Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Software Defect Prediction Based on Deep Learning Algorithms: A Systematic Literature Review

Akhlas Tariq Hasan[1] and Shayma Mustafa Mohi-Aldeen[2]

[1,2] Department of Computer Science, College of Computer Science and Mathematics, University of Mosul, Iraq
Email: ekhlas.23csp58@student.uomosul.edu.iq[1] and shaymamustafa@uomosul.edu.iq[2]

Article information

Article history:
Received 23 December ,2024
Revised 03 February ,2025
Accepted 16 February ,2025
Published 26 June ,2025

Keywords:
Computer
Mathematics

Correspondence:
Author Name
Email: author1@round.com

Abstract

Software bug prediction (SDP) techniques identify bugs in the early stages of the software development life cycle through a series of steps to produce reliable and high-quality software. Deep learning techniques are widely used in SDP, which can produce accurate and exceptional results in different fields.

The study aims to systematically review models, techniques, datasets, and performance evaluation metrics to gain a complete understanding of current methodologies related to SDP, and the use of DL in software defect prediction research between 2019 and 2024. A comprehensive review of studies in this area was completed to answer the research questions and summarize the results from the initial investigations. 30 primary studies that passed the systematic review quality assessment of the studies were used. However, the six most common evaluation metrics used in SDP were confusion matrix, Scoar-1F, recall, precision, accuracy, and area under the curve (AUC). The top three DL algorithms used in building SDP models and used in predicting software bugs were convolutional neural network (CNN), long-short-term memory (LSTM), and bidirectional LSTM. We conclude that the application of deep learning in SDP remains a challenge, but it has the potential to achieve better prediction performance. Future research directions focus on improving these models and exploring their applications across diverse programming environments

DOI: 10.33899/csmj.2025.156086.1160,

1. Introduction

With the rapid development of computer technology, software applications have expanded to all parts of people's daily lives, creating a situation in which the economy, production, and life are fully dependent on computer software. But software failure can bring about serious or even fatal consequences, especially for high-risk systems.[1]

System failure is more often caused by software defects, which are important factors effecting software quality.[2]

At the internal level of software, defects are errors or faults in the software development or maintenance process; A fault is an error that has effects on system behavior, at the external level of software, defects are violations or failures of the functions that the software needs to perform.[3]

Software Defect Prediction (SDP) as a crucial technique that aids developers in identifying defective software modules in advance, allowing for more efficient allocation of testing resources through the analysis of software repositories and training predictive models on gathered data.[4]

Software reliability and quality mainly depend on removing faults or defects in software. Although some defects might arise from causes unrelated to code (such as compilers or byte code representations), the main source of software faults is software code. The traditional way of finding software defects is by testing and conducting reviews. However, these activities may require extensive time and effort. On the other hand, automatic prediction of defective software modules at early stages may guide developers in improving code quality at a reduced cost compared to a fully manual approach ,Predicting defect-

prone parts of software before discovering faults by performing substantial efforts is a challenging task. The main challenge of SDP is identifying the faulty parts of source code with better fault prediction performance.[5]

Deep Learning is subfield of machine learning that uses supervised and unsupervised strategies. It has been very successful in various fields successful in various such as computer vision, Natural language processing. Deep learning enables computational models made of multiple layers to learn data representations at multiple levels of abstraction.[6]

Deep learning is chosen for its ability to capture complex from large datasets.[7]

The advantages of deep learning over machine learning are that deep learning has best-in-class performance, has the ability to extract features automatically and eliminates the feature engineering stage, and makes it easy to generalize the trained model to other domains. Deep learning is a rapidly growing research topic with many deep learning architectures, and new models are being developed to suit different research domains.[8]

In general, the utility of deep learning becomes better as the amount of training data increases. As a result, the ability to solve complex applications and its accuracy are constantly increasing. Deep learning is beating the AI community by making improvements to solving problems and will lead to more success in the future because it requires very little manual engineering.[9]

This paper is organized as follows: Section 2 describes the Research Questions, Section 3 describes the Algorithms Used in Software Defect Prediction, Section 4 Discussion of the Research Questions, Section 5 describes the Challenges in Applying Deep Learning, Section 6 describes the Practical applications of prediction techniques in real world programs, Section 7 describes the Related Works, Section 8 describes the Conclusion.

2. Research Questions

The aim of this study is to obtain a presentation and overview of current research in the field of software bug prediction using deep learning techniques that enable the system developer to create a set of high-quality tests that have the ability to detect software errors or defects and evaluate the quality of these techniques through the following questions :

RQ. 1 what are the main components of software defect prediction)SDP(?

RQ. 2 What kind of metrics are the most used for fault prediction?

RQ. 3 What are the types of software defect prediction(SDP)?

RQ. 4 What kind of methods are the most used for fault prediction?

2.1. Sources Of Information

This paper presents a systematic literature review of the work done in software defect prediction using deep learning, and in order to have a broad view, many papers and journals were searched and publications related to this study were selected in the time period from 2019 to 2024. The search strategy was based on the identification of alternative words and synonyms of terms used in research questions to decrease the effect of the differences in terms After selecting the publications related to the study in this period, 30 closely related articles in software defect prediction were found. **Figure 1** illustrates the steps for integrating deep learning with software defects.

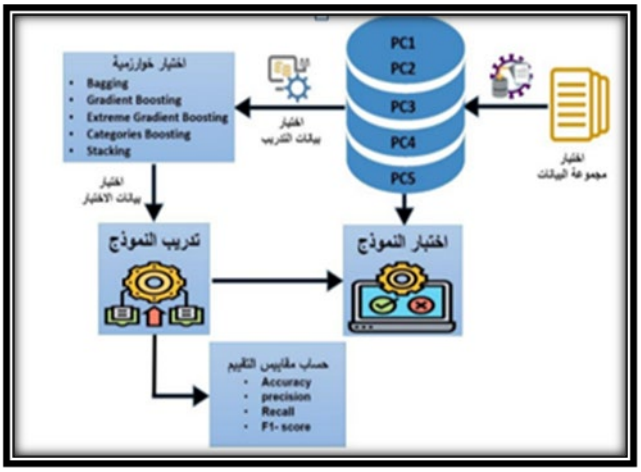

Figure 1. Integrating Deep Learning with Software Defects [10]

2.2. Software defect prediction

Defect prediction is a technique used in software engineering to identify and predict defects and errors in software systems before they occur.[11] The goal of defect prediction is to improve software quality and reduce the number of post-release problems by enabling developers to focus their efforts on high-risk areas. The benefits of defect prediction include the following:

Early detection of errors: Defect prediction allows developers to discover potential problems early in the development process, reducing the cost and effort required to fix them.

