Hindawi
Security and Communication Networks
Volume 2023, Article ID 9762076, 1 page
https://doi.org/10.1155/2023/9762076

Retraction

Retracted: Defect Prediction Technology in Software Engineering Based on Convolutional Neural Network

Security and Communication Networks

Received 11 July 2023; Accepted 11 July 2023; Published 12 July 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope
(2) Discrepancies in the description of the research reported
(3) Discrepancies between the availability of data and the research described
(4) Inappropriate citations
(5) Incoherent, meaningless and/or irrelevant content included in the article
(6) Peer-review manipulation

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] C. Liu, S. Sanober, A. S. Zamani, L. R. Parvathy, R. Neware, and A. W. Rahmani, "Defect Prediction Technology in Software Engineering Based on Convolutional Neural Network," *Security and Communication Networks*, vol. 2022, Article ID 5058461, 8 pages, 2022.

Hindawi
Security and Communication Networks
Volume 2022, Article ID 5058461, 8 pages
https://doi.org/10.1155/2022/5058461

Research Article

Defect Prediction Technology in Software Engineering Based on Convolutional Neural Network

Can Liu,[1] **Sumaya Sanober**,[2] **Abu Sarwar Zamani**,[3] **L. Rama Parvathy**,[4] **Rahul Neware**,[5] **and Abdul Wahab Rahmani**[6]

[1]*China University of Petroleum-Beijing At Karamary, Karamay, Xinjiang Uygur Autonomous Region 834000, China*
[2]*Prince Sattam Bin Abdul Aziz University, Wadi Aldwassir, Saudi Arabia*
[3]*Department of Computer and Self Development, Prince Sattam Bin Abdulaziz University, Al-Kharj, Saudi Arabia*
[4]*Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Tamil Nadu, India*
[5]*Department of Computing, Mathematics and Physics, Høgskulen På Vestlandet, Bergen, Norway*
[6]*Isteqlal Institute of Higher Education, Kabul, Afghanistan*

Correspondence should be addressed to Can Liu; canliu3@126.com, Rahul Neware; rane@hvl.no, and Abdul Wahab Rahmani; ab.wahab.professor@isteqlal.edu.af

Received 15 February 2022; Accepted 29 March 2022; Published 26 April 2022

Academic Editor: Mukesh Soni

Software defect prediction has become a significant study path in the field of software engineering in order to increase software reliability. Program defect predictions are being used to assist developers in identifying potential problems and optimizing testing resources to enhance program dependability. As a consequence of this strategy, the number of software defects may be predicted, and software testing resources are focused on the software modules with the most problems, allowing the defects to be addressed as soon as feasible. The author proposes a research method of defect prediction technology in software engineering based on convolutional neural network. Most of the existing defect prediction methods are based on the number of lines of code, module dependencies, stack reference depth, and other artificially extracted software features for defect prediction. Such methods do not take into account the underlying semantic features in software source code, which may lead to unsatisfactory prediction results. The author uses a convolutional neural network to mine the semantic features implicit in the source code and use it in the task of software defect prediction. Empirical studies were conducted on 5 software projects on the PROMISE dataset and using the six evaluation indicators of Recall, F1, MCC, pf, gm, and AUC to verify and analyze the experimental results showing that the AUC values of the items varied from 0.65 to 0.86. Obviously, software defect prediction experimental results obtained using convolutional neural networks are still ideal. Defect prediction model in software engineering based on convolutional neural network has high prediction accuracy.

1. Introduction

With the advancement of computer expertise, various software products are gradually applied in all aspects of human life; however, while computer software brings convenience to human life, it also brings some dangers and disasters; examples of catastrophic accidents due to some small software bug are not uncommon, such as the 1982 Soviet oil pipeline incident, the June 4, 1996, Space Shuttle 501 explosion, in 2000, the Panama City radiation dose exceeded the standard accident, and the Ariane 5 launch vehicle, the first operation failure event; therefore, the reliability of computer software has always been an issue of great concern [1]. Machine learning has been effectively implemented in a range of applications, and it is now a research hotspot in the field of artificial intelligence including speech recognition and image categorization. However, deep neural networks, for example, CNN and

RNN, are often employed yet have not even been demonstrated to be useful. Improvements have been made to the defect prediction mechanism [2, 3]. The creation of software flaws is caused by developers' incorrect understanding of the design of software requirements, or negligence when developing software, and problems with internal calls of system. A software defect is a property that does not conform to the intended design of the software; if it is not discovered and removed in time, it may have varying degrees of impact on the reliability of the software [4]. It is very crucial to find and fix software bugs as early and as early as possible in order to assure software quality. The most significant difficulty in ensuring long-term error-free operation of software systems is how to discover faults [5, 6]. Moreover, because system implementation code defects are regarded as the primary causes of failure, modern software is often extremely complex and prone to errors; reliability has become a critical challenge as its complexity has expanded [7, 8].

