International Research Journal of Innovations in Engineering and Technology (IRJIET)

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

A Comparative Study of CNN and MLP for Software Defect Prediction Using PROMISE Datasets

[1]Ekhlas Tariq Hasan, [2]Shayma Mustafa Mohi-Aldeen

[1]Department of Basic Science, College of Nursing, University of Mosul, Nineveh, Iraq
[2]Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Nineveh, Iraq
E-mails: [1]ekhlas.tarik@uomosul.edu.iq, [2]shaymamustafa@uomosul.edu.iq

Abstract **- Modern Software defects pose significant challenges, leading to critical system failures and substantial financial losses. As contemporary software systems become increasingly large and complex, identifying defects during the early stages grows more difficult. To address this, deep learning techniques, specifically multi-layer perceptron (MLP) and Convolutional Neural Network (CNN), are employed to predict software defects (SDP), integrated with code segment analysis during early development phases. This study applies both algorithms to detect software defects using 11 open-source datasets from the PROMISE repository. The evaluation emphasizes the prediction accuracy of the MLP and CNN models, alongside the F1 score-a crucial metric for assessing model performance on imbalanced datasets. Findings indicate that CNN outperforms MLP, achieving 86% prediction accuracy and an F1 score of 87.7%. In contrast, MLP attained 71% accuracy with an F1 score of 71.6%. These results demonstrate the superior predictive capability of CNN-based approaches in software defect prediction tasks.**

Keywords: Software Defects Prediction (SDP), Deep learning, CNN, MLP, RF.

I. INTRODUCTION

Software Defect Prediction (SDP) plays a pivotal role in supporting developers by enabling the rapid delivery of reliable, high-quality software systems [1]. The main goal of software testing is to ensure that applications are error-free and meet defined customer requirements. High-quality software, characterized by the absence of errors, is critical in applications deployed by governmental and official institutions [2]. The significance of (SDP) arises from the potential severity of programming errors, which can lead to hazardous situations, including accidents that result in human casualties and financial losses. As they have a direct impact on project planning and execution, software defect prediction and program error prediction are considered two core essential components within the field of software engineering.

These predictive processes involve identifying key factors such as development time, cost estimation, resource allocation, and the identification of failure-prone components [3]. The initial stages of the software development life cycle (SDLC) trigger organizations to use defect prediction software methods for identifying future errors. While deep learning methods specifically show promising capabilities for software defect identification, the current study deals with software unit categorization through using multiple models and methods [4].

Researchers have proposed several algorithms to predict software bugs, during which time various studies have evaluated the effectiveness of these methods to reach peak prediction accuracy. The ability of convolutional neural networks (CNNs) to handle adaptive data structures and quality makes them vital for bug prediction tasks. Since each program along with its distinctive qualities have different needs, software prediction data changes. Deep learning techniques are suitable for managing the extensive quantities of data found in source code as it frequently contains millions of lines and hundreds of files [5][6].

Software defect prediction utilises both (CNNs) and Multi-Layer Perceptron (MLP) models for analysis in this research study. The main research purpose targets the evaluation of CNN and MLP structures as defect prediction systems [7]. This paper is organized as follows: Section 2 describes the software defect prediction; Section 3 describes the Algorithms Used in Software Defect Prediction CNNs and MLPs. Section 4 describes the challenges in applying deep learning, section 5 describes the related works. The dataset used for software defect prediction is presented in section 6, while the research methodology described in with section7, the discussion and results presented in section 8, finally the conclusion are described in section 9.

II. SOFTWARE DEFECT PREDICTION (SDP)

Defect prediction is a technique used in software engineering to identify and predict defects and errors in software systems before they occur.[8] The goal of defect prediction is to improve software quality and reduce the number of post-release problems by enabling developers to

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

focus their efforts on high-risk areas. The benefits of defect prediction include the following: Early detection of errors: Defect prediction lowers the cost and effort needed to repair possible issues by enabling developers to identify them early in the development process.

Resource optimization: Better software quality can result from more effective use of resources by concentrating efforts on high-risk areas. Decision support: Project managers may make well-informed decisions regarding resource allocation and release planning by using the results of defect prediction. Process improvement: Organizations can pinpoint areas for improvement by using defect data analysis to get insight into the software development process [4].