Resource optimization: By focusing efforts on high-risk areas, resources can be used more efficiently, leading to better software quality.

Decision support: Project managers can use defect prediction results to make informed decisions about resource allocation and release planning.

Process improvement: Defect data analysis can provide insight into the software development process, enabling

organizations to identify areas for improvement.[12]

2.3. Deep Learning in Software Defect Prediction

Deep learning is known as a subcategory of machine learning techniques. In this type of learning, new architectures are used where multiple layers of processing units are employed to extract features and it is used as a new solution for most of the fields especially in software topics.[13]

Deep learning is used in predicting software bugs because it is able to process large amounts of data and identify complex patterns. Through the training process, deep learning models can learn to recognize signs of potential errors or vulnerabilities in software code, thus improving software quality and reliability[14]. It can also be used in anomaly detection, code review, and predictive maintenance. Deep learning began to be incorporated into software bug prediction when developers sought more efficient ways to analyze large databases and identify potential problems before they appeared in production. Traditional methods often rely on rule-based systems or simpler statistical techniques, which can be limited in scope. Deep learning, with its ability to learn from unstructured and imbalanced data and adapt to new patterns, allows for more accurate predictions and improved decision-making in software development processes. It is therefore a vital tool for improving software reliability and reducing the time and cost associated with patching and maintenance.[15]

Deep learning is an effective tool for predicting software defects, which contributes to improving software quality and reducing the costs associated with fixing errors.[16] .

One of the main benefits of deep learning is the automatic extraction of features from data, which saves time and effort , Deep learning models show superior performance in processing large and complex data, and it also has the ability to deal with unbalanced and unstructured data more effectively [12].Which increases the accuracy of predictions.

2.4. Comparison Between Traditional Methods and Deep learning

Traditional techniques and deep learning are two different approaches in the field of artificial intelligence, traditional methods rely heavily on designing a pre-defined model based on algorithms and statistical analysis of data, in contrast, deep learning relies on artificial neural networks that are able to learn directly from big data. The main differences between the two approaches lie in the ability to handle data. Traditional methods suffer from performance limitations when dealing with complex and unstructured data and require hand-designed features, deep learning, on the other hand, can process large amounts of data more efficiently and extract features automatically, making it suitable for applications such as image and video processing. [16]

3. Algorithms Used in Software Defect Prediction

This paper presents deep learning algorithms and traditional machine learning techniques to enhance software defect prediction (SDP).

3.1. Convolutional Neural Networks (CNNs)

CNNs are designed to process structured data using convolutional layers to automatically learn features from the input data. By automatically extracting features from the data, this reduces the need for manual feature engineering, which can be time-consuming [8].This model can recognize patterns regardless of their position in the input data. Additionally, CNNs learn features at multiple levels and consist of an input layer, an output layer, convolutional layers, pooling layers, and fully connected layers, making them adept at extracting features automatically, as shown in **Figure 2.[**17]

A given neuron has the same number of input neurons and the same weight, and the pooling layer reduces the size of the input neuron and increases the learning rate.[18]

CNNs are widely used in image processing, video recognition, and natural language processing. Their advantages include higher accuracy and efficiency in handling spatial sequences of data, including automatic feature learning and performance. They are easier to train and have fewer parameters compared to other deep learning architectures. However, they typically require larger datasets for training, a lot of time, and can be computationally intensive.[19],[20]

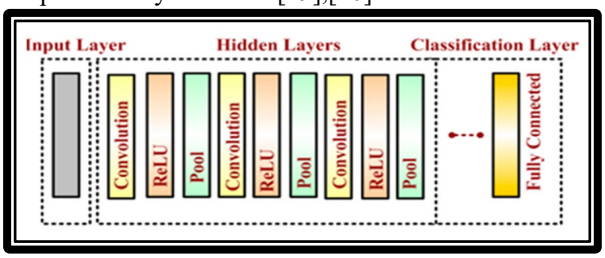

Figure 2. CNN architecture [17]

3.2. Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) algorithm is a type of recurrent neural network (RNN) designed to efficiently learn long-term dependencies in sequential data. LSTMs work by processing sequences of data step by step. LSTMs consist of memory cells that can retain information over extended periods, using three gates: an input gate, a forget gate, and an output gate ,as shown in **Figure 3** .The input gate controls the

flow of new information, the forget gate determines what information should be discarded, and the output gate determines what information should be passed to the next layer. It has the ability to mitigate the vanishing gradient problem, making it suitable for time series forecasting (such as wind speed forecasting). It can be computationally intensive and requires fine-tuning of parameters, and it updates its internal state based on both the current input and the previous hidden state, allowing it to make accurate predictions.[21]

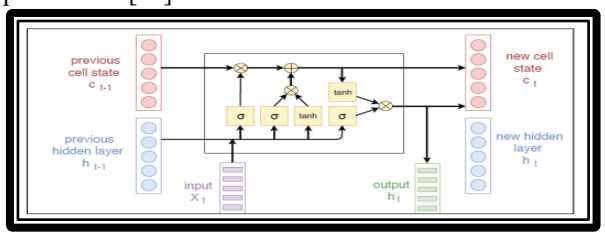

Figure 3. LSTM architecture [22]

3.3. Bidirectional LSTM (BI-LSTM)

The Bidirectional LSTM (BI-LSTM) algorithm is an extension of the LSTM algorithm described above that uses two LSTM algorithms on the input data. The input sequence is fed to the LSTM algorithm in the first round (i.e. the forward layer) and the reverse form of the input sequence is fed into the algorithm in the second round in the backward layer.Using two LSTMs improves the learning of long-term dependencies and as a result improves the accuracy of the model. BI-LSTM runs faster and takes less time to make predictions, **Figure 4.** Illustrates the architecture of BI-LSTM.[23]

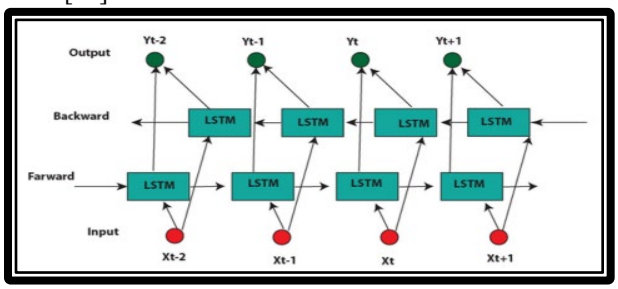

Figure 4. BI-LSTM architecture [23]