The goal of software defect prediction technology is to train model parameters using historical defect data, thereby a software defect prediction model is entrenched, and apply the established model to the prediction of unknown software, through this process, because the amount of software defects can be forecast, software testing resources are concentrated in the software modules with the most flaws, so that the problems may be fixed as quickly as feasible. The goal of software defect prediction technology is to train model parameters using historical defect data, ensure the rational allocation of time and budget, and save manpower and material resources [9]. Software fault forecasting is the process of building classifiers to anticipate code portions that may contain faults based on information including such code complexity and change history [10, 11]. From the 1970s to the present, in the realm of software engineering, software defect prediction technology has always been a research priority, with software defect prediction technology being used to direct software testing efforts, not only can it maximize the search for defects and optimize the software but also save a lot of costs; therefore, software defect prediction technology can analyze software quality and balance software cost, playing an important role [12]. Software process quality control and quality management methodologies, however, can no longer match the present demands because of the rising cost of testing and the increasing complexity of software systems. If deep learning technology is used during the software development and testing stages to deep seek, crawl, and analyze the program's historical problem data, the dispersion and quantity of faults in the software program may be predicted and counted in advance to a degree. Deep learning could also combine core defect information into semantic representations features and compensate for algorithms algorithms' shortcomings and is presently a research hotspot in the area of artificial intelligence. Popular learning algorithms, on the other hand, have yet to be applied to real-time defect prediction [13, 14]. Defects in software are errors or flaws in the software creation or maintenance process on the internal level; on the external level, defects are violations or failures of the functions that the software is supposed to execute. Software flaws influence software quality, and early detection and repair of faults are critical for software quality assurance [15]. Currently, deep learning technology is the forefront of research in the field of artificial intelligence and has been well used in several area, including object detection, natural language processing, and image recognition [16]. CNN effectiveness is significantly impacted by factors such as filter length and batch size. The model just would not converge even with the worst parameter values. As a consequence, parameter tuning is critical for training a good CNN [17, 18]. Therefore, a mainstream deep learning technique convolution neural network (CNN) is adopted for software defect prediction. Figure 1 shows the software defect prediction research. This article examined network techniques for real-time defect prediction. Traditional software assessment and quality assurance approaches, on the other hand, are no longer capable of meeting current needs, because of the enlarged price of testing and the rising complexity of software systems. Software engineers can retest application faulty components using the proposed software defect prediction method. Furthermore, by focusing so much effort on defect-prone software components rather than non-defect-prone application components, software project resources are used more efficiently, resulting in a significant reduction in the amount of social and financial capital ingested by analysis, lower analysis costs, and increased research and development effectiveness.

Figure 1 clearly illustrates the layout for the software defect prediction model. Software faults must be recognised and removed quickly in order for the process to improve the dependability and usefulness of the software.

2. Literature Review

Many authors have performed various studies on software defect prediction technology over the years, and a software defect prediction model based on machine learning and statistics has been presented. To the 146 components of the ADA system, the researchers used logistic regression, classification trees, and OSR approaches prediction studies; they choose performance metrics for correctness and completeness, in order to evaluate the constructed software defect prediction model; the results show that, after using OSR technology, the correctness and completeness of the prediction model reached 90% [19]. Researches on a telecommunications system containing 13 million lines of code, an artificial neural network model, and a discriminant model are established, and the two models are compared, the analytical presentation of artificial neural network models is discovered, and discriminative models are outperformed [20]. Researchers apply genetic methods to defect prediction in military communication systems and telecommunications systems. They applied eight-level metrics, including obtained high prediction accuracy [21]. Using decision trees and ensemble learning technology, the researchers created a dynamic software defect prediction model, the model uses the bagging ensemble learning method based on decision tree technology in the inner layer, a stochastic deep forest