III. SOFTWARE DEFECT PREDICTION USING DEEP LEARNING

Deep learning is an effective tool for predicting software defects, contributing to improved software quality and reducing the costs associated with bug fixes [8]. One of the most important benefits of deep learning is the automatic extraction of features from data. Deep learning models demonstrate superior performance in processing large and complex data, and they also have the ability to handle imbalanced data more effectively, increasing the accuracy of predictions. Deep learning has been used to detect software defects using several techniques, all of which rely on the analysis of software-related data [9].

Two deep learning algorithms have been used: the first is MLP. The structure of this Network is characterized by several layers, as shown in Figure 1. The input layer is the first layer whose function is to receive data, and the second layer consists of three hidden layers, depending on the complexity of the problem the network is trying to address. This layer uses the Relu activation function, which is the input to the output layer, and the last layer uses a Sigmoid-type activation function [10].

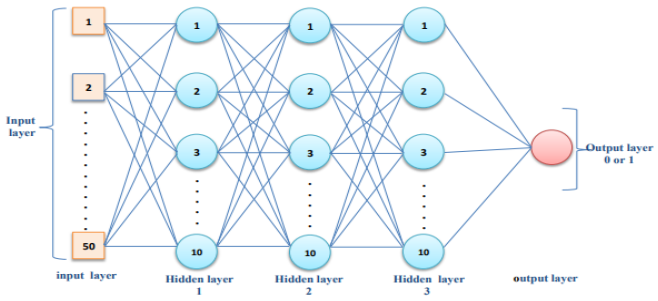

Figure 1: MLP architecture [10]

The second algorithm is the CNN, this algorithm is used to process structured data using convolutional layers to automatically learn features from the input data, consisting of an input layer, an output layer, convolutional layers, pooling layers, and fully connected layers, which makes it adept at extracting features automatically [11],[12], as shown in Figure (2).

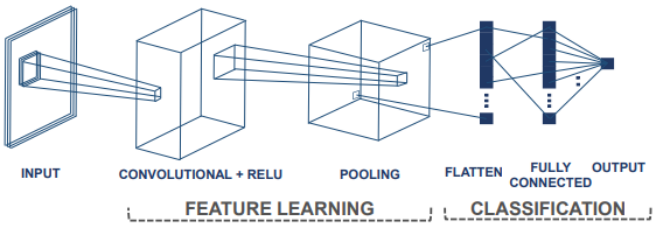

Figure 2: CNN architecture [13]

IV. CHALLENGES IN APPLYING DEEP LEARNING TO SOFTWARE DEFECT PREDICTION

The implementation of deep learning technology in many domains requires consideration of notable detriments within the system. The prediction of software engineering defects stands as a major ongoing obstacle during software development since it leads to system failure and enhances maintenance expenses, and complicates development processes. Serious problems exist in deep learning, but businesses can address them with proper knowledge and suitable implementation strategies. Two major obstacles faced by this implementation are:

1. As the principal obstacle facing deep learning models in software defect prediction, data often exists in an unbalanced state. The number of defects present in many software projects stands significantly lower than defect-free software occurrences. The prediction accuracy becomes low because the model displays bias by focusing on the most common data category (defect-free software) while ignoring the less common category (defects). Both issues can be solved by applying rebalancing techniques and implementing special algorithms to correct the imbalance problem.
2. The problem of the need for huge computational resources. Many organizations cannot utilize deep learning model training processes because they consume significant amounts of time, along with costly operations. Such models demand complex computing infrastructures that incorporate either GPUS or cloud services, which present both expenditure and technical complexity. Users can address these problems through Google Cloud cloud computing solutions that provide adjustable and affordable computational resources.
3. There exists an issue with interpreting standard evaluation outcomes. User confidence suffers when deep learning presents challenges for researchers and developers to interpret typical results, especially in

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

medical and security work fields. A solution to this problem requires building basic models that focus on important decision-making elements while receiving training from intermediate output results that assist in decision support. The reliability of data science projects improves significantly when data scientists team up with field experts to analyze sensitive information [14].

V. RELATED WORKS

Al Qasem and Akour (2019) conducted research into software fault prediction by implementing deep learning algorithms with Multi-layer Perceptron (MLPs) and Convolutional Neural Networks (CNNs) on NASA datasets. The authors reported that CNN provided better results than MLPS by reaching 100% accuracy on select datasets according to their research [10].

Alkaberi and Assiri (2024) built a deep learning model that used CNN and MLP for detecting software faults. Their evaluation used twelve open-source software project datasets obtained from PROMISE. MLP statistics showed a Kendall of 0.416 and MSE of 0.195, which surpassed CNN [15].

