НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
март–апрель 2024 Том 24 № 2 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
March–April 2024 Vol. 24 No 2 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

doi: 10.17586/2226-1494-2024-24-2-222-229

An optimized deep learning method for software defect prediction using Whale Optimization Algorithm

Anes Aliyu Aihong[1]✉, Badamasi Imam Ya'u[2], Usman Ali[3], Abuzairu Ahmad[4], Mustapha Abdulrahman Lawal[5]

[1,2,4,5] Abubakar Tafawa Balewa University (ATBU), Bauchi, 740272, Nigeria
[3] Federal College of Education (Technical), Gomber, 760101, Nigeria

[1] Anesaliyu123@gmail.com✉, https://orcid.org/0009-0009-5169-7593
[2] biyau@atbu.edu.ng, https://orcid.org/0000-0002-2710-8973
[3] usmanali@fcetgombe.edu.ng, https://orcid.org/0000-0001-9645-3642
[4] Abuzairuahmad2020@gmail.com, https://orcid.org/0000-0003-0229-739X
[5] musbaida@gmail.com, https://orcid.org/0000-0002-1037-2022

Abstract
The goal of this study is to predict a software error using Long Short-Term Memory (LSTM). The suggested system is an LSTM taught using the Whale Optimization Algorithm to save training time while improving deep learning model efficacy and detection rate. MATLAB 2022a was used to develop the enhanced LSTM model. The study relied on 19 open-source software defect databases. These faulty datasets were obtained from the tera-PROMISE data collection. However, in order to evaluate the model performance to other traditional approaches, the scope of this study is limited to five (5) of the most highly ranked benchmark datasets (DO1, DO2, DO3, DO4, and DO5). The experimental results reveal that the quality of the training and testing data has a significant impact on fault prediction accuracy. As a result, when we look at the DO1 to DO5 datasets, we can see that prediction accuracy is significantly dependent on training and testing data. Furthermore, for DO2 datasets, the three deep learning algorithms tested in this study had the highest accuracy. The proposed method, however, outperformed Li's and Nevendra's two classical Convolutional Neural Network algorithms which attained accuracy of 0.922 and 0.942 on the DO2 software defect data, respectively.

Keywords
deep learning, software defect prediction, whale optimization algorithms, long short-term memory, machine learning, optimization algorithm

Acknowledgements
This study is funded by the Abubakar Tafawa Balewa University (ATBU) in Bauchi, Nigeria.

For citation: Aliyu Aihong A., Imam Ya'u B., Ali U., Ahmad A., Abdulrahman Lawal M. An optimized deep learning method for software defect prediction using Whale Optimization Algorithm. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2024, vol. 24, no. 2, pp. 222-229. doi: 10.17586/2226-1494-2024-24-2-222-229

УДК 004.85

Оптимизированный метод глубокого обучения для прогнозирования дефектов программного обеспечения с использованием алгоритма оптимизации кита

Анес Алию Айхонг[1]✉, Бадамаси Имам Яу[2], Усман Али[3], Абузайру Ахмад[4], Мустафа Абдулрахман Лаваль[5]

[1,2,4,5] Университет Абубакара Тафавы Балева (ATBU), Баучи, 740272, Нигерия
[3] Федеральный педагогический колледж (технический), Гомбе, 760101, Нигерия

[1] Anesaliyu123@gmail.com✉, https://orcid.org/0009-0009-5169-7593
[2] biyau@atbu.edu.ng, https://orcid.org/0000-0002-2710-8973
[3] usmanali@fcetgombe.edu.ng, https://orcid.org/0000-0001-9645-3642
[4] Abuzairuahmad2020@gmail.com, https://orcid.org/0000-0003-0229-739X
[5] musbaida@gmail.com, https://orcid.org/0000-0002-1037-2022

Аннотация

Целью исследования является прогнозирование ошибки программного обеспечения с использованием долговременной кратковременной памяти (Long Short-Term Memory, LSTM). Предлагаемая система представляет собой LSTM, обучаемую с использованием алгоритма оптимизации китов (Whale Optimization Algorithm). Система обеспечивает экономию времени обучения. Одновременно повышается эффективность модели глубокого обучения (DL) и скорость обнаружения. Для разработки расширенной модели LSTM применен программный пакет MATLAB 2022a. Использованы 19 баз данных дефектов программного обеспечения с открытым исходным кодом. Ошибочные наборы данных получены из коллекции tera-PROMISE. Для оценки эффективности модели по сравнению с другими традиционными подходами объем исследования ограничен пятью наборами эталонных данных с наиболее высоким рейтингом (DO1, DO2, DO3, DO4 и DO5). Результаты экспериментов показали, что качество данных обучения и тестирования оказывает существенное влияние на точность прогнозирования ошибок. При анализе на наборах данных от DO1 до DO5 видно, что точность прогнозирования существенно зависит от результатов обучения и тестирования. Три алгоритма DL, протестированные на наборе данных DO2, показали самую высокую точность (0,942) в сравнении с двумя классическими алгоритмами с использованием сверточной нейронной сети Li's и Nevendra's (0,922).