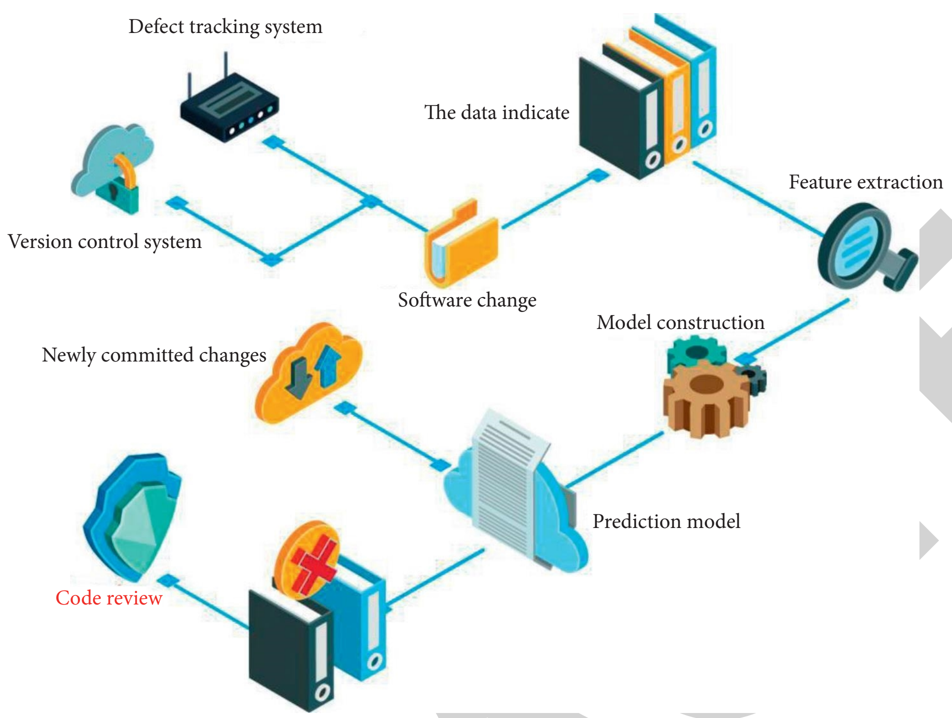

Figure 1: Research on software defect prediction.

model is constructed, and random sampling and stacking methods are used in the outer layer to train these random forest models, and a certain prediction effect is achieved [22]. The investigators used logistic regression to estimate software faults on 37 indicators of antenna configuration software, identify static software metrics and the amount of software flaws, and find a correlation between the two [23, 24]. Fuzzy clustering is used by researchers to forecast software problems, and he used the Radial Basis Function (RBF) to do so although the multilayer perceptron (MLP) is efficient, but the inspection cost is also high [25]. The accuracy of RBF is 83%, and the accuracy of delay is 60%. Therefore, the RBF method is more widely used in software fault prediction research than MLP [26]. The researchers built a Bayesian network defect prediction model and evaluated the outcomes using PD and PF as performance measures; through research, they concluded that performing software defect prediction work is the guarantee of improving software quality [27]. Traditional input-output nets are broader in Bayesian networks. This is because any variable in the graph may serve as both an input and an output. We could even anticipate the likelihood of an outcome and a lacking input happening concurrently. Describing variables in terms of its purpose as inputs is still a valid concept. To examine software flaws, authors have developed neural network prediction models, and several neural network optimization models have been suggested [28]. Various authors have introduced deep learning into the study field of software imperfection prediction technology [29–31]. It is stated that one of the three major elements affecting defect prediction performance is the metric setup [32].

In summary, software fault prediction technology has become a vital tool for reducing analysis costs and ensuring software quality, and how to increase the fault prediction model's prediction accuracy and the software defect prediction framework will be a research hotspot in the field of software fault prediction technology.

3. Research Methods

3.1. Classification of Software Engineering Defect Prediction Technology

3.1.1. Static Software Defect Prediction Technology. The static software defect prediction technology consists of analyzing software module code, designing a corresponding measurement element, analyzing historical software defect data, establishing a suitable software defect prediction model based on the metric element, and then using the established software defect prediction model to predict software defects. An effective software fault prediction (SFP) model may aid programmers in promptly and correctly discovering problems, hence enhancing the overall dependability and quality of the software project. Because of variances in the prediction performance of training strategies for various software systems, selecting an effective learning method for defect prediction modeling is difficult. The metrics used in the early days are mainly the number of lines of code, the connection among the number of faults (D) and LOC (L): $D = 4.86 + 0.018L$, nevertheless, the prediction accurateness of the software defect prediction model established by using a single metric element is low; therefore, researchers have introduced a variety of software metrics, for example, Halstead scientific metrics and McCabe loop complex paths [33].