A system for duplicate bug report detection was created by Kukkar et al. (2020) through implementing Convolutional Neural Network (CNN) algorithms. The authors tested their model using six publicly accessible datasets which produced accuracy results between 85% and 99%. The research showed better results than current systems operate at [16].

The authors, Nevendra and Singh (2021), introduced an advanced version of Convolutional Neural Network (CNN) for software defect prediction through analysis of 19 open-source defect datasets. The researchers demonstrated that their approach surpassed Li's CNN along with standard machine learning models through several evaluation measurements [17].

Sharma *et al.* (2022) implemented the CCFT-CNN model with Correlation Clustering fine-tuning to process the PROMISE dataset within their software defect prediction operations. The authors used an algorithm which extracted data through Abstract Syntax Tree (AST) metrics. The researchers achieved a 2% enhancement in F-measure through their study using baseline models [18].

Many studies focused on the use of deep learning algorithms in detecting errors, as shown in Figure 3, which displays the percentage of each algorithm according to previous studies. The most widely used algorithms are CNN and MLP. This research used an approach to predict software defects using these algorithms, depending on increasing the number of layers and the number of entries. Software engineering metrics have been used in order to calculate the effectiveness of these algorithms and compare the results of performance and quality to improve their efficiency.

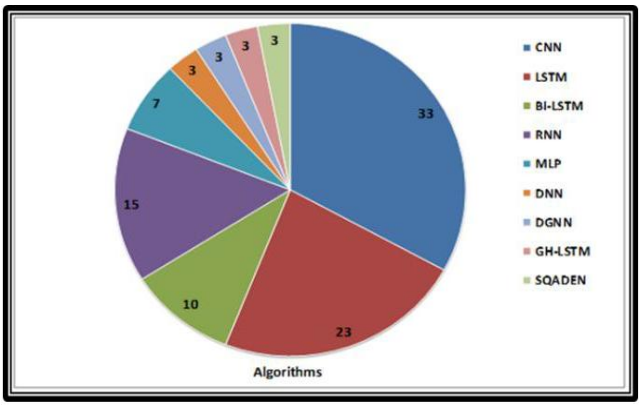

Figure 3: Description of Studies

VI. DATASET USED IN PREDICT SOFTWARE DEFECTS

The database that may contain software defects was adopted from an open-source software project from the PROMISE repository that was used to study software defect prediction, where the dataset consists of (11) software projects written in Java language, which are (ant, camel, ivy, xalan, velocity, log4j, jedit, poi, synapse, xerces, lucene), Table (1) shows the details of the dataset used.

Table 1: Dataset

Project	Version	No. of record
Ant	1.3	126
	1.4	179
	1.5	294
	1.6	352
	1.7	746
Camel	1	340
	1.2	609
	1.4	873
	1.6	966
Ivy	1	112
	1.1	242
	1.2	353
Xalan	2.4	724
	2.5	804
	2.6	886
	2.7	910
Velocity	1.4	197
	1.5	215
	1.6	230
Log4 j	1	136

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

	1.1	110
	1.2	206
Jedit	3.2	373
	4	307
	4.1	313
	4.2	368
	4.3	493
Poi	1.5	238
	2	315
	2.5	386
	3	443
Synapse	1	158
	1.1	223
	1.2	257
Xerces	1.1	163
	1.2	441
	1.3	454
	1.4.4	589
Lucene	2	196
	2.2	248
	2.4	341
Sum		15916

6.1 Preprocessing

Data preprocessing is the most important step of the main steps of machine learning and deep learning models. Data preprocessing includes data cleaning, feature extraction, and data splitting [19]. Performing data preprocessing steps has a significant impact on the performance efficiency of the model [20].

6.2 Data cleaning

The process of eliminating duplicate or inaccurate values from a data set is known as data cleaning [21]. It frequently occurs when two or more data sets are combined into a single file so that the data is appropriate for deep learning models. The number of effective and ineffective values records for each data set was calculated by sorting the error column (target) using Excel, where the first data set (ant) was calculated from a total of (125) records, the number of zero values was (105) records for ineffective values, while the remaining records contained (20) records for effective values.

Thus, the remaining ten data sets were sorted in the same way; these steps are used to provide a clear and comprehensive view of the data and software quality analysis, which contributes to improving software processes and making data-driven decisions. Full details of the data can be found in [22].

