The accuracy is compared with various conventional and proposed methods, which showed that the proposed approach performed well and brought the impressive results with less software indexes in short time. Preprocessing method is used to keep the values of used software parameters in a specific limit for better training and prediction results.

4.7. Software Defect Prediction Techniques Using Metrics Based on Neural Network Classifier. The authors in [8] proposed an approach for software defect prediction by integrating the feature reduction technique and ANN. Feature reduction is performed by Principal Component Analysis (PCA) technique. In proposed approach, PCA is improved by integrating the maximum-likelihood estimation to reduce the error in data reconstruction. The modified PCA approach improved the learning performance of ANN classifier due to nonredundant data as if the machine learning technique fails to identify the statistical consistency, then it could affect the learning performance. The proposed approach is tested on NASA software data sets in MATLAB tool, and the performance is evaluated in terms of Precision, Sensitivity, Specificity, Recall, F-measure, Classification Accuracy, etc. The performance of proposed technique is compared with other state-of-art techniques including k-NN, SVM, L-SVM, LS-SVM NB, and LDA. Comparative analysis reflected that the proposed technique outperformed other techniques.

4.8. Deep Neural Network-Based Hybrid Approach for Software Defect Prediction. Researchers in [48] presented a hybrid approach which integrated GA and DNN. GA is used for feature optimization whereas DNN performed the classification task. GA was improved in the proposed approach by including the new method for designing chromosome along with fitness function computation. Moreover, DNN was also included after improvement by using an adaptive auto-encoder which provided a better representation of selected software attributes. The proposed approach is tested on NASA software data sets in MATLAB tool and performance is evaluated in terms of Precision, Sensitivity, Specificity, Recall, F-measure, Classification Accuracy, etc. The performance of the proposed technique is compared with various other state-of-art techniques, and the comparative analysis reflected that the proposed technique outperformed other techniques.

5. Results and Discussion

We have finally selected the 8 most appropriate research papers by following the thorough research methodology, based on the selection criteria, explained in Section 3. These papers were already been discussed with detail in Section 4. This section deals with the extracted answers to the research questions defined earlier.

RQ1: which technique of ANN is proposed/used for software defect prediction?

Researchers in [43] used the CNN which automatically learns the semantic and structural features of programs and then combined those features with traditional hand-crafted features. Researchers in [44] performed a comparative analysis of Levenberg–Marquardt, Resilient backpropagation, and Bayesian Regularization backpropagation training algorithms for software defect prediction. In [45], researchers proposed a technique that consisted of three stages. In 1st stage, IQR-based method is proposed for noise removal in data sets, and 2nd stage deals with the TCA method which is used to reduce the feature distribution differences between source and target data for cross-project prediction. The 3rd' stage used the DSNN to deal with the class imbalance problem of the training data set. In [46], the BPSO algorithm is used for reducing the dimensionality of the measurement space while DNN is used for predicting the fault tendency of software modules. In [16], the ABC algorithm is used to train the traditional ANN for the achievement of optimal weights. In [47], ANN is used for classification and QPSO is applied to reduce dimensionality. Researchers in [8] used the ANN for classification, and an enhanced version of PCA is used for feature reduction. Researchers in [48] used a hybrid approach by integrating GA for feature optimization and DNN for classification.

RQ2: how the performance of the proposed/used technique of ANN is evaluated?

In [43], the researchers used the F-measure as performance measure. Researchers in [44] used MSE, RSME, R Square, Accuracy, Sensitivity, Specificity, False Negative Rate, and False Positive Rate to evaluate the performance. In [45], Precision, Recall, F-measure, and AUC were used. Researchers in [46] used AUC, AUC Mean, and Mean Calculation Time. In [16], AUC, PD, PF, BAL, and ACC were used. In [47], AUC was used. Researchers in [8] used Precision, Recall, F-measure, Accuracy, Sensitivity, and Specificity, whereas in [48], Precision Sensitivity Specificity Recall, F-measure, Accuracy, and AUC were used.

RQ3: with which technique(s), the proposed/used technique is compared?

