JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

An Optimized Deep Learning Method for Software Defect Prediction Using Whale optimization Algorithm (WOA): A Review

[1]Anes Aliyu Aihong, [1]Badamasi Imam Ya'u, [2]Usman Ali, [1]Abuzairu Ahmad
[1]*Department of Mathematical Sciences,*
Abubakar Tafawa Balewa University, Bauchi.
[2]*Federal College of Education (Technical), Gombe.*

ABSTRACT
Software defect pose significant challenges in the development and maintenance of software systems, leading to increased costs and potential quality issues. In this research, we propose an innovative approach to enhance software defect prediction using an optimized deep learning method. We leverage the whale optimization algorithm (WOA) to fine-tune the parameters of our deep learning model, thereby improving its predictive accuracy. By incorporating the WOA, we achieve a more efficient and effective defect prediction model compared to traditional methods. Our experimental results demonstrate the superiority of our approach in terms of prediction accuracy and reliability, showcasing its potential for practical software quality assurance. This study contributes to the field of software engineering by providing a novel framework that leverages the synergy between deep learning and optimization techniques for software defect prediction. The developed optimized LSTM model will be implemented using Python 3.10. To estimate the prediction capabilities of our LSTM model, we used 19 open-source software defect datasets. These defect datasets are collected from the tera-PROMISE data repository. The computer to be used is a HP laptop running on Windows 10 Operating System with 8GB RAM and Pentium ® Core i7 processor.

ARTICLE INFO
Article History
Received: June, 2023
Received in revised form: June, 2023
Accepted: July, 2023
Published online: September, 2023

KEYWORDS
Deep learning, SDP, Software defect prediction, WOA, Whale optimization algorithms, LSTM, Long short term memory, Machine learning

INTRODUCTION

Software defect prediction is a process of constructing machine learning classifiers to predict defective code snippets, using historical information in software repositories such as code complexity and change records to design software defect metrics (Omri & Sinz, 2020). The predicted results can assist developers to locate and fix potential defects, thereby improving software stability and reliability. According to whether source data and target data are homogeneous or heterogeneous, software defect prediction can be divided into within-project software defect prediction and cross-project software defect prediction (Fan, Diao, Yu, Yang, & Chen, 2019). In this research, we focus on within-project software defect prediction. The program granularity can be file level, change level, or function level, and we choose the file-level granularity as the representation of programs in this paper (Fan et al, 2019). Additionally, Industries should have free access to the dataset to conduct more research experiments for SDP. Researchers and developers should reduce the demand for labeled datasets of large size; they should apply self-supervised learning to large amounts of unlabeled data. They should add more features and test cases so DL can be easily used without over fitting (Abdu et al, 2022). Traditional defect prediction methods mainly consist of two stages: extracting software metrics from historical repositories and constructing a machine learning model for

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

classification. Previous research focuses on designing discriminative artificial metrics to achieve higher model accuracy. These manual metrics are mainly divided into Halstead features based on the number of operators and operands, dependency-based McCabe features, and CK features based on object-oriented programs (Fan et al, 2019). However, sometimes static code attributes cannot distinguish whether the code has defects because a clean code snippet and a buggy one may have the same values of static code attributes, which makes classifiers hard to differentiate. Since the syntactic and semantic information between them is different, features which contain such structural and semantic information should improve the performance of defect prediction. Programs have their own particular syntactic structures and rich semantic information hidden in ASTs, which help analyzing and locating defects more accurately. AST is the representation of source code. It uses a tree structure to describe the relationship of code context, which contains syntactic structures and semantic information of the program module. Leveraging deep learning methods to mine hidden features of ASTs can generate significant features that better reflect the code context information, leading to more accurate software defect prediction (Fan et al, 2019). Software defect prediction is one of the most active research areas in software engineering and plays an important role in software quality assurance. The growing complexity and dependency of the software have increased the difficulty in delivering a high quality, low cost, and maintainable software, as well as the chance of creating software defects. Software defect usually produces incorrect or unexpected results and behaviours in unintended ways (Li et al. 2018). From the perspective of different metrics, it can be divided into two defect prediction categories: static and dynamic. Static defect prediction techniques mainly involve predicting the number of defects or the defect distribution using static software metrics. Dynamic defect prediction techniques mainly involve predicting the distribution of system defects over time using the defect generation time. Most defect prediction techniques build defect prediction models that use traditional hand-crafted features, including Halstead's software volume metrics and McCabe's cyclomatic complexity metrics. Such approaches are based on statistical and machine learning theories. These approaches, which include support vector regression (SVR) (Qiao, Li, Umer, & Guo, 2020), fuzzy support vector regression (FSVR)(Qiao et al., 2020) random forest (RF)(Sun et al., 2018), and Adaboost (Ren, Qin, Ma, & Luo, 2014) models, first build a regression or classification model and then use the model to predict the number or probability of defects for a given unit of source code. These approaches have significantly facilitated the prediction of software defects; however, their performances. (e.g., mean square error and squared correlation coefficient) requires significant improvements (Qiao et al., 2020).