3.4. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed for processing sequences of data. Unlike traditional feed forward neural networks, RNNs have connections that loop back on themselves, allowing them to maintain a memory of previous inputs. This architecture makes RNNs particularly effective for tasks involving sequential data, such as time series analysis, natural language processing, and speech recognition, **Figure 5**. Illustrates the architecture of RNN.[24]

Advantages include improved accuracy in predictions, automatic feature extraction, and the ability to handle large volumes of data and real-time monitoring. However, they also have disadvantages, such as requiring substantial labeled training data, being computationally intensive, and lacking interpretability, which makes understanding their decision-making process challenging.[25]

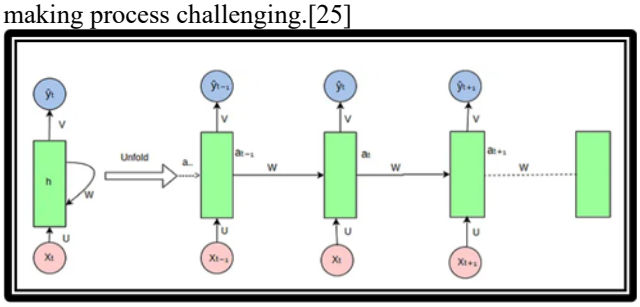

Figure 5. RNN architecture [22]

3.5 Multi-Layer Perceptron (MLPs)

MLPs consist of an input layer, multiple hidden layers, and an output layer, where each neuron in one layer connects to every neuron in the next, as shown in **Figure 6** .This structure allows MLPs to model complex relationships and detect patterns in data. Advantages include their ability to learn non-linear functions and flexibility in various applications. However, they can require extensive feature engineering and may struggle with large datasets due to over fitting.[9]

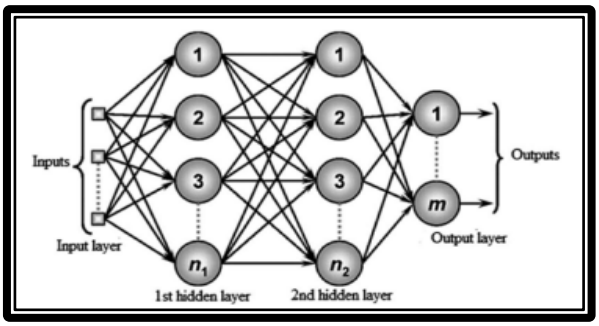

Figure 6. MLP architecture [26]

3.6 Deep Neural Networks (DNN)

Deep neural networks (DNNs) are used to predict the severity of software defects. The algorithm for this type consists of multiple layers of neurons that process the input data, enabling the model to learn complex patterns from defect reports. This approach is characterized by high accuracy and the ability to handle large data sets, making it suitable for complex tasks such as defect classification. Additionally, However, drawbacks include the need for large computational resources and the potential for over fitting if not properly organized. The algorithm works by converting textual defect descriptions into numerical vectors, which are then processed by the DNN to predict severity levels, with a neuron coverage measure used to assess the model's effectiveness in exploring the input space, **Figure 7** illustrates the architecture of DNN.[27]

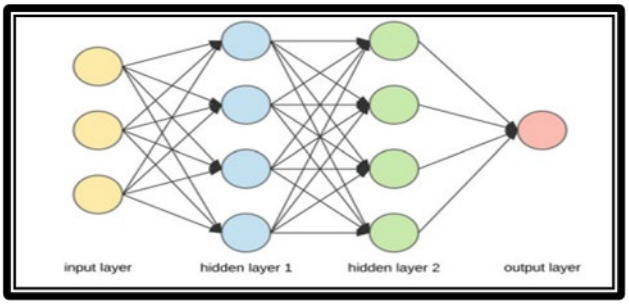

Figure 7. DNN architecture [28]

3.7 Deep Graph Neural Networks (DGNN)

We present DeepWukong, a new deep learning based approach that embeds both textual and code structured features into an effective representation to support detection of a wide range of vulnerabilities. DeepWukong first performs program slicing to extract fine-grained but complicated semantic features, and then combines with graph neural networks to produce compact and low-dimensional representation, that DeepWukong outperforms several state-of-the-arts, including traditional vulnerability detectors and deep-learning-based approaches, **Figure 8** shows the DGNN architecture.[29]

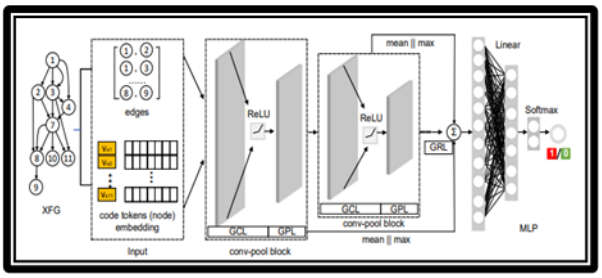

Figure 8. the structure of the graph neural network in Deep wukong [29]

3.8 Gated hierarchical long short-term memory (GH-LSTM)

Gated Hierarchical Long Short-Term Memory (GH-LSTM) Algorithm for Software Defect Prediction This algorithm consists of a hierarchical structure that uses two types of features: semantic features extracted from abstract syntax trees (ASTs) and traditional software metrics. The algorithm works by first processing the source code to extract both types of features, which are then fed into its LSTM networks. The outputs are combined using a closed-loop fusion mechanism, and the combined features are passed through a fully connected layer to predict whether a module is defective or clean. The advantages include improved prediction accuracy by leveraging semantic and traditional features that can capture different aspects of the code, leading to better defect detection, **Figure 9** shows the DGNN architecture The disadvantages include increased complexity in model training and the need for more computational resources.[30],[31]

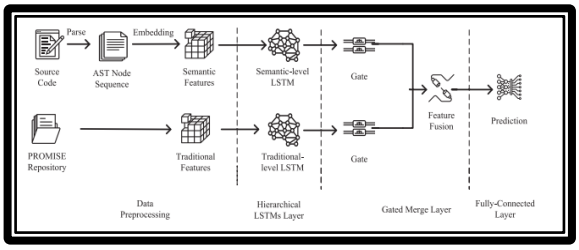

Figure 9. Overview of GH-LSTM [30]