The number of defective records was 10047, and the number of non-defective records was 5828. The defective records represented only 63.1% while the non-defective records represented 36.6% of the total data volume, which indicates the presence of a defect in the prediction data set, as shown in Table 2

Table 2: No. of effective and ineffective values in the dataset

Project	No. of effective values	No. of in effective values
Ant	20	105
	40	138
	32	261
	92	259
	166	579
Camel	13	326
	216	392
	145	727
	188	777
Ivy	63	48
	16	225
	40	312
Xalan	110	613
	387	416
	411	474
	898	11
Velocity	147	49
	142	72
	78	151
Log4 j	34	101
	37	72
	189	16
Jedit	190	182
	75	231
	79	233
	48	319
	11	481
Poi	141	96
	37	277
	248	137
	281	161
Synapse	16	141
	60	162
	86	170
Xerces	77	85
	71	369
	69	384
	437	151
Lucene	91	104
	144	103

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

	203	137
Sum	10047	5828

6.3 Extract features

The current information age is moving towards the production of huge amounts of data. Due to the vast nature of the data [23], it becomes difficult for the algorithm to process the data and transform it into useful knowledge [24]. Moreover, not all the data collected will apply to a particular task [25]. Feature selection is the process of identifying and selecting a subset of features that are most relevant to the information from a larger set of features available from the dataset [26], [27]. This process aims to enhance the performance of the deep learning model by removing unwanted and redundant features and simplifying the complexity of the model. Data preprocessing plays an important role [28].

The features were ranked by calculating the importance of each feature using a type of decision tree-based classifier (RF model), which has good performance in predicting software defects [29],[30]. Feature selection can speed up the training of the classifier and improve accuracy [31][32]. The RF classifier ranks the importance of features of a dataset by selecting relevant features and excluding fewer valuable aspects. The interpretability of the RF classifier is one of its main advantages [33],[34]. The RF classifier provides the value of each feature as a number between 0 and 1. Table 3 shows the Code Quality Indicators and Feature Importance for (20) Feature [35].

Table 3: Code Quality Indicators and Feature Importance for (20) Feature

Abbreviation	Name	Importance of the feature
WMC	Weighed methods per class	0.04344
DIT	Depth of inheritance tree	0.02355
NOC	Number of children	0.0144
CBO	Coupling between object classes	0.07172
RFC	Response for a class	0.07584
LCOM	Lack of cohesion in methods	0.0444
CA	Afferent couplings	0.06011
LOC	Lines of code	0. 11211
DAM	Data access metric	0.0274
MOA	Measure of aggregation	0.02022
MFA	The measure of function abstraction	0.05523
CAM	Cohesion among the methods of the class	0.07102
CBM	Coupling between methods	0.0324
NPM	Number of public methods	0.0475
CE	Efferent couplings	0.05504
LCOM3	Lack of cohesion in methods	0.0574
IC	Inheritance coupling	0.01138
AMC	Average method complexity	0.09094
Max-cc	Max McCabe's cyclomatic complexity	0.0328
Avg-cc	Avg McCabe, s cyclomatic complexity	0.0527

6.4 Dataset splitting

The dataset is usually divided into a set for training the algorithm (Training Set) and a set for testing it (Testing Set) [36]. The data in this research were divided into 80% training data and 20% testing data; The training dataset contains the inputs and outputs (Target) used to train the model. The test set is used to generate model predictions to evaluate the performance and prediction capabilities of the optimal algorithm after it has been subjected to the training set, so the test set and the training set must be mutually exclusive and the test sample must not appear in the training set or be used in the training process.

VII. RESEARCH METHODOLOGY

This section discusses the methodology used to evaluate the effectiveness of deep learning algorithms in predicting software defects. The primary goal is to build predictive models capable of identifying defect-prone software components early in the software development lifecycle. The process began by collecting 11 open-source datasets from the PROMISE repository. These datasets include software projects with diverse characteristics, allowing for broad generalization of the results. A series of preprocessing steps were performed. First, the class labels in the target variable for binary classification were normalized: all values equal to or greater than 1 were scored as 1 (indicating the presence of a defect), while values equal to 0 remained unchanged (indicating the absence of a defect). Code attributes were then selected using the Random Forest (RF) algorithm, which ranks attributes based on their importance in classification. CNN and MLP algorithms were then applied. Model performance was evaluated using metrics such as prediction accuracy and F1 score, as shown in Figure 4.