The recently developed deep learning model, called deep neural network (NN) model, has suitable prediction performance. This deep learning model not only deepens the layer levels but can also adapt to capture the training characteristics. A comprehensive, in-depth study and feature excavation ultimately shows the model can have suitable prediction performance (J. Wang & Zhang, 2018). Developing successful software with no defects is one of the main goals of software projects. In order to provide a software project with the anticipated software quality, the prediction of software defects plays a vital role. Machine learning, and particularly deep learning, has been advocated for predicting software defects; however, both suffer from inadequate accuracy, overfitting, and complicated structure (Zain et al, 2022).Thus, this research work proposed to optimized deep learning approach specifically the novel deep recurrent neural network (LSTM) using whale optimization algorithm, in order to enhance the prediction performance of software fault using deep learning algorithms using nineteen open-source software defect datasets, with respect to different evaluation metric.

Statement of the problem

An increasing number of defects in software, damages the quality and reliability of

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

that software. The detection of defective instances is becoming increasingly important, and current detection techniques require a great deal of improvement. However, Machine Learning (ML) techniques are effectively used, to detect defects in software (Chen, Fang, Shang, & Tang, 2018). The primary purpose of ML techniques in Software Defect Prediction (SDP) is to predict defects, according to historical data. Establishing a critical SDP model on high-dimensional and limited data is still a challenging task. Thus, in the existing study in (Nevendra & Singh, 2021) proposed an approach to detect defective modules in software using enhanced Convolutional Neural Networks (CNNs).

The paper aims to identify the defective instance using the enhanced deep learning method. However, the network training was slow resulting to the need of training time reduction and acceleration. Additionally, as suggested by the author (Nevendra & Singh, 2021), the exploration of other robust deep learning algorithm can be explored to enhance the prediction accuracy of software defect. Furthermore, Convolutional neural networks (CNNs) are designed to process data with a grid-like topology, such as images, and are not well-suited for handling time series data, which has a sequential structure (Agga, Abbou, Labbadi, El Houm, & Ali, 2022).

CNNs can potentially be applied to time series data by framing the data as images, but this approach is not widely used and is generally not as effective as using a model specifically designed for time series data, such as a recurrent neural network (RNN) or long short-term memory (LSTM) network (Gholamiangonabadi & Grolinger, 2023). RNNs and LSTMs are well-suited for handling time series data because they are designed to process sequential data. RNNs and LSTMs both have the ability to "remember" past events in the sequence, which allows them to take into account the context and dependencies between events in the data (Abdulrahman et al., 2021). Hence, this research explores the potential of whale optimization base LSTM algorithm for the development of more efficient prediction framework. Base on the authors knowledge, this is the first time an LSTM model is optimized with WOA to enhance the efficacy of software prediction method.

Aim and Objectives of the Study

The aim of this work is to develop a more efficient and optimized Deep Learning models for software defect prediction. This research has the following objectives to;

i. Develop an efficient LSTM base WOA model for software defect prediction
ii. Improve the general prediction performance of the proposed model in terms of detection rate
iii. Evaluate the performance of the propose model using metrics such as accuracy, precision, recall and F-1

LITERATURE REVIEW

The following sub sections provide an overview of some basic ideas that are frequently used in relation to this planned research project.

Software Metrics

Software metrics are measures of the success of a software process. In theory, metrics can help to improve the development process and provide companies with information that makes future projects more predictable, efficient. A number of software defect prediction techniques that use software metrics have been proposed. This can be divided into two types: supervised defect prediction approaches and unsupervised defect prediction approaches. Supervised defect prediction approaches use historical datasets to train a defect prediction model for example, (Benaddy, El Habil, El Meslouhi, & Krit, 2018; Sun et al., 2018; J. Wang & Zhang, 2018) used historical data from Samsung and NASA software fault metric datasets to predict the reliability of the respective project. Unsupervised defect prediction approaches can predict defect proneness without requiring a defect dataset (Yang et al., 2016). This approach can be used when a training dataset is insufficient or is not available. For example (Yang et al., 2016) proposed an unsupervised approach that ranks the change metrics in descending order based on the reciprocal of the raw value of each change metric. The review of some popular fundamental concept pertaining to this proposed

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

researched work is presented in the following sub sections.

Fuzzy Logic

Fuzzy logic has been successfully used to solve variety of problems in system identification, signal processing and control. A fuzzy model structure can be represented by a set of fuzzy If-Then rules. A fuzzy rule has two parts the antecedent and the consequence. The antecedent variables reflect information about the process operating conditions. The consequent of the rule is usually a linear regression model which is valid around the given operating condition. For example (Aljahdali & Sheta, 2011) explore the use of fuzzy logic to build a SRGM. The proposed fuzzy model consists of a collection of linear sub-models joined together smoothly using fuzzy membership functions to represent the fuzzy model. Results and analysis-based data set developed by John Musa of Bell Telephone Laboratories are provided to show the potential advantages of using fuzzy logic in solving this problem Le et al. (1998) firstly proposed Convolution neural networks (CNNs) for image processing, where spatially spatial pooling and shared weights are the main features of the framework.

CNNs where widely applied in various computer application successfully including sequential data, processing of natural language, speech recognition and image classification (Abdel-Hamid, Mohamed, Jiang & Penn, 2012). Convolutional Neural Networks mainly focus on learning features that are abstract; this is achieved by stacking and alternating pooling layers and convolution layers respectively. This convolutional kernels which are the convolution layers in CNN convolve raw input data with multiple local filters thus producing translation invariant local features together with the subsequent pooling layers' extract features and a fixed-length over sliding windows of the raw input data in steps of several rules including average, max and other parameters according to (Zhao, Yan, Wang, & Mao, 2017). Convolutional neural network (CNN) differs from SAE and DBM in fewer parameters and no pre-training process. It replaces matrix calculation with convolution operation.