3.9 Software Quality Assessment Deep Encoder Network (SQADEN)

The Deep Quadrilateral Advectional Regression Neural Network with Non-Parametric Statistical Measurement (SQADEN) for Software Fault Prediction. This algorithm consists of two main components: feature selection and classification. Feature selection uses a non-parametric Torgerson-Gower measurement technique to identify relevant software metrics while reducing time complexity. Classification is performed using a supervised Quadrilateral Advectional Regression deep neural network, which analyzes training and test samples to predict software defects, **Figure 10** shows the DGNN architecture. The advantages of SQADEN include improved prediction accuracy, reduced time and space complexity compared to existing methods, and the ability to handle high-dimensional datasets efficiently. However, disadvantages may include potential implementation complexity and dependence on input data quality for optimal performance. The algorithm works by identifying important features and then using deep learning techniques to classify software modules as defective or non-defective, ultimately achieving accurate predictions with minimal errors.[32]

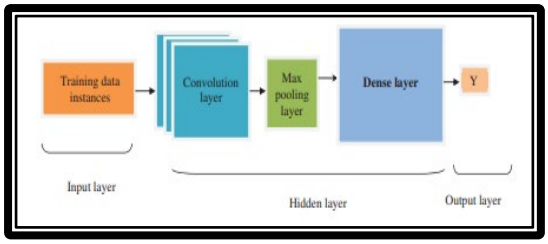

Figure 10. Schematic diagram of Quadratic Censored regressive convolution deep neural network[32]

4. Discussion of the Research Questions

This section discusses the answers from the researches that described the research questions .

4.1. Research question 1: What are the main components of SDP?

In the SDP process, three main components are relied upon: the dependent variables, the independent variables and the model, The dependent variables are the defect data

of a piece of code, either defective or non-defective, and can be binary or ordinal variables.

The independent variables (inputs) are the metrics in which the program code is recorded. The model contains the rules or algorithms that predict the dependent variable from the independent variables, The inputs (variables) are divided into training and test datasets to determine the effectiveness of the classifier, The training dataset is used to create the classifier. It is then used to predict potential defects in the test dataset and evaluate these predictions using different performance metrics to determine whether they are correct or not. **Figure 11**.

Software metrics play a fundamental role in SDP, and most SDP strategies rely on software metrics as independent variables, Metrics are designed to support bug finding in software projects, Given the huge diversity of software applications, identifying, locating, and detecting software defects has become a daunting task for researchers, Moreover, the density of defects also poses a challenge in detecting and predicting software defects. Typically, faulty software databases are composed of naturally imbalanced data, which generates randomness in the pattern properties, this motivates the development of an efficient and accurate SDP model.[33],[34]

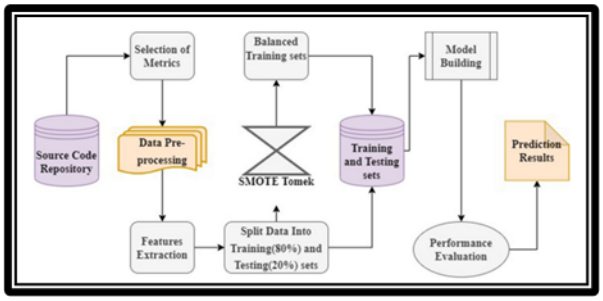

Figure 11. Proposed method of SDP[33]

4.2 Research question 2: What kind of metrics are the most used for fault prediction

Metrics actually aim to measure the accuracy of algorithms in estimating, by comparing the actual evaluation results derived from the dataset of predicting software defects with the expected evaluation resulting from applying the algorithms using a set of evaluation metrics. Several different evaluation metrics are used, among which five metrics are prominent and widely used and have been used in These metrics are based on a matrix (Accuracy, precision, Recall, F1-Score) and include the Confusion Matrix (Confusion Matrix) which is a table that displays the prediction results for classifying software defects, summarizing the correct and incorrect prediction values by comparing them with the training values which are described as true and false with the prediction values which are described as positive and negative as shown in **Figure 12.[**12]

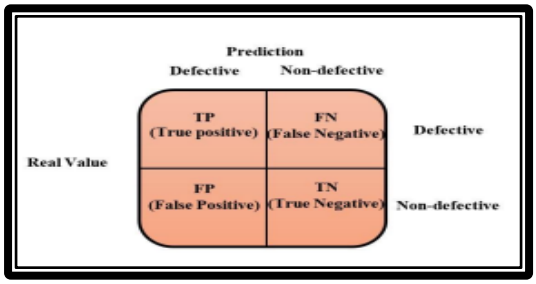

Figure 12. shows the confusion matrix for predicting software defects [33]

The confusion matrix, includes the terms below:

1. : (True Positive (TP It is the true positive expectation of a software defect, i.e. if there is a software defect and it is expected to be a software defect.

2. :(TN)True Negative It is the true genetic prediction of the software defect, i.e. if there is no software defect and its prediction is no software defect .

3. (False Positive (FP : It is a false positive prediction of a software defect, i.e. if there is no software defect and it is predicted to be a software defect.

4. False Negativ (FN(: It is a false negative prediction of a software defect, i.e. if there is a software defect and it is predicted, there is no software defect.[34],[35]

Accuracy

It is the total number of correct predictions divided by the total number of predictions made on the data set. The best accuracy is 1 while the worst accuracy is 0, which can be calculated using the following equation:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

precision

It is the percentage of correct positive predictions (TP) divided by the total number of positive predictions. The best accuracy is 1 and the worst accuracy is 0, which can be calculated using the following equation:

$$\text{precision} = \frac{TP}{TP+FP}$$

Recall

It is the ratio of true positive predictions (TP) divided by true positive predictions plus false negative predictions, which can be calculated using the following equation:

$$Recall = \frac{TP}{TP+FN}$$

F1- Score

It is the harmonic mean of the evaluation accuracy and recall, which can be calculated using the following equation.[36]

$$F1 - score = 2 * \frac{precision*Recall}{precision+Recall}$$

AUC

It indicates the relative performance of the True Positive Rate (TPR) and False Positive Rate (FPR). The greater the area under the Receiver Operating Characteristic (ROC) curve, the higher the model's.[33]

4.3 Research question 3: What are the types of software defect prediction?

Early detection and prediction of software defects plays an important role in modern software development. To address this problem of software defect prediction, software defect prediction can be classified into two main types:

1. Within-project defect prediction (WPDP)

This type involves prediction of defects within the same project where the model is trained and tested on data from the same code base and allows it to take advantage of the specific characteristics and patterns present in that project.