Ключевые слова

глубокое обучение, SDP, прогнозирование дефектов программного обеспечения, WOA, алгоритмы оптимизации китов, LSTM, долговременная память, машинное обучение, алгоритм оптимизации

Благодарности

Исследование финансируется Университетом Абубакара Тафавы Балева (ATBU) в Баучи, Нигерия.

Ссылка для цитирования: Алию Айхонг А., Имам Яу Б., Али У., Ахмад А., Абдулрахман Лаваль М. Оптимизированный метод глубокого обучения для прогнозирования дефектов программного обеспечения с использованием алгоритма оптимизации кита // Научно-технический вестник информационных технологий, механики и оптики. 2024. Т. 24, № 2. С. 222–229 (на англ. яз.). doi: 10.17586/2226-1494-2024-24-2-222-229

Introduction

Long Short-Term Memory (LSTM) are recurrent neural networks which are frequently used to model sequential data such as time series or natural language [1]. The Convolutional Neural Network (CNN) are built neural networks [2], and they are mostly used for image recognition and categorization [1]. Whale Optimization Algorithm (WOA) is a novel optimization technique for tackling optimization issues [3]. Recurrent Neural Network (RNN) is a sort of artificial neural network that processes data sequences [4]. Software Defect Prediction (SDP) goal is to detect problematic modules in order to allocate testing resources more effectively, which is an economically significant activity in software quality assurance [5].

Review of Related Literature

Software faults frequently result in incorrect or unexpected outputs and unpleasant actions [6]. We concentrated on models that outperform a randomly selected defective class of greater than 80 % accuracy [7]. The prediction of software mistakes is crucial to delivering the required software quality on a software project. Despite the fact that Machine Learning (ML), particularly Deep Learning (DL), has been advocated for forecasting software problems, both suffer from insufficient accuracy, over fitting, and complicated structure [8].

Statement of the Problem

The increasing quantity of software faults degrades its quality and reliability [9]. Defect detection is becoming increasingly critical, and present detection approaches might be improved significantly. Creating a critical SDP model on high-dimensional and restricted data remains a difficult task. Thus, the current study in [10] provides a strategy for detecting problematic modules in software using modified CNNs. However, because network training was slow, training time needed to be reduced and accelerated. Also, as indicated by the author [10], more robust DL algorithms can be investigated to improve the prediction accuracy of SDP. As a result, this study investigates the possibilities of the WOA-based LSTM algorithm in the construction of a more efficient prediction framework. According to the authors' knowledge, this is the first time an LSTM model has been adjusted with WOA to improve the SDP method efficacy.

Aim and objectives

The goal of this research is to create a more efficient and optimal DL model for SDP. This research has the following objectives to:

— Develop an efficient LSTM-based WOA model for SDP.
— Accelerate the network training time of the proposed model.
— Enhance the suggested model overall prediction performance in terms of detection rate.
— Evaluate the proposed model accuracy, precision, recall, and F1.

Methodology

The primary purpose of this research is to outperform the current system by employing the WOA to train an LSTM for SDP. As a result, this part analyzes and discusses the existing framework flaws, as well as the suggested system, data collection, implementation, and assessment metrics.

Fig. 1 depicts the overall workflow of the proposed approach. The improved LSTM-based WOA model developed here was used to predict flaws in software projects. This method involves two stages: model creation and prediction. The WOA is used to train the LSTM model in order to reduce the training time of the DL model. Because of its higher accuracy, LSTM is preferred over CNN for processing time series data.