3.1.2. Dynamic Software Defect Prediction Technology. Dynamic software defect prediction is substantial and vital job with in early phases of the software development life cycle. This technique has received a lot of attention in recent years since it leads to the guarantee of existing software quality. The order to forecast erroneous or imperfect artefacts phase of the software development process can assist the design team in employing current assets more expertly and efficiently to create great software platforms within the time frame given. Dynamic software fault forecast technology is mostly aimed at number of historical software defects that appear over time, observing the regular changes of software defects with their life cycle; therefore, software defect prediction is carried out according to this law. In the process of software defect prediction, when we analyze the cause of defect, it takes a certain amount of time, this time is called the delay time, and the cumulative defect distribution observed with the delay time conforms to the S-Curve model, the S-Curve model is a reliability growth model that satisfies the inhomogeneous Poisson process, and it accurately captures the link between latency and software defect distribution, thus refining the accurateness of piece defect prediction. Rayleigh model can describe the distribution law of the amount of faults in entire software cycle and is one of the earliest software reliability models. The model function of the Rayleigh model is as follows:

$$F(t) = K\left[\left(\frac{1}{\text{peaak}}\right)^2 te^{-\left(1/2\text{peak}^2\right)t^2}\right], \tag{1}$$

where K is the total number of defects, peak is the maximum value of the time curve, and t is the time parameter.

3.2. Software Engineering Defect Prediction Model. Building a software defect prediction model using static metrics is a productive method to forecast software flaws. The attributes that better represent the projection outcomes from these metric elements are selected when metrics are abstracted from failure history data [34]. The Halstead scientific metric and the McCabe loop complex path metric are the most fundamental metric elements.

3.2.1. Halstead Scientific Metrics. A computer program is a set of symbols that may be classified as operator or operands and are used to carry out an algorithm. Halstead's measures are employed in a wide range of software product paragraph systems today. Halstead scientific metric is based on the number of operators and operands in executable lines of code in a program. Program complexity grows with the number of operators and operands. Let N1 and N2 denote the total number of operators and operands, n1 and n2 be the total number of different operators and operands, then the actual Halstead length is $N = N_1 + N_2$, and the predicted Halstead length is $H = n_1 \log_2^{n1} = n_2 \log_2^{n2}$.

Table 1: Statistical results of 5 evaluation indicators of convolutional neural network on 5 items.

Index	Recall	F1	MCC	pf	Gm
Ant-1.5	0.6646	0.7135	0.4002	0.2482	0.7056
Ant-1.6	0.6061	0.6700	0.4781	0.1278	0.7011
Camel-1.2	0.5222	0.5360	0.1553	0.3575	0.5675
Camel-1.4	0.6718	0.7130	0.4000	0.1770	0.7318
Poi-2.0	0.6000	0.7257	0.3467	0.2222	0.5718
Average	0.6631	0.7071	0.4263	0.2285	0.7002

3.2.2. McCabe Loop Complex. A testing metric used to quantify the complexity of a software programme is the McCabe loop complex, also known as Cyclomatic Complexity in Testing Process. It is a quantifiable measure of the number of unique routes in a software application's source code. Control flow graphs of events, packages, methods, or classes on the inside of a software programme can be used to determine cyclomatic complexity. The concept of complex road metrics for McCabe includes cyclocomplexity, cyclomatic density complexity, normalized cyclomatic complexity, and other metrics. The complexity of the McCabe loop is mainly through the abstraction of the program, the flow control diagram V(G) of the program is drawn, though $V(G) = e - n + p$, its loop complex path can be calculated, where e denotes the number of edges and n the number of nodes, and the value of P is 2. If the control graph of a program module contains 9 edges and 6 nodes, the cycle complexity is 9–6 + 2 = 5.

Different software defect prediction models, even the same model with different parameters, have different scopes of application; by comparing different software defect prediction models, the advantages and disadvantages of the models and the scope of application can be obtained. Therefore, researching new software defect prediction models, finding different parameter optimization methods, different processing of data, and feature selection methods are the keys to building a best software fault estimate model [35,36]. The model built by machine learning has the advantages of simple training and short time; the model constructed by statistics has the advantages of good fitting effect and accurate prediction. The current mainstream defect prediction models are based on these two sorts of approaches containing the support vector machine model, C4.5 decision tree model, Bayesian network model, neural network model, and logistic regression model.

3.3. Convolution Neural Networks. Several levels of artificial neurons make up convolutional neural networks. Artificial neurons, as actual neurons, are mathematical functions that determine the weighted sum of several inputs and provide an activation value. The weights of each neuron govern its activity. When given image pixels, a CNN's perceptron recognizes a variety of visual features. A convolution neural network is a feed forward neural network with convolutional computing capabilities and multiple network layers, it has powerful feature extraction ability, and it can map low-level original defect features into high-level abstract deep