Classification and Prediction Models for Software Defect Prediction

Regression analysis is a form of predictive modelling technique which investigates the relationship between a dependent (target) and independent variable (s) (predictor). This technique is used for forecasting, time series modelling and finding the causal effect relationship between the variables. Whereas classification models have been widely studied in defect prediction, research on defect prediction using regression models is more limited. A regression model considers defect prediction as a ranking task. The software modules are ranked according to the number of predicted defects they contain. Predicting the number of defects explicitly in software modules is not easy.

Therefore, an accurate regression model is necessary (Qiao et al., 2020). A number of regression model have been proposed in the literature, first study conducted using a linear regression model to predict software failures on a change-level dataset was presented in (Mockus & Weiss, 2000). A recurrent neural network can be extended to a deep recurrent neural network via different ways as shown by (Pascanu, Gulcehre, Cho, & Bengio, 2013), the concept of depth in RNN was initially argued, but base on the architecture of RNN, three areas can be make deeper in other to construct a DRNN, this includes; hidden to hidden function, hidden to hidden transition and hidden to output function (Pascanu et al 2014).. The idea of RNNS with gated units was first introduced by (Hochreiter & Schmidhuber, 1997). LSTM is suitable for modeling long range dependencies; LSTM's architecture has memory blocks Compared to RNN which has hidden units (Chniti, Bakir, & Zaher, 2017). They memory cells which are modulated by nonlinear sigmoidal gates are contained in a memory block, which are multiplicatively applied. Memory of the same gates share cells so as to reduce the parameters, the model stores the values at the gates (when the gates evaluate to 1) or remove them (when the

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

gates evaluate to 0) as determine by the gates. With this, long-range temporal context can exploit by the network (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2015). In order to solve the problem of vanishing gradient during forward and backward phases, the vanishing gradient are alleviated to allow the network to preserve it memory over some time steps. If CNNs are seen as specially designed for grid data like images, recurrent neural networks (RNNs) are constructed for sequential data. However, when the traditional RNNs try to learn wind speed series with long-term dependence, the gradient tends to disappear or sometimes explode after propagation. Therefore, some scholars used gated RNNs to solve this problem, which include long short-term memory (LSTM) and gated recurrent unit (GRU). The LSTM uses memory cells to memorize the temporary state. The input gate, forget gate and output gate are used to change the information of cell state (Liu et. al., 2019).

Auto encoder (AE)

An auto encoder (AE) is a specific kind of unsupervised artificial neural network that provides compression and other functionality in the field of machine learning. The specific use of the auto encoder is to use a feedforward approach to reconstitute an output from an input. The input is compressed and then sent to be decompressed as output, which is often similar to the original input. That is the nature of an auto encoder – that the similar inputs and outputs get measured and compared for execution results (Duan et al, 2019). Auto encoders are an unsupervised learning technique in which we leverage neural networks for the task of representation learning. Specifically, we'll design neural network architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input. If the input features were each independent of one another, this compression and subsequent reconstruction would be a very difficult task. However, if some sort of structure exists in the data (ie. correlations between input features) this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck (Jeremy Jordan 2018)

In general, combination of different approaches such as mixing physical and statistical approaches or combining short term and medium-term models, etc., is referred to as a hybrid approach. For example, radioactive transfer and NN techniques are combined with Special Sensor Microwave/Imager (SSM/I) to get ocean surface wind speeds and direction. Results show that combination of NN would substantially enhance the impact of these data in NWPs than just using SSM/I (Soman, Zareipour, Malik, & Mandal, 2010). Ensemble strategy, due to the diversity, the models can obtain numerous individual forecasting results. It is crucial to adopt appropriate strategies to combine the individual forecasting results. The utilized ensemble strategies in wind energy forecasting models can be classified into two types: weighted-based and learner-based. In the field of ensemble learning, simple averaging method is sometimes adopted to integrate the individual forecasting results, which assigns equal combination weights to individual predictors. Weight-based ensemble model derived from the simple averaging method. The traditional weight-based ensemble model is a linear combination that adds up the forecasting results of N individual predictors (Liu et al., 2019).

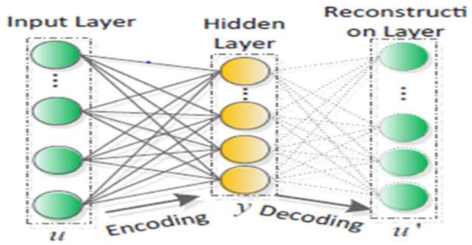


Figure 1: The basic unit of an auto-encoder (Wang et al, 2019)

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

Recurrent Neural Networks (RNNs)

RNNs are well-suited for sequential data, such as speech, text, or time series data. They have the ability to capture temporal dependencies by maintaining internal memory. RNNs use recurrent connections to pass information from previous steps to the current step, allowing them to process sequences of variable lengths. Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) are popular types of RNNs that address the vanishing gradient problem and enhance the memory capabilities of traditional RNNs. LSTM stands for long short-term memory networks, is used in the area of deep learning. It is a modification of recurrent neural networks (RNNs) that has a capability of learning long-term dependencies in data, especially in sequence prediction problems. LSTM has a feedback connection; it means that, LSTM has capability of processing the entire sequence of data, apart from single data points such as images. This finds application in speech recognition, machine translation, etc. LSTM is a special kind of recurrent neural network, which shows outstanding performance on a huge variety of problems (Weerakody et al, 2021).