Researchers in [43] compared the proposed techniques with Traditional DBN and DBN. The research [44] was itself a comparative study. Levenberg–Marquardt, Resilient backpropagation, and Bayesian Regularization backpropagation training algorithms were compared for software defect prediction. In [45], the proposed technique was compared with TCA, TCA+, and VAB-SVM. In [46], the proposed technique was compared with LDA, QDA, LogReg, NB, Bayes Net, LARS, RVM, K-NN, K∗, MLP-1, MLP-2, RBF, SVM, L-SVM, LS-SVM, LP, VP, DT, CART, ADT, LMT, DNN, PSO + DNN, BPSO + SVM, BPSO + DNN, PCA + DNN, and PLS + DNN. Researchers in [16] compared the proposed technique with NB, RF, DT, Immunos, and AIRS. In [47], the researchers compared the proposed technique with the following techniques: LDA, QDA, NB, LogReg, Bayes Net, LARS, RVM, k-NN, K∗, MLP-1, MLP-2, RBF, SVM, L-SVM, LS-SVM, LP, VP, DT, CART, ADT, RndFor, LMT, ANN, PSO + ANN, QPSO + SVM, and QPSO + ANN. In [8], k-NN, SVM, L-SVM, LS-SVM, NB, and LDA were compared with the proposed technique.

Researchers in [48] compared the proposed technique with NB, RF, DT, Immunos, ANN-ABC, Hybrid self-organizing map, SVM, Majority vote, Ant Miner+, ADBBO-RBFNN, NN GAPO + B, DT, and KNN.

RQ4: which data set is used for the performance evaluation of used technique?

The data sets used by [43] are Camel, jEdit, lucene, Xalan, xerces, synapse, and poi. Researchers in [44] used PC1, PC2, PC3, PC4, PC5, KC2, and KC3. In [45], researchers used EQ, JDT, LC, ML, PDE, Safe, Apache, and ZXing. In [46], researchers used PC1, JM1, KC1, and KC3. Researchers in [16] used KC1, KC2, CM1, PC1, and JM1. In [47], researchers used PC1, JM1, KC1, and KC3. Researchers in [8] used PC3, PC4, KC1, and JM1. In [48], researchers used KC1, CM1, PC3, and PC4.

RQ5: what is the source of used data set?

Researchers of [43, 44] and [8, 16, 46–48] used the data set from PTOMISE repository; however, researcher in [45] used the data sets from PROMISE repository as well as from SOFTLAB.

RQ6: which tool is used for the experiment and simulation of results?

Researchers in [43] used "Python Keras" library for the experiment and simulation of results. MATLAB tool was used by [44, 46] and [8, 47, 48] for simulations and experiments.

6. Conclusion and Future Work

Today's modern life is getting digitalized day by day, due to which the exponent increase in the demand for software systems triggered the need for high-quality maintainable softwares at lower cost. Timely delivery of qualitative software is now considered as one of the main concerns of software developers. However, due to the growing complexities in software systems, it is becoming more difficult to deliver a high-quality maintainable software at lower cost. This issue can be solved by using software defect prediction approaches. ANN is considered as one of the widely used supervised machine learning techniques to predict the defects at early stages of SDLC. This research has focused to extract latest published literature from 2015 till 2018 which used ANN for software defect prediction. Three well-known and widely used online search libraries were used for the literature extraction such as IEEE, Elsevier, and Springer. A step-by-step systematic research process was followed in order to extract and shortlist the most relevant latest papers for critical review. The research protocol of this research starts with the identification of research questions which are then answered at the end after critical review. It has been concluded that ANN has wide scope for software defect prediction especially when used with a hybrid approach after integration with any other technique such as future selection or any other preprocessing technique. Moreover, it has also been seen that most of the researches have worked on preprocessing technique in order to improve the performance of ANN classifier. The performance of proposed techniques was evaluated in terms of various measures, calculated from confusion matrix and compared with conventional base classifiers as well as with proposed optimized techniques. Data set for the experiment was mostly used from the Promise repository. For simulation, MATLAB was the most common tool. This research can be used as a base study for innovations and for further comparisons and reviews. Further studies can be carried out by focusing on other machine learning techniques other than ANNs on software defect prediction. Moreover, particular data sets can be investigated by keeping in view the reported accuracies of newly proposed techniques. Modern feature selection and ensemble techniques can also be investigated on the problem of software defect prediction.

Data Availability

The data used in this paper can be requested from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this work.

References

[1] K. Gao and T. Khoshgoftaar, "Software defect prediction for high-dimensional and class-imbalanced data," in *Proceedings of the 23rd SEKE*, vol. 2, pp. 89–94, Miami Beach, FL, USA, July 2011.

[2] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proceedings of the 33rd International Conference on Software Engineering*, pp. 481–490, Honolulu , HI, USA, April 2011.

[3] Y. Zhang, D. Lo, X. Xia, and J. Sun, "An empirical study of classifier combination for cross-project defect prediction," *IEEE Annual International Computer Software and Applications Conference*, vol. 2, pp. 264–269, 2015.

[4] J. Goyal and R. Ranjan Sinha, "Software defect-based prediction using logistic regression: review and challenges," in *Proceedings of the 2nd International Conference on Sustainable Technologies for Computational Intelligence*, pp. 233–248, Dehradun, India, January 2022.