2. Cross-project defect prediction (CPDP)

This type aims to generalize predictions across different software systems where the model is trained on data from one project and tested on a different project which is more challenging due to the differences in code structures and characteristics between projects.[36],[37],[38]

4.4 Research question 4: What kind of methods are the most used for fault prediction?

Deep learning algorithms are collected after a comprehensive study of the research in Table 1, which shows a set of researches specialized in deep learning algorithms.

The percentage of each algorithm was calculated according to previous studies.

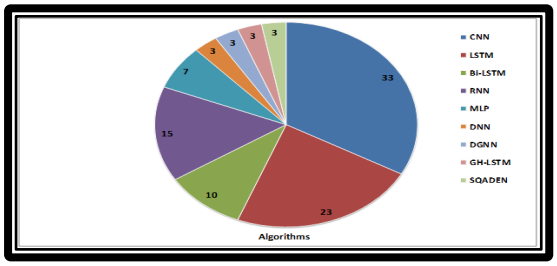

Figure 13. Percentage of Algorithms in previous studies

The **Figure 13** above shows the distribution of different algorithms used, where CNN represents the largest share with 33% of the total due to its high ability to extract spatial features, followed by LSTM algorithm with 23% due to its ability to handle sequential data, RNNs with 15% have connections that loop back on themselves, allowing them to maintain a memory of previous inputs, BI-LSTM with 10% it uses information in both directions, and many other algorithms that constitute smaller percentages. This **Figure** provides a visual breakdown of the relative use or prevalence of these different algorithms. These higher percentages for these three algorithms are due to their proven effectiveness in predicting software errors based on their ability to handle complex data efficiently.

5. Challenges in Applying Deep Learning to software defect prediction

The challenges associated with the use of deep learning are important issues that must be taken into consideration when applying this technology in various field In software development, predicting defects in software engineering is still a major challenge, leading to system failure, increasing maintenance costs, and making the development process more difficult, but they are not insurmountable by understanding these challenges and applying appropriate strategies for them. The most prominent of these challenges are:

1. The problem of imbalanced data In many cases, the data used to train deep learning models is imbalanced, which is one of the most prominent challenges facing deep learning models in the field of software defect prediction, in many software projects, the number of defects is much less compared to defect-free software, this imbalance can lead to model bias, the model may tend to predict the most common data category (defect-free software) and neglect the less common category (defects), leading to low accuracy in prediction, also, the evaluation is inaccurate, traditional metrics such as accuracy may appear good, while the actual performance in defect recognition is poor, this problem can be solved by using rebalancing techniques and applying special algorithms that deal with the problem of imbalance.

2. The problem of the need for huge computational

resources

Deep learning models require large computational resources, which represents a real challenge, especially in environments that lack the necessary resources, including time and cost, deep learning model training processes require a long time and high costs, which is impractical for many organizations, these models also require a powerful infrastructure in terms of computing power, such as graphics processing units (GPUs) or clouds, which can be expensive and difficult, this problem can be solved using cloud computing services (Google Cloud)that provide flexible and affordable computational resources.[21],[39]

3. The problem of difficulty in interpreting typical results

Interpreting typical results in deep learning is a challenge for researchers and developers and affects users' confidence, especially in the medical and security fields, to solve this problem, techniques such as (Attention Mechanisms) can be used and models of a simple nature can be designed that use the most influential features in decision-making and then trained on intermediate outputs that contribute to analyzing decisions, and cooperation between data scientists and specialists in this field, which contributes to increasing reliability, especially in sensitive fields.[40]

6. Practical Applications of Prediction Techniques in Real world programs

Many practical applications of prediction techniques in real-world software, where they are used in diverse areas such as predicting software bugs by analyzing changes in code and reducing errors in software development, which contributes to improving quality and reducing maintenance costs. In the financial sector, prediction models are used to analyze data and forecast market movements, helping investors make informed decisions, in healthcare, prediction techniques are used to analyze medical data and predict diseases, helping doctors provide better care to patients, in general, prediction techniques enhance the ability to make data-driven decisions, leading to improved performance and efficiency across various industries.

There are several suggestions for improving deep learning models in the future to meet industry needs.

Proposal 1: Integrating deep learning with traditional methods

To effectively meet industry needs, deep learning models should be integrated with traditional methods in defect prediction, this integration can improve prediction accuracy by leveraging the strengths of both methods, leading to more reliable models that are able to handle changing programming environments.[41],[42]

Proposal 2: Applying active learning and collaborating with industry

Building partnerships with industrial organizations to identify their specific needs, and ensuring that models meet these needs by collecting relevant data and modifying the input features, this strategy not only enhances the accuracy of models, but also ensures their compatibility with industry requirements, leading to more effective solutions in processing complex programming data.[43],[44]

7. Related Works

This section presents the previous works that used deep learning in SDP. Different methods of deep learning have been used in software defects

Table.1. Summary of Related Works
(The List of Studies in the Field of Software Defect Prediction)

No	Authors	Objective of the study	Methodology/ approaches/ tools/techniques used	Metrics	Data set	Key Finding	Accuracy
1	Hoang *et al*., 2019	for just-in-time defect prediction	developed DeepJIT using (CNN)	AUC	QT and OPENSTACK	achieving improvements of 10.36-13.69% in Area Under the Curv e (AUC)	-
2	Deyu Chen *et al*., 2019	for cross-project defect prediction	proposed the DeepCPDP method using (Bi -LSTM) with the Sim AST representation for cross-project defect prediction	AUC	PROMISE	The results showed that DeepCPDP significantly outperformed eight state-of-the-art baselines, achieving an average performance improvement of 6.18% to 21.17%.	-
3	Al Qasem and Akour, 2019	investigated software fault prediction using deep learning algorithms	(MLPs) and (CNNs)	Accuracy	NASA	the CNN algorithm outperformed MLPs, achieving accuracies of up to 100% on some datasets	98%
4	Liang *et al*., 2019	for software defect prediction	proposed the Seml framework using a Long Short Term Memory (LSTM) algorithm	prediction Recall F1- score	1,306 open-source Java projects	that Seml outperformed state-of-the-art approaches in both within-project and cross-project defect prediction	49.81%
5	Lei Qiao *et al.*,	defect prediction	using a specially designed	MSE	MIS and KC2	Their approach	-