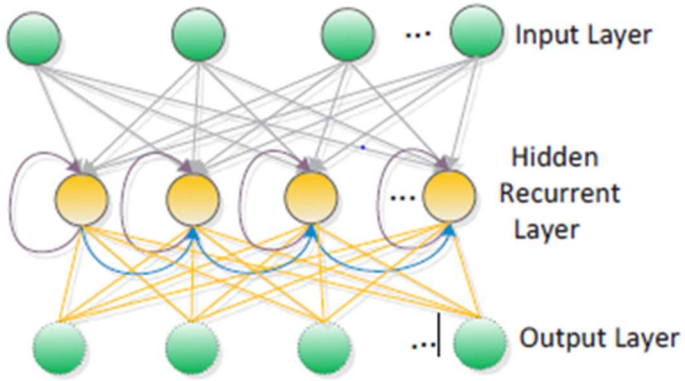


Figure 2: A Typical Structure of a Recurrent Neural Network

Application of Deep Learning Concepts

Deep learning is a subfield of machine learning that focuses on artificial neural networks and their ability to learn and make predictions or decisions. It is an important area of study and research due to its remarkable performance in various domains, including computer vision, natural language processing, speech recognition, and many others (Buenrostro-Mariscal, Santana-Mancilla, Montesinos-López, Nieto Hipólito, & Anido-Rifón, 2022). Here are some key reasons why deep learning is important: Representation learning: Deep learning models have the ability to automatically learn and extract hierarchical representations of data. Through multiple layers of interconnected nodes, deep neural networks can capture complex patterns and features in the input data, allowing them to understand and interpret high-dimensional data in a more effective manner (Bengio, Courville, & Vincent, 2013).

Handling big data: Deep learning algorithms excel at handling large volumes of data. As the availability of data continues to grow rapidly, deep learning techniques provide powerful tools to extract valuable insights from massive datasets. By leveraging their ability to learn from vast amounts of information, deep learning models can discover intricate patterns and make accurate predictions (Anagnostopoulos, Zeadally, & Exposito, 2016). Feature extraction and engineering: Deep learning models can automatically learn relevant features from raw data, eliminating the need for manual feature engineering. Traditionally, in machine learning, engineers would spend significant effort handcrafting features that are relevant to the problem at hand. Deep learning algorithms bypass

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

this step by automatically learning and extracting features directly from the data, which can save time and improve performance (Çayir, Yenidoğan, & Dağ, 2018).

Image and speech recognition: Deep learning has revolutionized computer vision and speech recognition tasks. Convolutional Neural Networks (CNNs) have shown remarkable performance in image classification, object detection, and segmentation. Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) networks, have achieved significant progress in speech recognition, natural language processing, and machine translation (Bantupalli & Xie, 2018). Real-world applications: Deep learning has enabled significant advancements in various real-world applications. It has been used in autonomous vehicles for tasks like object detection, pedestrian detection, and lane detection. In healthcare, deep learning has shown promise in diagnosing diseases from medical images, predicting patient outcomes, and designing personalized treatment plans. It has also found applications in finance, cybersecurity, recommender systems, and many other domains (Sarker, 2021).

Continuous advancements: Deep learning continues to evolve and improve. Researchers and practitioners are constantly developing new architectures, optimization algorithms, and techniques to enhance deep learning models' performance and efficiency. This active research community contributes to ongoing breakthroughs and advancements in the field. Democratization of AI: Deep learning frameworks and libraries, such as TensorFlow, PyTorch, and Keras, have made it easier for researchers and developers to implement and experiment with deep learning models. These tools have contributed to the democratization of AI, allowing more people to leverage deep learning for their own applications and contribute to the field (Sit & Demir, 2023). Overall, deep learning's ability to learn from data, automatically extract features, and make accurate predictions has made it a critical component of modern AI systems. Its impact spans a wide range of industries and applications, driving advancements and unlocking new possibilities in various domains.

Review of Related Studies

Software defect prediction is one of the most active research areas in software engineering and plays an important role in software quality assurance. The growing complexity and dependency of the software have increased the difficulty in delivering a high quality, low cost and maintainable software, as well as the chance of creating software defects. Software defect usually produces incorrect or unexpected results and behaviors in unintended ways (Li et al, 2018). We concentrated on analysis on models that rank a randomly chosen defective class higher than a casually selected clean class with over 80% accuracy (Esteves et al, 2020). With the increasing scale and complexity of software systems, the defects of software are increasing every day. Software defect data is the foundation of research and application of software reliability. Currently, the lack of software defect data, its insufficient coverage, and the limits of the software types involved have become the bottleneck of software reliability research and application (Xu et al, 2019) Due to their roles in assisting developers and testers to automatically allocate the suspicious defects and prioritizing testing efforts, software defect prediction (SDP) techniques are gaining much popularity. SDP could also identify whether a software project is faulty or healthy in the early project life cycle and effectively improve the testing process and software quality.