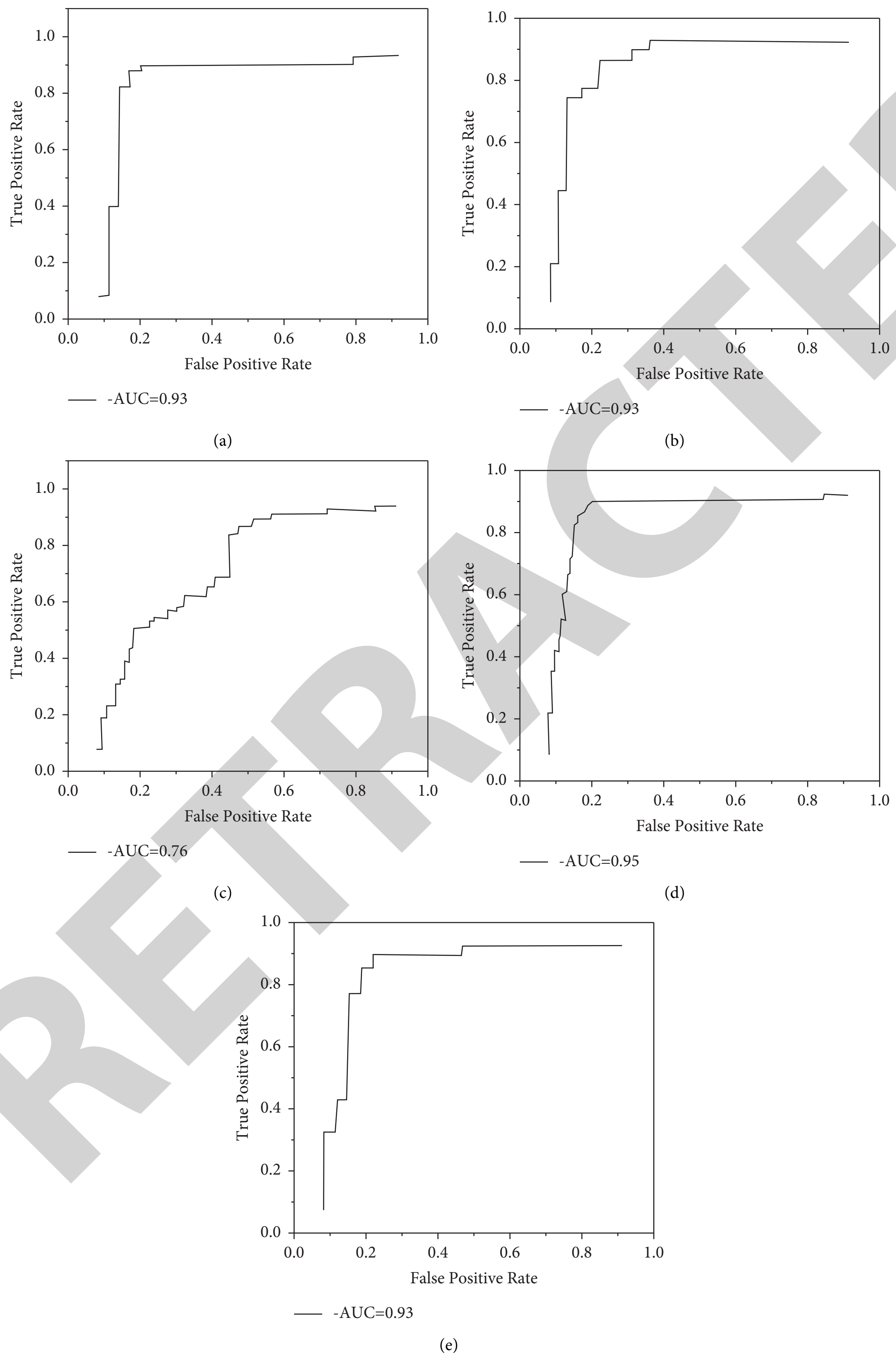

FIGURE 2: AUC values: (a) ant-1.5; (b) ant-1.6; (c) camel-1.2; (d) camel-1.4; (e) poi-2.0.

semantic features. The convolution kernel parameter sharing in the hidden layer, as well as the sparse structure of the link between layers in the convolution neural network, primarily acts on the network, allowing it to lattice features with a small amount of computation [37]. It considers dimensional features such as the input layer, convolution layer, and Relu nonlinear excitation. The convolutional layer comprehends multiple convolution kernels, which can accomplish attributes abstraction on input defect features; each element inside it relates to a weight coefficient and a bias. Matrix element multiplication and summation and stacking offset operations are performed by the convolution kernel on the input defect characteristic in accessible field, and the convolution process is shown in the following formula:

$$C^{l+1}(i, j) = \left[C^{l} \otimes W^{l+1}\right](i, j) + c, \tag{2}$$

where C^l and C^{l+1} denote the convolution input and output of the l+1th layer in turn, that is, the feature map, W^{l+1} represents the network weight, and c represents the bias.