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

The First algorithm MLP has used one input layer, which includes 50 nodes, and three hidden layers, each layer includes 10 nodes, the last layer is the output layer which is of the dense type, the output is either 0 or 1 (presence of a defect or not), where the data was trained on this algorithm with a number of cycles (100 cycles), and three experiments were conducted for the same model to test the accuracy of the metrics.

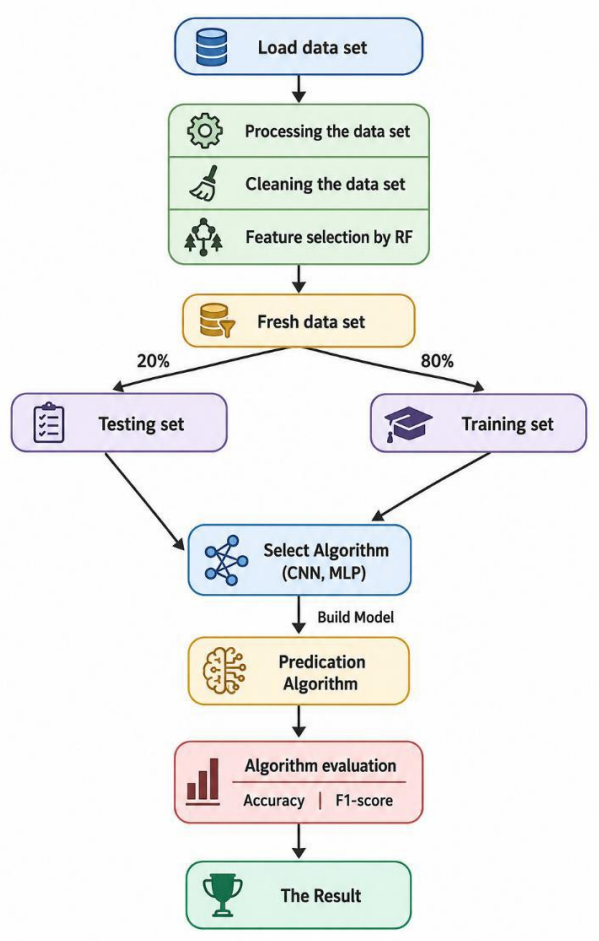

Figure 4: Methodology Workflow

Experiment 1: 20 features from software engineering metrics were entered, which are used to evaluate software quality and improve its performance; an accuracy of 62% was achieved.

Experiment 2: The features were reduced to 10 features and selected based on their importance, which is (cbo, rfc, ca, ce, lcom3, loc, mfa, cam, amc, avg-cc), and an accuracy of 69% was achieved.

Experiment 3: The features were reduced to 6 features and selected based on their importance, which are (CBO, RFC, ca, loc, cam, and AMC), and an accuracy of 71% were achieved. Through these three experiments, it was observed that the set of features in the third experiment achieved the highest accuracy in the result, and the importance of these features was obtained using the Random Forest classifier method (RF model).

The second algorithm is CNN, which consists of an input layer (data), 3 convolutional layers, and a pooling layer in which processing is done to extract important features automatically. In this layer, the resulting matrix is converted into a one-dimensional vector and then entered into the fully connected layers. The output layer is either 0 or 1 (presence of a defect or not). The metrics feature software engineering standards were entered, and this data was trained and tested. The number of cycles was 5 cycles).

The CNN algorithm achieved better results than the MLP algorithm, as shown in Table 5.

The Relu activation function was used in the hidden layers, and the Sigmoid activation function was used in the outputs. The performance measures (F1-score, and accuracy) were used to measure the results for each group of the data set used in both algorithms, as well as to measure the accuracy of these algorithms in estimation using these measures [37],[38], which are:

- Accuracy is the total number of correct predictions divided by the total number of predictions made for the dataset. The best accuracy is 1, while the worst accuracy is 0 and can be calculated using the 1 equation [39].
- F1-score is the harmonic mean of precision and recall. The ideal model has an F-score of 1.

VIII. RESULTS AND DISCUSSION

SDP has attracted significant research attention due to its ability to reduce testing costs. While most existing studies have focused on classifying software modules as faulty, predicting software bugs is also useful. In this study, two deep learning models, MLP and CNN, were used to predict each software module, and their performance was evaluated on 11 software project datasets (ant, camel, ivy, xalan, velocity, log4j, jetit, poi, synapse, xerces, lucene), as shown in Table 4.