[5] T. Wang, W. Li, H. Shi, and Z. Liu, "Software defect prediction based on classifiers ensemble," *Journal of Information and Computing Science*, vol. 8, no. 16, pp. 4241–4254, 2011.

[6] M. Pandit, D. Gupta, D. Anand et al., "Towards design and feasibility analysis of DePaaS: AI based global unified software defect prediction framework," *Applied Sciences*, vol. 12, no. 1, p. 493, 2022.

[7] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," *Information and Software Technology*, vol. 58, pp. 388–402, 2015.

[8] R. Jayanthi and L. Florence, "Software defect prediction techniques using metrics based on neural network classifier," *Cluster Computing*, vol. 22, pp. 1–12, 2018.

[9] A. Iqbal and S. Aftab, "A classification framework for software defect prediction using multi-filter feature selection technique and MLP," *International Journal of Modern Education and Computer Science*, vol. 12, no. 1, 2020.

[10] M. Ahmad, S. Aftab, and S. S. Muhammad, "Machine learning techniques for sentiment analysis: a review," *International*

Journal of Multidisciplinary Sciences and Engineering, vol. 8, no. 3, p. 27, 2017.

[11] M. Ahmad, S. Aftab, I. Ali, and N. Hameed, “Hybrid tools and techniques for sentiment analysis: a review,” *International Journal of Multidisciplinary Sciences and Engineering*, vol. 8, no. 3, 2017.

[12] M. Ahmad and S. Aftab, “Analyzing the performance of SVM for polarity detection with different datasets,” *International Journal of Modern Education and Computer Science*, vol. 9, no. 10, pp. 29–36, 2017.

[13] M. Ahmad, S. Aftab, M. S. Bashir, N. Hameed, I. Ali, and Z. Nawaz, “SVM optimization for sentiment analysis,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 4, 2018.

[14] M. Ahmad, S. Aftab, and I. Ali, “Sentiment analysis of tweets using SVM,” *International Journal of Computer Application*, vol. 177, no. 5, pp. 25–29, 2017.

[15] J. Li, P. He, J. Zhu, and M. R. Lyu, “Software defect prediction via convolutional neural network,” in *Proceedings of the 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, pp. 318–328, Prague, Czech, July 2017.

[16] Ö. F. Arar and K. Ayan, “Software defect prediction using cost-sensitive neural network,” *Applied Soft Computing*, vol. 33, pp. 263–277, 2015.

[17] D.-L. Miholca, G. Czibula, and I. G. Czibula, “A novel approach for software defect prediction through hybridizing gradual relational association rules with artificial neural networks,” *Information Sciences*, vol. 441, pp. 152–170, 2018.

[18] M. Gayathri and A. Sudha, “Software defect prediction system using multilayer perceptron neural network with data mining,” *International Journal of Recent Technology and Engineering*, vol. 32, pp. 2277–3878, 2014.

[19] M. Singh and D. Singh Salaria, “Software defect prediction tool based on neural network,” *International Journal of Computer Application*, vol. 70, no. 22, pp. 22–28, 2013.

[20] P. B. Crosby, *Quality Is Free : The Art of Making Quality Certain*, McGraw-Hill, New York, NY, USA, 1979.

[21] Y. Koroglu, A. Sen, D. Kutluay et al., “Defect prediction on a legacy industrial software: a case study on software with few defects,” in *Proceedings of the 2016 4th International Workshop on Conducting Empirical Studies in Industry*, pp. 14–20, Austin, TX, USA, May 2016.

[22] T. Sethi and G. Gagandeep, “Improved approach for software defect prediction using artificial neural networks,” in *Proceedings of the 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, pp. 480–485, Noida, India, September 2016.

[23] C. He, J. Xing, R. Zhu, J. Li, Q. Yang, and L. Xie, “A new model for software defect prediction using Particle Swarm Optimization and support vector machine,” in *Proceedings of the 2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 4106–4110, Guiyang, China, May 2013.

[24] T. Wang and W. H. Li, “Naïve Bayes software defect prediction model,” in *Proceedings of the 2010 International Conference on Computational Intelligence and Software Engineering*, pp. 0–3, Wuhan, China, December 2010.

[25] M. Kakkar and S. Jain, “Feature selection in software defect prediction: a comparative study,” in *Proceedings of the 2016 6th International Conference, Cloud System and Big Data Engineering (Confluence)*, pp. 658–663, Noida, India, January 2016.

[26] G. K. Armah, G. Luo, and K. Qin, “Multi-level data preprocessing for software defect prediction,” in *Proceedings of the 2013 6th International Conference on Information Management, Innovation Management and Industrial Engineering ICIII 2013*, pp. 170–174, Xi’an, China, November 2013.