Software is a set of instructions, programs, or data used to execute specific tasks in various applications such as cyber security, robotics, autonomous vehicles, image processing, prediction of malignant mesothelioma, recommendation systems, etc. (Abdu et al, 2022). Previous studies have used learning techniques based on traditional metrics to build such SDP models. The specific and limited source code features have been manually generated to identify the faulty files among others such as McCabe features, Halstead features, Chidamber and Kemerer (CK) features, MOOD features, change features, and other object-oriented (OO) features.

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

Several machine learning techniques for SDP have been implemented, including Support Vector Machine (SVM), Naive Bayes (NB), Neural Network (NN), Decision Tree (DT), Dictionary Learning, and Kernel Twin Support Vector Machines (KTSVM) (Abdu et al, 2022). Developing successful software with no defects is one of the main goals of software projects. In order to provide a software project with the anticipated software quality, the prediction of software defects plays a vital role. Machine learning, and particularly deep learning, has been advocated for predicting software defects; however, both suffer from inadequate accuracy, overfitting, and complicated structure (Zain et al, 2022).

Table1. Summary of The related work by proposed technique, compared method for evaluation, findings and weakness/limitation

Authors	Model	Contributions	Results	Future work
(Li et al, 2018)	ML combinations	This paper attempts to systematically summarized al typical works on software defect prediction in recent years.	This paper will help researchers to better understand the previous defect prediction studies from datasets.	There need to make more partnership with business partners and have access to their data repositories.
(Nevendra et al., 2021)	CNN	We performed 10-fold across-validation and calculated the average of all runs, resulting in the final result.	The proposed approach and other presented ML shows that the proposed approach achieves the highest ranking.	Exploration of other software metrics, aimed at the development of more efficient DL models, will be considered
(Wahono et al, 2019)	ML model	The experimental results show that the LR performs best in most NASA MDP datasets.	Our results show that the logistic regression perform well.	In this research, a comparison framework is proposed, which aims to benchmark the performance of a wide range of classification models within the field.
(Farid etal, 2021)	CNN and Bi-LSTM	The proposed model CBiL achieves good results for both WPDP and CPDP.	RNN is the best performance of all baseline models.	In future, it is valuable to add quality metrics like defect density, types of defect, and defect severity.
(Liu et al., 2022)	CNN Model	The performance evaluation indicators were tested.	The experimental outcomes demonstrate the dominance of the present method.	We will also focus on software engineering duties such as code completion and code clone detection.
(Zhu et al, 2021)	WOA	We leverage the	The experimental	We also leverage a

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

		recently proposed WOA and another complementary simulated anneaing (SA) to construct and enhanced metaheuristic feature selection algorithm.	results demonstrate that the EMWS algorithm and the WSHCKE predictor can achieve superior prediction performance on four evaluation indications.	hybrid deep neural network.
(Dos et al., 2020)	ML	We concentrated on analysis on models that rank a randomly chosen defective class higher than a casually selected clean class with over 80% accuracy.	Our result indicates that change metric features are more present than entropy or class-level metrics.	This work has focused only to concentrate on analysis models that rank a randomly chosen defective class.
(Giray et al, 2023)	DL algorithm	We increased trend of SDP research over the recent years, with a big variety of fundamental techniques.	Our results can be beneficial for both new canners to SDP researchers to see that landscape of different approaches.	Using different approaches and established researchers to focus their efforts in the coming years.
(Zhou et al, 2019)	DNN	We conduct large-scale empirical experiments on 27 project versions.	The experimental results demonstrate the superiority of our LDFR framework in detecting defective model.	It is recommended for future research to train DNN to learn top-level feature representation.
(Majd, 2020)	LSTM	A formulation of 32 novel source code matric at statement-level.	The experimental results suggested that SLDeep is effective for statement-level SDP.	This work is limited only on to apply LSTM model.
(Zhao et al, 2022)	SDNN	We compared the performance of our SDNN method with the state-of-the-art SDP methods on 10 releases of software defect datasets.	Experimental results show that our SDNN is a competitive approach and is able to improve the prediction performance.	Is limited to compare the SDNN approach with the state-of-the-art SDP approaches utilizing o 10 software defect datasets.

Corresponding author: Anes Aliyu Aihong
✉ anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

(Abdu et al., 2022)	DL Model	A total 283 studies were gathered from electronic resources, 90 articles were considered after applying the research selection criteria.	A comprehensive survey is conducted to systematically show recent software defect prediction techniques based on the source code's key features.	It is essential to compare the performance DL techniques to other DL/statistical approaches to assess their potential comparison to SDP.
(Jindal et al., 2019)	CNN	The research provides a critical analysis of the latest literature, published from 2015 to 2018 on the use of CNN software defect prediction.	Many machine learning-based approaches for software defect prediction have been proposed to achieve the higher accuracy.	This study will focus on ANNs and will also provide a baseline for future innovations, comparisons, and reviews
(Qiao et al., 2020)	Deep learning model	The proposed method significantly reduces the mean square error by more than 14% and increases the square correlation coefficient by more than 80%	The evaluation results illustrate that the proposed approach is a accurate and can improve upon the state-of-the-art approaches.	It is recommended to train other machine learning approaches in the future using the same algorithm
(Balasubr et al, 2022)	CNN+SALO	The measures like sensitivity, accuracy, specify, FNR,FDR,F1-score,mce,,FPR, precision, were computed and verified the node	The adopted SDP with CNN+SALO approach was executed in matlab and its result was verified.	We should strive to keep the number of defects to a bare minimum
(Fan et al, 2019)	DP-ARNN	We apply dictionary mapping and high-dimensional word embedding to convert programs.	The result shows that, average, DP-ARNN improves the F1-measure by 14% and AUC by 7%.	We will try to embed static code attributes into DP-ARNN, and then test whether the performance of defect prediction can be improved.