Relu nonlinear excitation function: the nonlinear excitation function in the convolutional layer can perform nonlinear transformation on complex features, thereby enhancing the expressive ability of features. The process is shown in the following formula:

$$D^{l}_{i,j,k} = f\left(C^{l}_{i,j,k}\right), \tag{3}$$

where f(.) represents the nonlinear excitation function.

First, the 20-dimensional features of each software project into 25-dimensional features are mapped and reshaped to a 5x5 matrix. Then, two consecutive rounds of convolution operations are performed; the convolution kernel sizes are 3 × 3 and 5 × 5 and have 32 and 16 convolution kernels, respectively; there is a ReLU nonlinear excitation function in each round of convolution operation. Next, the fully connected layer performs nonlinear combination of the extracted features and output in the form of 1-dimensional features. Finally, the previously extracted defect features are classified by the softmax layer; thus, defect features and defect-free features are obtained [38, 39]. The convolutional neural network's network model is trained using a cross-entropy loss function, with a learning rate of 0.001, a batch size of 32, and 1000 training iterations.

4. Discussion of Results

4.1. Dataset. Software defect prediction on 5 software projects in the PROMISE dataset; these software projects are widely used and open source baseline datasets. The number of features of the five projects is all 20 dimensions, the number of instances is at least 293 and the most is 803, and the defect rate ranges from 10.0% to 48.2%. Due to the problem of class imbalance in all 5 items, before using convolutional neural network for defect prediction, all 5 items were class-balanced [40].

4.2. Evaluation Indicators. Five evaluation indicators are used to conduct comprehensive statistics and analysis on the experimental results, namely, recall rate (Recall), F1, MCC (Matthews's correlation coefficient), false positive rate (pf), gm (G-measure), and AUC. Except for the false positive rate, other indicators are that the bigger the better [41].

4.3. Experimental Results. For the various indices, the performance evaluation indicators were tested. It is clearly stated in Table 1 that the F1 has the maximum average value. As a result of this method, the amount of software defects may be forecast, and software testing resources can be concentrated on the software modules that have the greatest issues, allowing defects to be resolved as quickly as possible. Experiment in software engineering using convolutional neural networks, Recall, F1, MCC, pf, and gm are shown in Table 1, and AUC is shown in Figure 2. Recall, F1, MCC, pf, and gm are shown in Table 1, and AUC is shown in Figure 2.

From Table 1, it can be seen that the averages of the five indicators of Recall, F1, MCC, pf, and gm are 0.6631, 0.7071, 0.4263, 0.2285, and 0.7002, respectively. Focused on the F1 indicator, except for the value of 0.5360 in the camel-1.4 project and 0.6700 in the ant-1.6 project, the F1 values on the remaining items were all greater than 0.7, and the average value of 0.7071 on the 5 items was also greater than 0.7. From Figure 2, the AUC values of the items varied from 0.65 to 0.86. Obviously, the experimental results of software defect prediction obtained by using convolutional neural networks are still ideal.

5. Conclusion

To test the accuracy of a defect prediction approach based on a convolutional neural network in software engineering, empirical research is carried out on 5 software projects in the PROMISE dataset, and the verification analysis is carried out using the 6 evaluation indicators of Recall, F1, MCC, pf, gm, and AUC, and the experimental outcomes demonstrate the dominance of the present method. As a result of this method, the amount of software defects may be forecast, and software testing resources can be concentrated on the software modules that have the greatest issues, allowing defects to be resolved as quickly as possible. Convolutional neural networks can utilize convolutional layers, deep mining, and extraction of features hidden in software defect data and transform the original fault characteristics into advanced abstract deep semantic features, which have a greater discriminative capacity for software flaws than typical machine learning techniques, and these powerful discriminative abilities are not held by traditional machine learning methods. In the future, class imbalance concerns should be addressed to these datasets. To enhance enactment, ensemble learning and feature selection techniques should be examined. We will also focus on software engineering duties such as code completion and code clone detection.

Data Availability

On request, the data used to support the results of the study are obtained from the corresponding author.

Conflicts of Interest

The authors state that the publishing of this work does not include any conflicts of interest.

References

[1] L. Zhao, Y. Zhang, and J. Li, "Research on constructing a degradation index and predicting the remaining useful life for rolling element bearings of complex equipment," *Journal of Mechanical Science and Technology*, vol. 35, no. 10, pp. 4313–4327, 2021.

[2] S. F. Suhel, V. K. Shukla, S. Vyas, and V. P. Mishra, "Conversation to automation in banking through chatbot using artificial machine intelligence language," in *Proceedings of the 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, June 2020.

[3] P. Balaji and K. Chidambaram, "Cancer diagnosis of microscopic biopsy images using a social spider optimisation-tuned neural network," *Diagnostics*, vol. 12, no. 1, p. 11, 2021.