No	Authors	Objective of the study	Methodology/ approaches/ tools/techniques used	Metrics	Data set	Key Finding	Accuracy
	2019		neural network (DPNN model)			outperformed existing models, reducing mean square error by over 14% and increasing correlation by more than 8%.	
6	Ahmad Hasanpour *et al*., 2020	or software defect prediction.	Using tack Sparse Auto-Encoder (SSAE) and Deep Belief Network (DBN) most evaluation metrics	prediction Accuracy	NASA	SSAE model achieved better results than DBN in most evaluation metrics enhancing prediction accuracy.	83.8%
7	Cheng *et al*.,2020	static detection of software vulnerabilities in C/C++ programs	DeepWukong (DGNN)	Accuracy F1- score	dataset of 105,428 real-world programs	demonstrating superior performance compared to traditional static detectors and existing deep-learning approaches.	86%
8	Ghosh and Singh, 2020	Prediction for software fault effectively calculated suspicious cores for program statements	(CNN)		dataset comprising test case results and statement coverage from example programs	approach effectively calculated suspicious scores for program statements, improving fault localization accuracy	-
9	Chakraborty *et al*., 2020	vulnerability detection	used a graph-based model with representation learning	Accuracy precision recall F1- score	new dataset from real-world projects (Chromium and Debian)	study achieved a precision improvement of up to 33.57% and a recall increase of 128.38%	93%
10	Deng et al., 2020	proposed a defect prediction framework	(LSTM) leveraging Abstract Syntax Trees (ASTs)	F1- score	seven open-source projects from the PROMISE	LSTM approach outperformed traditional and state-of-the-art methods	-
11	Kukkar *et al*., 2020	developed a duplicate bug report detection system	(CNN)	Accuracy Recall F1- score	six publicly available datasets	achieving an accuracy rate between 85% and 99%	91%
12	Kumar *et al*., 2020	developed a defect severity prediction model	using various deep learning	Accuracy F1- score AUC	six software projects	achieving high predictive power with AUC values close to 1 when using SMOTE for data balancing	80%
13	Mnyanghwalo *et al*., 2020	for fault detection in electrical secondary distribution networks	(RNN)	Accuracy AUC	dataset collected from a low-voltage transformer in Tanzania between 2012 and 2020	found that RNNs were efficient in detecting and classifying faults, with accuracy improving as complexity increased.	95.6%
14	Kumar *et al*., 2021	predicting software defect severity levels	using various embedding and feature selection techniques	Accuracy AUC Neuron coverage	six software projects (CDT, JDT, PDE, Platform, Bugzilla, Thunderbird)	that models utilizing word embeddings and SMOTE significantly improved prediction accuracy and neuron coverage	-
15	Yu et al., 2021	Defect prediction within and across projects	developed model called DPSAM, utilizing a self-attention mechanism (DBN) (DP-CNN)	Accuracy Precision Recall F1- score	seven open-source Java projects PROMISE	achieving notable F1 score improvements of 16.8% in within-project defect prediction and 23% in cross-project defect prediction	66.8%
16	Bani-Salameh et al., 2021	detecting the priority of bug reports and and allows developers to find the highest priority bug reports	(RNN-LSTM)	Accuracy AUC F1- score	2000 bug reports from JIRA	achieved an accuracy of 90.8%, outperforming other algorithms like KNN (74%) and SVM (87%)	90.8%
17	Nevendra and Singh, 2021	The aims to identify the defective instance using the enhanced deep learning method	(CNN)	Accuracy Precision Recall F1- score	19 open-source defect datasets	approach significantly outperformed Li's CNN and standard machine learning models	77,5%

No	Authors	Objective of the study	Methodology/ approaches/ tools/techniques used	Metrics	Data set	Key Finding	Accuracy
18	Wang et al., 2021	the goal GH-LSTMs model to extract both semantic and traditional software features and effectively combine the extracted features using gated fusion mechanism. and demonstrate that proper feature fusion can significantly boost the performance of defect predictio	(GH-LSTMs)	Precision Recall F1- score	10 open-source projects from PROMISE	GH-LSTMs outperformed existing methods	51%
19	Elsaraiti and Merabet, 2021	has been proposed to forecast wind speed	(LSTM) (RNN)		forecast wind speed using hourly data from Halifax, Canada	LSTM model significantly improved prediction accuracy, achieving RMSE values of 8.5128 for spring and 4.7796 for summer	-
20	Liu et al., 2021	study and analysis of software defect prediction methods in a cloud environmen Autoencoder models in deep learning theory can automatically learn features from the original data and obtain feature representations of the input data	cost-sensitive deep ladder network algorithm (CSDLN)	Accuracy F1- score	used a dataset from various projects	found that their improved deep belief network method achieved better prediction accuracy compared to traditional models	-
21	Sharma *et al*., 2022	predict the regions of source code that contain faults	(CCFT-CNN) (CNN)	F1- score	PROMISE	2% improvement in F-measure over baseline models	-
22	Rizvi, 2023	power outage prediction and fault detection	(CNNs) (RNNs) (GANs)	Accuracy Precision	-	achieved a 95% accuracy in predicting outages and 92% average precision in classifying fault types	95%
23	Borandag et al., 2023	research was to statistically demonstrate that DL algorithms outperformed ML algorithms	(RNN) (CNN) (LSTM) (Bi-LSTM)	ACC AUC	SFP XP-TDD and e Eclipse and Apache Active MQ	They found that deep learning algorithms outperformed traditional machine learning techniques	-
24	Zain et al., 2023	To synthesize literature on SDP using DL, pertaining to measurements, models, techniques, datasets, and achievements	(CNN) (DNN) (LSTM) (DBN) (SDAE)	Accuracy Precision Recall F1- score AUC MCC PRC	PROMISE and NASA	indicated that DL models generally outperformed traditional Machine Learning models in terms of accuracy, f-measure, and AUC	-
25	Giray et al., 2023	tudy, identify, analyze and summarize the current state of use of deep learning algorithms for SDP	(CNN) (RNN) (LSTM) (GRU) (MLP) (DBN)	-	analyzing 102 studies	the most frequently used DL algorithm is CNN. The other widely used algorithms are RNN/LSTM/GRU, MLP, and DBN	-
26	Batool and Khan, 2023	identify faults at the early stages of the software development life cycle (SDLC)	(LSTM) (Bi-LSTM) (RBFN)	Accuracy Precision Recall F1- score	CK metrics-based datasets And Git repository	found that LSTM and BILSTM performed better in accuracy, while RBFN was faster	93%
27	Lv, 2024	identifying and anticipating mechanical failures is explored through an examination of vibration datasets sourced from actual industrial machinery	(CNN – LSTM)	Accuracy Recall F1- score	vibration datasets from industrial machinery	achieved an accuracy of 95% outperforming traditional methods like SVM and Random Forest	95%
28	Alkaberi and Assiri, 2024	determine whether a software unit is faulty	(CNN) (MLP)	Kendall MSE	twelve open-source software project from the PROMISE	MLP achieved a Kendall value of 0.416 and a mean squared error (MSE) of 0.195, outperforming the CNN	-