Table 4: Accuracy measure for each dataset in the MLP and CNN algorithms

Project	Version	Accuracy	
		CNN	MLP
Ant	1.3	0.93	0.8
	1.4	0.78	0.67
	1.5	0.78	0.9
	1.6	0.78	0.73
	1.7	0.76	0.81
Camel	1	0.96	0.96
	1.2	0.76	0.59

International Research Journal of Innovations in Engineering and Technology (IRJIET)
ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

	1.4	0.57	0.6
	1.6	0.6	0.49
Ivy	1	0.86	0.48
	1.1	0.87	0.84
	1.2	0.57	0.85
Xalan	2.4	0.82	0.82
	2.5	0.35	0.64
	2.6	0.5	0.71
	2.7	0.35	1
Velocity	1.4	0.29	0.85
	1.5	0.35	0.72
	1.6	0.65	0.85
Log4 j	1	0.74	0.78
	1.1	0.66	0.86
	1.2	0.94	0.88
Jedit	3.2	0.57	0.78
	4	0.65	0.73
	4.1	0.57	0.79
	4.2	0.57	0.89
	4.3	0.66	0.97
Poi	1.5	0.76	0.71
	2	0.5	0.86
	2.5	0.85	0.74
	3	0.6	0.72
Synapse	1	0.94	0.91
	1.1	0.77	0.71
	1.2	0.73	0.79
Xerces	1.1	0.35	0.67
	1.2	0.35	0.72
	1.3	0.35	0.87
	1.4.4	0.92	0.85
Lucene	2	0.57	0.59
	2.2	0.57	0.56
	2.4	0.56	0.75

Table 5 shows the results obtained using the performance measures (F1-score, Accuracy), where the CNN algorithm achieved a better rate of results than the MLP algorithm in predicting software defects.

Table 5: Comparison of the performance results of the two models

MODEL	Experiment1			Experiment 2			Experiment 3		
	NO. of features without RF	Accuracy	F1	NO. of features with RF	Accuracy	F1	NO. of features with RF	Accuracy	F1
MLP	20	62%	62.5	10	69%	69.7	6	71%	71.6%
CNN	20	87%							

Successful results were achieved using a CNN-based deep learning model with multiple layers in defect prediction compared to the MLP algorithm in terms of accuracy rate and F1 score, where CNN showed 86% accuracy rate and 87.7% F1 score while MLP showed 71% accuracy rate and 71.6% F1 score, as shown in Figure 5.

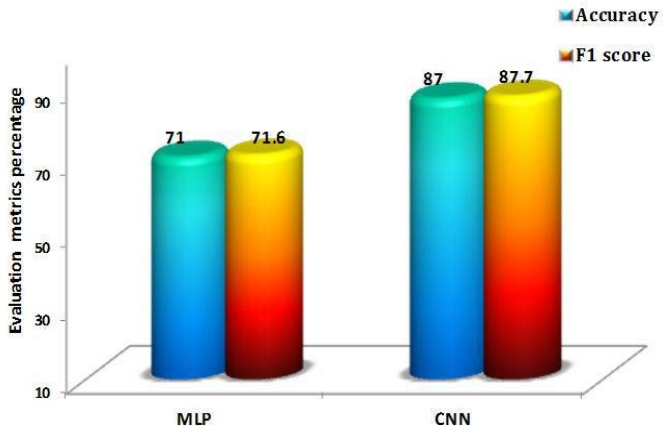

Figure 5: Compare results between MLP, CNN

IX. CONCLUSIONS

In this research, two deep learning algorithms were used to predict software defects (CNN, MLP) and 11 datasets from the PROMISE website were used. CNN algorithm achieved better results than the MLP algorithm in terms of accuracy rate and F1 score; its accuracy reached 86% and 87.7% for F1 score, while MLP showed accuracy results of 71% and 71.6% for F1 score. Future work plans include expanding this approach to include techniques that take into account the semantics of code while predicting defective modules.

ACKNOWLEDGMENTS

I would like to express my deepest thanks and gratitude to the Deanship of the College of Computer and Mathematics at the University of Mosul for their assistance and facilitation of research requirements.

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

REFERENCES

[1] I.A. qbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, and A. Husen. "Performance analysis of machine learning techniques on software defect prediction using NASA datasets", *International Journal of Advanced Computer Science and Applications*, Vol. 10, No. 5, 300–308, 2019.