[4] A. A. Bangash, H. Sahar, A. Hindle, and K. Ali, "On the time-based conclusion stability of cross-project defect prediction models," *Empirical Software Engineering*, vol. 25, no. 6, pp. 1–38, 2020.

[5] L. Gong, S. Jiang, L. Bo, L. Jiang, and J. Qian, "A novel class-imbalance learning approach for both within-project and cross-project defect prediction," *IEEE Transactions on Reliability*, vol. 69, no. 1, pp. 40–54, 2020.

[6] R. Yogesh, A. K. Dubey, R. R. Arora, and A. Mathur, "Fruit defect prediction model (fdpm) based on three-level validation," *Journal of Nondestructive Evaluation*, vol. 40, no. 2, pp. 1–12, 2021.

[7] G. S. Sriram, "Challenges of cloud compute load balancing algorithms," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1186–1190, 2022.

[8] H. Li, M. Shabaz, and R. Castillejo-Melgarejo, "Implementation of python data in online translation crawler website design," *International Journal of System Assurance Engineering and Management*, vol. 2021, p. 7, 2021.

[9] G. Esteves, E. Figueiredo, A. Veloso, M. Viggiato, and N. Ziviani, "Understanding machine learning software defect predictions," *Automated Software Engineering*, vol. 27, no. 3-4, pp. 369–392, 2020.

[10] G. S. Sriram, "Security challenges of big data computing," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1164–1171, 2022.

[11] X. Huang, V. Jagota, E. Espinoza-Muñoz, and J. Flores-Albornoz, "Tourist hot spots prediction model based on optimized neural network algorithm," *International Journal of System Assurance Engineering and Management*, vol. 13, pp. 63–71, 2022.

[12] T. Gantala and K. Balasubramaniam, "Automated defect recognition for welds using simulation assisted tfm imaging with artificial intelligence," *Journal of Nondestructive Evaluation*, vol. 40, no. 1, pp. 1–24, 2021.

[13] J. Bhola, M. Shabaz, G. Dhiman, S. Vimal, P. Subbulakshmi, and S. K. Soni, "Performance evaluation of multilayer clustering network using distributed energy efficient clustering with enhanced threshold protocol," *Wireless Personal Communications*, vol. 2021, p. 21, 2021.

[14] G. S. Sriram, "Resolving security and data concerns in cloud computing by utilizing a decentralized cloud computing option," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 4, no. 1, pp. 1269–1273, 2022.

[15] K. Zhu, N. Zhang, S. Ying, and D. Zhu, "Within-project and cross-project just-in-time defect prediction based on denoising autoencoder and convolutional neural network," *IET Software*, vol. 14, no. 3, pp. 185–195, 2020.

[16] S. Amasaki, "Cross-version defect prediction: use historical data, cross-project data, or both?" *Empirical Software Engineering*, vol. 25, no. 2, pp. 1573–1595, 2020.

[17] S. Tang and M. Shabaz, "A new face image recognition algorithm based on cerebellum-basal ganglia mechanism," *Journal of Healthcare Engineering*, vol. 2021, Article ID 3688881, 11 pages, 2021.

[18] M. N. Kumar, V. Jagota, and M. Shabaz, "Retrospection of the optimization model for designing the power train of a formula student race car," *Scientific Programming*, vol. 2021, Article ID 9465702, 9 pages, 2021.

[19] F. Spranger, M. Lopes, S. Schirdewahn, J. Degner, M. Merklein, and H. Kai, "Microstructural evolution and geometrical properties of tib2 metal matrix composite protrusions on hot work tool steel surfaces manufactured by laser implantation," *International Journal of Advanced Manufacturing Technology*, vol. 106, no. 1-2, pp. 481–501, 2020.

[20] S. Deif and M. Daneshmand, "Multiresonant chipless rfid array system for coating defect detection and corrosion prediction," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 10, pp. 8868–8877, 2020.

[21] S. Simeng, C. Xiaoxin, Q. Yong, and W. Hui, "Research on time-domain transfer impedance measurement technology for high frequency current transformers in partial discharge detection of cables," *Journal of Shanghai Jiaotong University*, vol. 25, no. 1, pp. 10–17, 2020.

[22] K. Pekkan and J. N. Oshinski, "Shaping the field of cardiovascular fluid mechanics: the 40th anniversary of ajit yoganathan's research laboratory," *Cardiovascular Engineering and Technology*, vol. 12, no. 6, pp. 557-558, 2021.

[23] A. N. Gorban, E. M. Mirkes, and I. Y. Tukin, "How deep should be the depth of convolutional neural networks: a backyard dog case study," *Cognitive Computation*, vol. 12, no. 1, pp. 388–397, 2020.