No	Authors	Objective of the study	Methodology/ approaches/ tools/techniques used	Metrics	Data set	Key Finding	Accuracy
29	Khleel and Nehéz, 2024	aims to combine a bidirectional long short-term memory (Bi-LSTM) network with oversampling techniques. To establish the effectiveness and efficiency of the proposed model	(Bi-LSTM)	Accuracy Precision Recall F1- score Mcc AUC MSE The focus on Accuracy and F-measures	six public software defect datasets (ant, camel, ivy, jedit, log4j, and xerces) PROMISE	achieved average accuracies of 88%, 94%, and 92% on original and balanced datasets	92%
30	Sivavelu and Palanisamy, 2024	predictions are used to identify defective modules before the testing and to minimize the time and cost	(SQADEN)	Accuracy Precision Recall F1- score time complexity	PROMISE	achieved superior accuracy, precision, and recall compared to existing methods, with significant reductions in prediction time and space complexity	98%

Conclusion

This study highlights the critical role and transformative potential of deep learning techniques in software defect prediction (SDP). As software applications become increasingly important in everyday life, the reliability of these systems has become of paramount importance. A systematic review of the existing literature reveals that traditional defect detection methods are often inadequate due to their reliance on manual testing and a growing consensus on the effectiveness of deep learning methods in identifying defect-prone modules early in the development process.

The results confirm that deep learning models such as convolutional neural networks (CNNs) and long-short-term memory (LSTM) networks, and we demonstrate that these models significantly outperform traditional machine learning methods in terms of accuracy, recall, and overall predictive performance. These advances not only enhance the efficiency of the software development lifecycle but also reduce the costs associated with defect management and maintenance.

In conclusion, the integration of deep learning into software defect prediction represents a major advance in ensuring software reliability and quality and not only improves the identification of potential bugs, but also paves the way for more resilient and efficient software systems in the future. Future research should focus on improving these models and exploring their applicability across diverse programming languages and environments.

Future research should focus on enabling organizations to apply deep learning models to define clear goals such as improving software quality, comprehensively analyzing customer data from identifying customer behavior patterns to improving marketing strategies, collecting balanced data, reducing costs associated with defect fixing, forming teams of developers and project managers, collaborating between these teams to understand business needs to design models that meet those needs, seamlessly integrating models into workflows, training technical teams to ensure optimal use, leading to better outcomes in real-world business applications, and exploring their applicability across diverse programming languages and environments.

Acknowledgement

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.

Conflict of interest

None.

References

[1] Liu, W., Wang, B., & Wang, W. (2021). Deep learning software defect prediction methods for cloud environments research. Scientific Programming, 2021(1), 2323100.

[2] Liang, H., Yu, Y., Jiang, L., & Xie, Z. (2019). Seml: A semantic LSTM model for software defect prediction. IEEE Access, 7, 83812-83824.

[3] Altaie, A. M., Hamo, A. Y., & Alsarraj, R. G. (2021). Software Fault Estimation Tool Based on Object-Oriented Metrics. Iraqi Journal of Science, 63-69.

[4] Qiao, L., Li, X., Umer, Q., & Guo, P. (2020). Deep learning based software defect prediction. Neurocomputing, 385, 100-110.

[5] Giray, G., Bennin, K. E., Köksal, Ö., Babur, Ö., & Tekinerdogan, B. (2023). On the use of deep learning in software defect prediction. Journal of Systems and Software, 195, 111537.

[6] Chakraborty, S., Krishna, R., Ding, Y., & Ray, B. (2021). Deep learning based vulnerability detection: Are we there yet?. IEEE Transactions on Software Engineering, 48(9), 3280-3296.

[7] Hoang, T., Dam, H. K., Kamei, Y., Lo, D., & Ubayashi, N. (2019, May). Deepjit: an end-to-end deep learning framework for just-in-time defect prediction. In 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR) (pp. 34-45). IEEE.

[8] Chen, D., Chen, X., Li, H., Xie, J., & Mu, Y. (2019). DeepCPDP: Deep learning based cross-project defect prediction. IEEE Access, 7, 184832-184848.

[9] Al Qasem, O., & Akour, M. (2019). Software fault prediction using deep learning algorithms. International Journal of Open Source Software and Processes (IJOSSP), 10(4), 1-19.

[10] Edan, T.N.O.(2023) Software defect prediction based on Ensemble learning. Unpublished thesis, College of Computer Science and Mathematics, 99p.

[11] Kumar, L., Dastidar, T. G., Goyal, A., Murthy, L. B., Misra, S., Kocher, V., & Padmanabhuni, S. (2020). Predicting software defect severity level using deep-learning approach with various hidden layers. In Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28 (pp. 744-751). Springer International Publishing.

[12] Özakıncı, R., & Kolukısa Tarhan, A. (2023). A decision analysis approach for selecting software defect prediction method in the early phases. Software Quality Journal, 31(1), 121–177.

[13] Hasanpour, A., Farzi, P., Tehrani, A., & Akbari, R. (2020). Software defect prediction based on deep learning models: Performance study. arXiv preprint arXiv:2004.02589.

[14] Kukkar, A., Mohana, R., Kumar, Y., Nayyar, A., Bilal, M., & Kwak, K. S. (2020). Duplicate bug report detection and classification system based on deep learning technique. IEEE Access, 8, 200749-200763.

[15] Mnyanghwalo, D., Kundaeli, H., Kalinga, E., & Hamisi, N. (2020). Deep learning approaches for fault detection and classifications in the electrical secondary distribution network: Methods comparison and recurrent neural network accuracy comparison. Cogent Engineering, 7(1), 1857500.