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

Research Gap

The existing literature focuses on time reduction and accelerated network training in deep learning (DL) models. However, there is a lack of exploration and consideration of other software metrics that could contribute to the development of more efficient DL models. This research gap suggests that there is a need to investigate and identify additional software metrics that can be incorporated into DL model development, apart from time reduction and accelerated network training. By exploring these metrics, researchers can potentially discover new approaches or techniques that optimize DL models for improved efficiency and performance.

METHODOLOGY

The Proposed System

The main goal of this research is to train long short term memory LSTM using whale optimization algorithm WOA in a software defect prediction, in order to have a better result as compared with the existing system.

Analysis of the Existing System

CNN model is trained using the generated metrics; it can predict the defective instances once the model is trained. The proposed enhanced CNN model significantly improves compared to existing benchmark classification schemes such as KNN, SVM, AdaBoost and Li's CNN model. The existing system performed 10-fold across-validation and calculated the average of all runs, resulting in the final result. In order to train the CNN model, the FS technique is applied to the selected K-1 parts of data because the CNN model's inputs size should be expressed in metric units of n∗n. the n∗n metrics cannot accommodate m number of features in our dataset. So, either increase the number of features in the dataset or remove those that aren't needed. This is why we removed unnecessary features to convert our dataset into n∗n format in order to avoid performance degradation. We used Chi square (Chi2) filter-based FS to select N number of features in our experiment. It has been demonstrated by Nam et al, that Chi-square is the most effective FS of all approaches. When features and classes are linked together, Chi2 performs well. When selecting relevant features, it helps to identify the frequency between class and feature. In our execution, we selected only those features that had the highest Chi2 scores. Calculation of the Chi2 score is denoted by:

$$Chi - square\ (x^2) = \frac{(observed\ frequency - expected\ frequency)^2}{expected\ frequency} \quad \ldots (1)$$

The number of examined classes is defined as the observed frequency. However, if there were no association between feature and target, the predicted class number would represent the expected frequency of that future. Chi FS ranked the features based on their relationship to the class as long as they are ranked in order. To create n*n features metrics; we removed the lower-ranked feature metric from the features metrics. n∗n 2D metrics are created after N features are chosen from the source data.

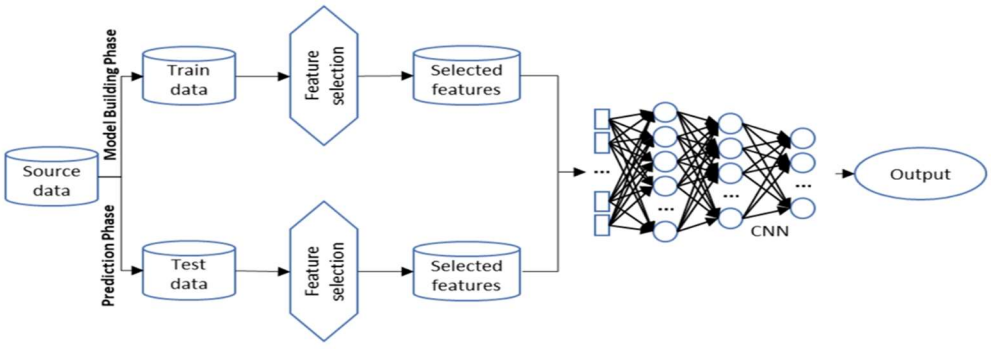


Figure 3: workflow of the existing model (Nevendra et al, 2021)

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

Weakness of the Existing System

It has been proved that the existing system enhanced CNN model significantly improves compared to existing benchmark classification schemes such as KNN, SVM, Adaboost and Li's CNN model. Scott Knott ESD's mean ranking comparison of the existing approach and other presented ML approaches shows that the existing approach achieves the highest ranking. However, the major weakness of the existing system is that the CNN trained with a very huge datasets.

Long Short-Term Memory

A Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition and anomaly detection in network traffic or IDSs (intrusion detection systems). A common LSTM unit is composed of a cell, and input gate, and output gate and a forget gate.

The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and making predictions based onetime series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

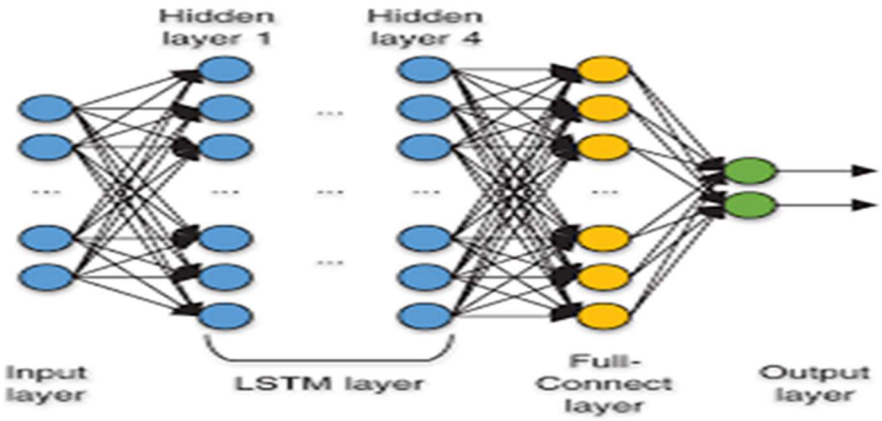


Figure 4: Structure of Long Short-Term Memory (LSTM) neural network (Abuzairu et al, 2022)