[24] V. Bhatia, S. Kaur, K. Sharma, P. Rattan, V. Jagota, and M. A. Kemal, "Design and simulation of capacitive MEMS switch for ka band Application," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 2021513, 8 pages, 2021.

[25] M. Poongodi, M. Hamdi, A. Sharma, M. Ma, and P. K. Singh, "DDoS detection mechanism using trust-based evaluation system in VANET," *IEEE Access*, vol. 7, pp. 183532–183544, 2019.

[26] Z. Huang, J. Zhu, J. Lei, X. Li, and F. Tian, "Tool wear monitoring with vibration signals based on short-time fourier transform and deep convolutional neural network in milling," *Mathematical Problems in Engineering*, vol. 2021, no. 6, 14 pages, Article ID 9976939, 2021.

[27] S. Qin, T. Jin, R. C. Van Lehn, and V. M. Zavala, "Predicting critical micelle concentrations for surfactants using graph convolutional neural networks," *The Journal of Physical Chemistry B*, vol. 125, no. 37, pp. 10610–10620, 2021.

[28] A. Zhang, J. He, Y. Lin, Q. Li, W. Yang, and G. Qu, "Recognition of partial discharge of cable accessories based on convolutional neural network with small data set," *COMPEL: The International Journal for Computation & Mathematics in Electrical & Electronic Engineering*, vol. 39, no. 2, pp. 431–446, 2020.

[29] Z. Yan, Y. Yu, and M. Shabaz, "Optimization research on deep learning and temporal segmentation algorithm of video shot in basketball games," *Computational Intelligence and Neuroscience*, vol. 2021, Article ID 4674140, 10 pages, 2021.

[30] V. Jagota and R. K. Sharma, "Wear volume prediction of AISI H13 die steel using response surface methodology and artificial neural network," *Journal of Mechanical Engineering and Sciences*, vol. 14, no. 2, pp. 6789–6800, 2020.

[31] Y. Xiao and J. Bhola, "Design and optimization of prefabricated building system based on BIM technology," *International Journal of System Assurance Engineering and Management*, vol. 2021, p. 24, 2021.

[32] Z. Lv, H. Ding, L. Wang, and Q. Zou, "A convolutional neural network using dinucleotide one-hot encoder for identifying dna n6-methyladenine sites in the rice genome," *Neurocomputing*, vol. 422, no. 4, pp. 214–221, 2021.

[33] K. C. Burak, M. K. Baykan, and H. Uuz, "A new deep convolutional neural network model for classifying breast cancer histopathological images and the hyperparameter optimisation of the proposed model," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 1–17, 2021.

[34] S. Baroud, S. Chokri, S. Belhaous, and M. Mestari, "A brief review of graph convolutional neural network based learning for classifying remote sensing images," *Procedia Computer Science*, vol. 191, no. 1, pp. 349–354, 2021.

[35] E. Gardini, M. J. Ferrarotti, A. Cavalli, and S. Decherchi, "Using principal paths to walk through music and visual art style spaces induced by convolutional neural networks," *Cognitive Computation*, vol. 13, no. 2, pp. 570–582, 2021.

[36] Y. Wang, J. Peng, and Z. Jia, "Brain tumor segmentation via c-dense convolutional neural network," *Progress in Artificial Intelligence*, vol. 10, no. 2, pp. 147–156, 2021.

[37] A. A. Hidayat, T. W. Cenggoro, and B. Pardamean, "Convolutional neural networks for scops owl sound classification," *Procedia Computer Science*, vol. 179, no. 4, pp. 81–87, 2021.

[38] M. A. Iqbal, Z. Wang, Z. A. Ali, and S. Riaz, "Automatic fish species classification using deep convolutional neural networks," *Wireless Personal Communications*, vol. 116, no. 2, pp. 1043–1053, 2021.

[39] Y. Xu, H. Cui, B. Fan et al., "Integrative model of ct imaging and clinical features using attentional multi-view convolutional neural network (am-cnn) for prediction of esophageal fistula in esophageal cancer," *International Journal of Radiation Oncology, Biology, Physics*, vol. 108, no. 3, pp. e637–e638, 2020.

[40] A. H. Ansari, O. De Wel, K. Pillay et al., "A convolutional neural network outperforming state-of-the-art sleep staging algorithms for both preterm and term infants," *Journal of Neural Engineering*, vol. 17, no. 1, pp. 0160281–01602811, 2020.

[41] A. Tahir, K. E. Bennin, X. Xiao, and S. G. Macdonell, "Does class size matter? an in-depth assessment of the effect of class size in software defect prediction," *Empirical Software Engineering*, vol. 26, no. 5, pp. 1–38, 2021.