[16] Taye, M. M. (2023). Understanding of Machine Learning with Deep Learning Architectures, Workflow, Applications and Future Directions. In Computers (Vol. 12, Issue 5). https://doi.org/10.3390/computers12050091.

[17] Pandey, S. K., Haldar, A., & Tripathi, A. K. (2023). Is deep learning good enough for software defect prediction?. Innovations in Systems and Software Engineering, 1-16.

[18] Lv, J. (2024). Research on Mechanical Fault Diagnosis and Prediction Technology Based on Deep Learning. Transactions on Computer Science and Intelligent Systems Research, 4, 112-117.

[19] Sharma, K. K., Sinha, A., & Sharma, A. (2022). Software Defect Prediction using Deep Learning by Correlation Clustering of Testing Metrics. International journal of electrical and computer engineering systems, 13(10), 953-960.

[20] Alkaberi, W., & Assiri, F. (2024). Predicting the Number of Software Faults using Deep Learning. Engineering, Technology & Applied Science Research, 14(2), 13222-13231.

[21] Elsaraiti, M., & Merabet, A. (2021). Application of long-short-term-memory recurrent neural networks to forecast wind speed. Applied Sciences, 11(5), 2387.

[22] Laura , W., Lars , G., & Timo , K. (2020). "Detecting Software Vulnerabilities with Deep Learning". Berlin: Humboldt University of Berlin.

[23] Batool, I., & Khan, T. A. (2023). Software fault prediction using deep learning techniques. Software Quality Journal, 31(4), 1241-1280.

[24] Borandag, E. (2023). Software fault prediction using an RNN-based deep learning approach and ensemble machine learning techniques. Applied Sciences, 13(3), 1639.

[25] Rizvi, M. (2023). Leveraging Deep Learning Algorithms for Predicting Power Outages and Detecting Faults: A Review. Advances in Research, 24(5), 80-88.

[26] Ramkumar, M., Babu, C. G., Kumar, K. V., Hepsiba, D., Manjunathan, A., & Kumar, R. S. (2021, March). ECG cardiac arrhythmias classification using DWT, ICA and MLP neural networks. In Journal of Physics: Conference Series (Vol. 1831, No. 1, p. 012015). IOP Publishing.

[27] Kumar, L., Dastidar, T. G., Murthy Neti, L. B., Satapathy, S. M., Misra, S., Kocher, V., & Padmanabhuni, S. (2021). Deep-learning approach with Deepxplore for software defect severity level prediction. In Computational Science and Its Applications–ICCSA 2021: 21st International Conference, Cagliari, Italy, September 13–16, 2021, Proceedings, Part VII 21 (pp. 398-410). Springer International Publishing.

[28] Satapathy, S. C., Jena, A. K., Singh, J., Bilgaiyan, S., Ghosh, D., & Singh, J. (2020). A novel approach of software fault prediction using deep learning technique. Automated Software Engineering: A Deep Learning-Based Approach, 73-91.

[29] Cheng, X., Wang, H., Hua, J., Xu, G., & Sui, Y. (2021). Deepwukong: Statically detecting software vulnerabilities using deep graph neural network. ACM Transactions on Software Engineering and Methodology (TOSEM), 30(3), 1-33.

[30] Wang, H., Zhuang, W., & Zhang, X. (2021). Software defect prediction based on gated hierarchical LSTMs. IEEE Transactions on Reliability, 70(2), 711-727.

[31] Sivavelu, S., & Palanisamy, V. (2024). Nonparametric Statistical Feature Scaling Based Quadratic Regressive Convolution Deep Neural Network for Software Fault Prediction. Computers, Materials & Continua, 78(3).

[32] Khleel, N. A. A., & Nehéz, K. (2024). Software defect prediction using a bidirectional LSTM network combined with oversampling techniques. Cluster Computing, 27(3), 3615-3638.

[33] Vujović, Ž. (2021). Classification model evaluation metrics. International Journal of Advanced Computer Science and Applications, 12(6), 599–606.Wahono, R. S. (2015). A systematic literature review of software defect prediction. Journal of Software Engineering, 1(1), 1–16.

[34] Zain, Z. M., Sakri, S., & Ismail , N. H. A. (2023). Application of deep learning in software defect prediction: systematic literature review and meta-analysis. Information and Software Technology, 158, 107175.

[35] Bani-Salameh, H., Sallam, M., & Al shboul, B. (2021). A deep-learning-based bug priority prediction using RNN-LSTM neural networks. e-Informatica Software Engineering Journal, 15(1).

[36] Yu, T. Y., Huang, C. Y., & Fang, N. C. (2021, August). Use of deep learning model with attention mechanism for software fault prediction. In 2021 8th International Conference on Dependable Systems and Their Applications (DSA) (pp. 161-171). IEEE.

[37] Nevendra, M., & Singh, P. (2021). Software defect prediction using deep learning. Acta Polytechnica Hungarica, 18(10), 173-189.

[38] Zhang, X., Li, Y.,& Wang, J. (2023).Interpretable Deep Learning: A Comprehensive Survey. IEEE Transaction on Neural Networks and Learning Systems .Advance online publication .

[39] Aldabbagh, G. M. T., & Hasoon, S. O. (2024). DEFECT SEVERITY CODE PREDICTION BASED ON ENSEMBLE LEARNING. Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, 14(4), 146-153.

[40] Al-Mansoori, A., & Al-Hamadi, A. (2023). Sustainable practices in the manufacturing sector: Challenges and opportunities. Sustainability, 15(1), 234-250. https://doi.org/10.3390/su150100234 .

[41] Fadel, N. A. A., & Bahnam, B. S. (2024). Determining the Quality of Dairy Products Using Machine Learning Techniques. Journal of Education and Science, 33(1), 17-31. doi: 10.33899/edusj.2023.144324.1405.

[42] Faidhi Hamad, K., Celik, B., & Maghded Ahmed, R. (2024).

Classification of Circular Mass of Breast Cancer Using Artificial Neural Network vs. Discriminant Analysis in Medical Image Processing. IRAQI JOURNAL OF STATISTICAL SCIENCES,21(1), 46-58.

doi: 10.33899/iqjoss.2024.0183231.

[43] Hamid,D.N.,&Younis,M.C.(2024).Electronic Health Data Records for Diabetes Patients Based on Deep Learning Models: A Review, 18(2), 52-64.

[44] Al-Zakarya,M.A,& Al-Irhaim,Y.F. (2023). Unsupervised and Semi-Supervised Speech Recognition System: A Review, 17(1), 34-42.