The figure above shows a typical LSTM with an input layer, four hidden layers, and an output layer. The input layer contains input neurons that send information to the hidden layers. Every neuron has weighted inputs (synapses), an activation function (that defines the output of a given input), and one output. Synapses are the adjustable parameters that convert a neural network to a parameterized system. The weighted sum of the inputs produces the activation signal that is passed to the activation function to obtain one output from the neuron. The commonly used activation functions are linear, step, sigmoid, tanh, and ReLU functions.

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

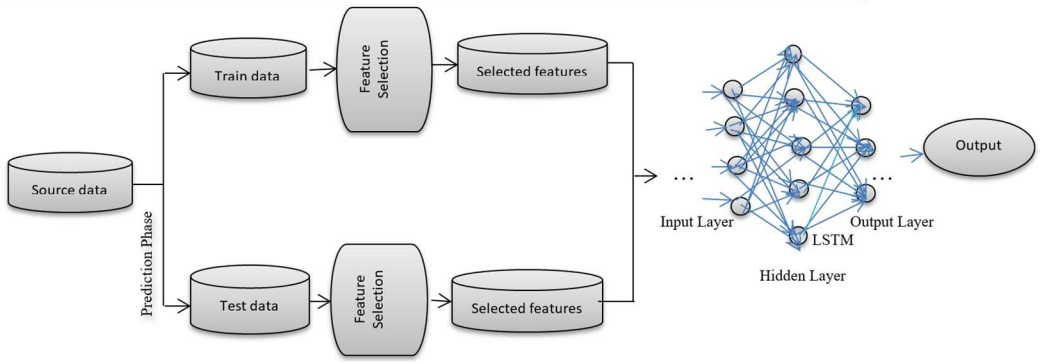


Figure 5: workflow of the proposed model

The Whale Optimization Algorithm

Whale optimization algorithm (WOA) is a recently proposed stochastic optimization algorithm (Aljerah et al, 2018). It utilizes a population of search agents to determine the global optimum for optimization problems. Similarly, to other population-based algorithms, the search process starts with creating a set of random solutions (candidate solutions) for a given problem. It then improves this set until the satisfaction of an end criterion. The main difference between WOA and other algorithms is the rules that improve the candidate solutions in each step of optimization. In fact, WOA mimics the hunting behavior of humpback whales in finding and attacking preys called bubble-net feeding behavior. The bubble-net hunting model is shown in the figure below:

Figure 6: Bubble net hunting behavior (Aljerah et al, 2018)

It may be observed in this figure that a humpback whale creates a trap with moving in a spiral path around preys and creating bubbles along the way. This intelligent foraging method is the main inspiration of the WOA. Another simulated behavior of humpback whales in WOA is the encircling mechanism. Humpback whales circle around preys to start hunting them using the bubble-net mechanism. It is worth mentioning here that bubble-net feeding is a unique behavior that can only be observed in humpback whales. It was proven by the inventors of WOA that this

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 11(3), SEPTEMBER, 2023
ISSN: 2277-0011; Journal homepage: www.atbuftejoste.net

algorithm is able to solve optimization problems of different kinds. It was argued in the main work that this is due to the flexibility, gradient-free mechanism, and high local optima avoidance of this algorithm. These motivated our attempts to employ WOA as a trainer for FFNNs due to the difficulties of learning process. Theoretically speaking, WOA should be able to train any ANN subject to proper objective function and problem formulation. In addition, providing the WOA with enough number of search agents and iterations is another factor for the success of this algorithm. To measure the fitness value of the generated WOA

Evaluation Parameters and Performance Metrics

After implementation, the proposed system will be evaluated based on its performance. Accuracy, convergence speed, and precision score are the performance metrics of this work. These parameters are mathematically computed as follow;

Accuracy = $\frac{TP+TN}{TP+TN+FP+F}$ ... (2)

The accuracy is the percentage of correct classification in the total number of classifications, Where: T and F stand for true and false, P and N stand for positive and negative samples respectively.

Precision = $\frac{TP}{TP+FP}$ ... (3)

The precision is calculated by dividing the number of correct classifications by the number of incorrect. $Recall = \frac{TP}{TP+F}$ ....(4)

The recall measures the number of correct classifications minus the number of missed entries.

$$F1 - score = 2 * \frac{precision * Recall}{Precision + Recall} \quad \ldots (5)$$

On the other hand, an F1- score is a derived effectiveness measurement that measures the harmonic mean of precision and recall.

Implementation of the System

The developed optimized LSTM model will be implemented using Python 3.10. To estimate the prediction capabilities of our LSTM model, we used 19 open-source software defect datasets. These defect datasets are collected from the tera-PROMISE data repository. The computer to be used is a HP laptop running on Windows 10 Operating System with 8GB RAM and Pentium ® Core i7 processor.