[27] J. H. Lo, “A data-driven model for software reliability prediction,” in *Proceedings of the IEEE International Conference on Granular Computing*, vol. 29, pp. 326–331, Washington, DC, USA, August 2012.

[28] A. D. Oral and A. B. Bener, “Defect prediction for embedded software,” in *Proceedings of the 22nd International Symposium on Computer Science and Computational Technology ISC 2007*, pp. 346–351, San Fancisco, CA, USA, October 2007.

[29] A. Singh and R. Singh, “Assuring software quality using data mining methodology: a literature study,” in *Proceedings of the 2013 International Conference on Dependable Systems and Networks, ISCON 2013*, pp. 108–113, Mathura, India, March 2013.

[30] V. Challagulla, F. Bastani, I.-L. Yen, and R. Paul, “Empirical assessment of machine learning based software defect prediction techniques,” in *Proceedings of the 10th IEEE International Workshop Object-Oriented Real-Time Dependable Systems*, pp. 263–270, Sedona, Arizona, February 2005.

[31] R. S. Wahono, “A systematic literature review of software defect prediction,” *Journal of Software Engineering*, vol. 1, no. 1, pp. 1–16, 2015.

[32] A. Okutan and O. T. Yıldız, “Software defect prediction using Bayesian networks,” *Empirical Software Engineering*, vol. 19, no. 1, pp. 154–181, 2014.

[33] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering –A systematic literature review,” *Information and Software Technology*, vol. 51, pp. 7–15, 2008.

[34] P. Brereton, B. A. Kitchenham, D. Budgen, M. Turner, and M. Khalil, “Lessons from applying the systematic literature review process within the software engineering domain,” *Journal of Systems and Software*, vol. 80, no. 4, pp. 571–583, 2007.

[35] Z. Nanli, Z. Ping, L. Weiguo, and C. Meng, “Sentiment analysis: a literature review,” in *Proceedings of the 2012 International Symposium on Management of Technology (ISMOT)*, pp. 572–576, Hangzhou, China, November 2012.

[36] S. Ashraf, S. Aftab, and S. Aftab, “Latest transformations in scrum: a state of the art review,” *International Journal of Modern Education and Computer Science*, vol. 9, no. 7, pp. 12–22, 2017.

[37] S. Ashraf and S. Aftab, “Scrum with the spices of agile family: a systematic mapping,” *International Journal of Modern Education and Computer Science*, vol. 9, no. 11, pp. 58–72, 2017.

[38] F. Anwer, S. Aftab, and S. Aftab, “Latest customizations of Xp: a systematic literature review,” *International Journal of Modern Education and Computer Science*, vol. 9, no. 12, pp. 26–37, 2017.

[39] M. Ahmad, S. Aftab, M. S. Bashir, and N. Hameed, “Sentiment analysis using SVM: a systematic literature review,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 2, 2018.

[40] S. Aftab, M. Ahmad, N. Hameed, M. S. Bashir, I. Ali, and Z. Nawaz, “Rainfall prediction using data mining techniques: a systematic literature review,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 5, 2018.

[41] F. Matloob, S. Aftab, M. Ahmad et al., “Software defect prediction using supervised machine learning techniques: a systematic literature review,” *Intelligent Automation & Soft Computing*, vol. 29, no. 2, pp. 403–421, 2021.

[42] F. Matloob, T. M. Ghazal, N. Taleb et al., "Software defect prediction using ensemble learning: a systematic literature review," *IEEE Access*, 2021.

[43] J. Li, P. He, J. Zhu, and R. L. Michael, "Software defect prediction via convolutional neural network," in *Proceedings of the QRS'17: International Conference on Software Quality, Reliability and Security*, pp. 318–328, Prague, Czech, July 2017.

[44] I. Arora and A. Saha, "Comparison of back propagation training algorithms for software defect prediction," in *Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 51–58, Noida, India, December 2016.

[45] Q. Cao, Q. Sun, Q. Cao, and H. Tan, "Software defect prediction via transfer learning based neural network," in *Proceedings of the 2015 the 1st International Conference on Reliability Systems Engineering (ICRSE)*, Beijing, China, October 2015.

[46] W. Geng, "Cognitive Deep neural networks prediction method for software fault tendency module based on bound particle swarm optimization," *Cognitive Systems Research*, vol. 52, pp. 12–20, 2018.

[47] C. Jin and S.-W. Jin, "Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization," *Applied Soft Computing*, vol. 35, pp. 717–725, 2015.

[48] C. Manjula and L. Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics," *Cluster Computing*, pp. 1–17, 2018.