[2] S. K. Pandey and A. K. Tripathi, "SDC-estimator: an effectual software defect count estimation technique for the upcoming version of software project. *Innovations in Systems and Software Engineering*", 1-36, 2025.

[3] F. Provost and T. Fawcett, "Data science and its relationship to big data and data-driven decision-making Big data", Vol. 1, No. 1, 51-59, 2013.

[4] C. W. Yohannese and T. Li, "A combined learning-based framework for improved software fault prediction", *International Journal of Computational Intelligence Systems*, Vol. 10, No. 1, pp. 647, 2017.

[5] S. Sahel, M. Alsahafi, M. Alghamdi, and T. Alsubait, "Logo detection using deep learning with pre-trained CNN models", *Engineering, Technology & Applied Science Research*, Vol. 11, No. 1, pp. 6724–6729, 2021.

[6] A.Alsaeedi and M. Z. Khan, "Software defect prediction using supervised machine learning and ensemble techniques: A comparative study", *Journal of Software Engineering and Applications*, Vol. 12, No. 5, pp. 85–100, 2019.

[7] E. E. Miandoab and F. S. Gharehchopogh, "A novel hybrid algorithm for software cost estimation based on Cuckoo optimisation and K-Nearest Neighbors algorithms", *Engineering, Technology & Applied Science Research*, Vol. 6, No. 3, pp. 1018–1022, 2016.

[8] S. C. Satapathy, A. K. Jena, J. Singh, S. Bilgaiyan, D. Ghosh, and J. Singh, "A novel approach of software fault prediction using deep learning technique", *in Automated Software Engineering: A Deep Learning-Based Approach*, pp. 73–91, 2020.

[9] D. Mnyanghwalo, H. Kundaeli, E. Kalinga, and N. Hamisi, "Deep learning approaches for fault detection and classifications in the electrical secondary distribution network: Methods comparison and recurrent neural network accuracy comparison", *Cogent Engineering*, Vol. 7, No. 1, PP. 1857500, 2020.

[10] O. A. AL Qasem and M. Akour, "Software fault prediction using deep learning algorithms", *International Journal of Open-Source Software and Processes*, Vol. 10, No. 4, pp. 1–19, 2019.

[11] H. M. Yahya and D. B. Taha "The Development of the Secure Quality Dataset (SQDS): Combining Security and Quality Measures Using Deep Machine Learning for Code Smell Detection", *International Journal of Computing and Digital Systems*, Vol. 16, No.1, 995-1006 ,2024.

[12] S. K. Pandey, A. Haldar, and A. K. Tripathi, "Is deep learning good enough for software defect prediction", *Innovations in Systems and Software Engineering*, pp. 1–16, 2023.

[13] T. Hoang, H. K. Dam, Y. Kamei, D. Lo, and N. Ubayashi, "Deepjit: An end-to-end deep learning framework for just-in-time defect prediction", *in IEEE/ACM 16th International Conference on Mining Software Repositories*, 2019.

[14] A.Al-Mansoori and A. Al-Hamadi, "Sustainable practices in the manufacturing sector: Challenges and opportunities", *Sustainability*, Vol. 15, No. 1, pp. 234–250, 2023.

[15] W. Alkaberi and F. Assiri, "Predicting the number of software faults using deep learning", *Engineering, Technology and Applied Science Research*, Vol. 14, No. 2, pp. 13222–13231, 2024.

[16] A.Kukkar, R. Mohana, Y. Kumar, A. Nayyar, M. Bilal, and K. S. Kwak, "Duplicate bug report detection and classification system based on deep learning technique", *IEEE Access*, Vol. 8, pp. 200749–200763, 2020.

[17] M. Nevendra and P. Singh, "Software defect prediction using deep learning", *Acta Polytechnica Hungarica*, Vol. 18, No. 10, pp. 173–189, 2021.

[18] K. K. Sharma, A. Sinha, and A. Sharma, "Software defect prediction using deep learning by correlation clustering of testing metrics", *International Journal of Electrical and Computer Engineering Systems*, Vol. 13, No. 10, pp. 953–960, 2022.

[19] X. Chen, D. Zhang, Y. Zhao, Z. Cui, and C. Ni, "Software defect number prediction: Unsupervised vs supervised methods", *Information and Software Technology*, Vol. 106, pp. 161–181, 2018.