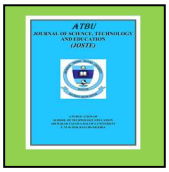
Data Collection

This research work uses a journal published by Nevendra et al, (2021) as a primary source of data collected. Many other journals and websites were also consulted during the data collection. Thus, the research makes use of 19 open-source software defect datasets. These defect datasets are collected from the tera-PROMISE data repository.

CONCLUSIONS AND FUTURE RESEARCH DIRECTION

In conclusion, our study has successfully demonstrated the effectiveness of integrating the whale optimization algorithm (WOA) into a deep learning framework for software defect prediction. Through rigorous experimentation and analysis, we have shown that this optimized approach outperforms traditional methods in terms of predictive accuracy and reliability. The result affirms the potential of the proposed method for practical software quality assurance, highlighting its ability to contribute significantly to the field of software engineering. By harnessing the power of deep learning and fine-tune capabilities of the WOA, our research offers a promising solution to address the challenges associated with software defects. It opens the door to more efficient defect detection and mitigation, ultimately reducing costs and enhancing the overall quality of software systems. As software complexity continues to grow, the adoption of such advanced techniques becomes increasingly crucial, and our study paves the way for further exploration and application of similar methodologies in the software development industry.

Corresponding author: Anes Aliyu Aihong
anesaliyu123@gmail.com
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.



REFERENCES

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

Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. Soft Computing, 22(1), 1-15.

Ahmad, A., Musa, K. I., Zambuk, F. U., & Lawal, M. A. (2022). Optimizing Connection Weights in a Long Short-Term Memory (LSTM) Using Whale Optimization Algorithm (WOA): A Review. ATBU Journal of Science, Technology and Education, 10(3), 362-373.

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

Manjula, C., & Florence, L. (2018). Software Defect Prediction Using Deep Belief Network With L1-Regularization Based Optimization. International Journal of Advanced Research in Computer Science, 9(1).

Samir, M., El-Ramly, M., & Kamel, A. (2019, November). Investigating the use of deep neural networks for software defect prediction. In 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA) (pp. 1-6). IEEE.

Balasubramaniam, S., & Gollagi, S. G. (2022). Software defect prediction via optimal trained convolutional neural network. Advances in Engineering Software, 169, 103138.

Fan, G., Diao, X., Yu, H., Yang, K., & Chen, L. (2019). Software defect prediction via attention-based recurrent neural network. Scientific Programming, 2019.

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.

Xu, J., Yan, L., Wang, F., & Ai, J. (2020, January). A GitHub-based data collection method for software defect prediction. In 2019 6th International Conference on Dependable Systems and Their Applications (DSA) (pp. 100-108). IEEE.

Zain, Z. M., Sakri, S., Ismail, N. H. A., & Parizi, R. M. (2022). Software defect prediction harnessing on multi 1-dimensional convolutional neural network structure. Computers, Materials and Continua, 71(1), 1521.

Xu, Z., Li, S., Xu, J., Liu, J., Luo, X., Zhang, Y., ... & Tang, Y. (2019). LDFR: Learning deep feature representation for software defect prediction. Journal of Systems and Software, 158, 110402.

Zhao, L., Shang, Z., Zhao, L., Qin, A., & Tang, Y. Y. (2018). Siamese dense neural network for software defect prediction with small data. IEEE Access, 7, 7663-7677.\

Abdu, A., Zhai, Z., Algabri, R., Abdo, H. A., Hamad, K., & Al-antari, M. A. (2022). Deep Learning-Based Software Defect Prediction via Semantic Key Features of Source Code—Systematic Survey. Mathematics, 10(17), 3120.

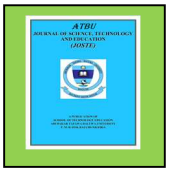
Khan, M. A., Elmitwally, N. S., Abbas, S., Aftab, S., Ahmad, M., Fayaz, M., & Khan, F. (2022). Software defect prediction using artificial neural networks: A systematic literature review. Scientific Programming, 2022.

Li, Z., Jing, X. Y., & Zhu, X. (2018). Progress on approaches to software defect prediction. Iet Software, 12(3), 161-175.

Wahono, R. S., Herman, N. S., & Ahmad, S. (2014). A comparison framework of classification models for software defect prediction. Advanced Science Letters, 20(10-11), 1945-1950.

Farid, A. B., Fathy, E. M., Eldin, A. S., & Abd-Elmegid, L. A. (2021). Software defect prediction using hybrid model (CBIL) of convolutional neural network (CNN)

Corresponding author: Anes Aliyu Aihong
✉ *anesaliyu123@gmail.com*
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.



and bidirectional long short-term memory (Bi-LSTM). PeerJ Computer Science, 7, e739.

dos Santos, G. E., & Figueiredo, E. (2020, September). Failure of One, Fall of Many: An Exploratory Study of Software Features for Defect Prediction. In 2020 IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM) (pp. 98-109). IEEE.

Majd, A., Vahidi-Asl, M., Khalilian, A., Poorsarvi-Tehrani, P., & Haghighi, H. (2020). SLDeep: Statement-level software defect prediction using deep-learning model on static code features. Expert Systems with Applications, 147, 113156.

Jindal, R., Malhotra, R., & Jain, A. (2014, October). Software defect prediction using neural networks. In Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization (pp. 1-6). IEEE.

Corresponding author: Anes Aliyu Aihong
✉ *anesaliyu123@gmail.com*
Department of Mathematical Sciences, Abubakar Tafawa Balewa University, Bauchi.