[20] N. H. Qazan and S. M. Al-Mashhadany, "Performance Evaluation of the Plagiarism Systems a Systematic Review", *International Journal of Electrical Engineering and Intelligent Computing*, Vol. 1, No.1, 15-23,2023.

[21] R. S. Gu, "Data cleaning framework: An extensible approach to data cleaning", 2011.

[22] M. Jureczko, "Significance of different software metrics in defect prediction", *Software Engineering: An International Journal*, Vol. 1, No. 1, pp. 86–95, 2011.

[23] H. Motie Ghader, S. Gharaghani, Y. Masoudi-Sobhanzadeh, and A. Masoudi-Nejad, "Sequential and mixed genetic algorithm and learning automata (SGALA, MGALA) for feature selection in QSAR", *Iranian Journal of Pharmaceutical Research: IJPR*, Vol. 16, No. 2, p. 533, 2017.

ISSN (online): 2581-3048
Volume 10, Issue 7, pp 101-109, July-2026
https://doi.org/10.47001/IRJIET/2026.107011

[24] J. Miao and L. Niu, "A survey on feature selection", *Procedia Computer Science*, Vol. 91, pp. 919–926, 2016.

[25] H. A. Younis and Y. F. Mohammad, "Arabic speech recognition based on self-supervised learning", *In 16th International Conference on Developments in eSystems Engineering (DeSE)* pp. 528-533, 2023, *IEEE.*

[26] S. A. M. Al-Taie, and B. I. Khaleel, "Palmprint Identification Using Dolphin Optimization", *Periodica Polytechnica Electrical Engineering and Computer Science*, Vol.68, No.3, 295-308, 2024.

[27] 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.

[28] H. E. Solayman and R. P. Qasha, "Portable Modeling for ICU IoT-based Application using TOSCA on the Edge and Cloud", *In International Conference on Computer Science and Software Engineering (CSASE)*, pp. 301-305,2022, *IEEE.*

[29] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance", *Journal of Big Data*, Vol. 6, No. 1, 2019.

[30] S. S. Rathore and S. Kumar, "An empirical study of some software fault prediction techniques for the number of faults prediction", *Soft Computing*, Vol. 21, No. 24, pp. 7417–7434, 2017.

[31] Z. Vujovic, "Classification model evaluation metrics", *International Journal of Advanced Computer Science and Applications*, Vol. 12, pp.599–606, 2021.

[32] S. J. Mohammed and D. B. Taha, "Performance evaluation of RSA, ElGamal, and paillier partial homomorphic encryption algorithms", *In International Conference on Computer Science and Software Engineering (CSASE)*, pp. 89-94, 2022, *IEEE.*

[33] V. Vemuri, "The Hundred-Page Machine Learning Book", *Journal of Information Technology Case and Application Research*, Vol. 22, 2020.

[34] D. E. Birba, "A comparative study of data splitting algorithms for machine learning model selection", 2020.

[35] N. Palanivel, S. Deivanai, G. Lakshmi Priya, and B. Sindhuja, "Biosignals based smoking status prediction using standard autoencoder and artificial neural network", *International Journal of Computing and Digital Systems*, Vol. 17, No. 1, pp. 1–14, 2024.

[36] F. Emmert-Streib, S. Moutari, and M. Dehmer, "A comprehensive survey of error measures for evaluating binary decision making in data science", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Vol. 9, e1303, 2019.

[37] B. Sun, Y. S. Sun, T. Ban, T. Takahashi, S. C. Chang, and D. Inoue, "A scalable and accurate feature representation method for identifying malicious mobile applications", *Proceedings of the ACM Symposium on Applied Computing*, Part F1477, pp. 1182–1189,2019

[38] A.Alashqar, "A classification of Quran verses using deep learning", *International Journal of Computing and Digital Systems*, Vol. 16, No. 1, pp. 1041–1053, 2024.

[39] R. Özakıncı and A. Tarhan, "A decision analysis approach for selecting software defect prediction method in the early phases", *Software Quality Journal*, Vol. 31, No. 1, pp. 121–177, 2023.

Citation of this Article:

Ekhlas Tariq Hasan, & Shayma Mustafa Mohi-Aldeen. (2026). A Comparative Study of CNN and MLP for Software Defect Prediction Using PROMISE Datasets. *International Research Journal of Innovations in Engineering and Technology - IRJIET*, 10(7), 101-109. Article DOI https://doi.org/10.47001/IRJIET/2026.107011

****