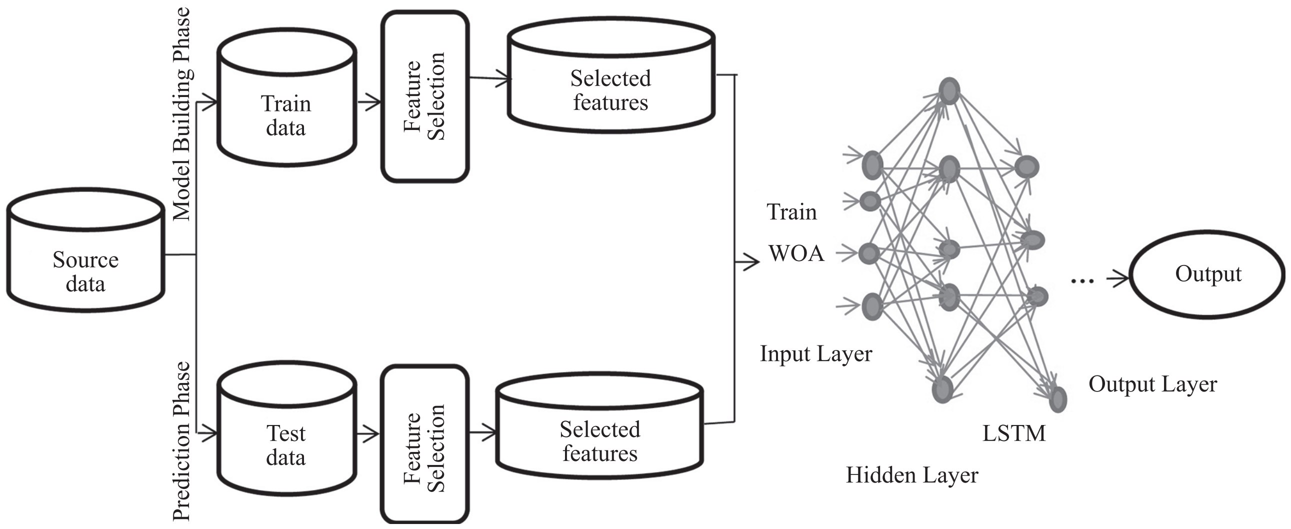

Fig. 1. Workflow of the proposed model

Long Short-Term Memory

LSTM is more accurate than traditional RNNs [11]. It was first proposed in [12]. Memory blocks, as opposed to RNN, are discrete units found in the LSTM recurrent hidden layer [11]. Memory blocks are made up of memory cells with self-connections that record the network temporal state as well as specific multiplicative units called gates that regulate information flow. In the original architecture, each memory block featured three distinct gate types: an input gate, an output gate, and a forget gate [11].

Whale Optimization Algorithm

The WOA makes use of a population of search agents tasked with locating the best global solution to optimization problems. Like alternative population-based algorithms, the search process starts with the generation of a set of randomly generated solutions (candidate solutions) for a given problem [3]. WOA is distinguished from different algorithms by rules that enhance the candidate solutions at every stage of optimization. In reality, WOA imitates the hunting behavior of humpback whales by locating and attacking prey using a technique known as “bubble-net” feeding. LSTM optimization for shorter training durations is a critical problem, particularly when working with large datasets and complex models [3]. As a result, the WOA was used in this study to accelerate the training of LSTM networks, as one of our key goals is to investigate the computing time of the algorithms as a measure for evaluating the model quality.

Dataset Description

This study gathered information from an article published in [10]; many journals and websites were consulted during the data collection procedure. The University of California Irvine (UCI) ML repository is a highly important site for collecting open source and free datasets for ML[1]. The researcher used 19 open-source software defect datasets to estimate the prediction abilities of the proposed LSTM base WOA model. These problematic datasets were obtained from the tera-PROMISE data collection. The scope of this study, however, is limited to five of the best ranking benchmark datasets in order to compare the model performance to that of other traditional approaches. The PROMISE repository was inspired by the UCI ML repository[1]. The datasets were read as XLS file and were divided into 80 % for training and 20 % for testing in order to develop a model. The dataset was pre-processed by carrying out data cleaning/scrubbing to remove typographical errors and inconsistencies in the data. We then saved the dataset in the format required by MATLAB through data formatting phase.

Table 1 shows the statistics of utilized datasets. Column one shows the dataset ID (D.ID), column two shows the dataset name, column three shows the line of code, and column four show the number of instances and defects. The line of code (LOC) it was also introduced.

Table 2 lists the characteristics of the dataset. Columns 1 and 3 include the feature Ids (F.ID), while columns 2 and 4 provide the features name for all datasets.

To obtain the desired result, the input size was set to 5, the number of hidden units to 200, and the number of classes to 3. These parameters are described as follows:

— The series Input Layer function takes an argument called Input Size. It is the feature dimension, or the number of rows in the matrix in each cell.
— Num Hidden Units is an LSTM Layer function parameter that specifies the number of hidden units in the LSTM network.
— The argument of completely Connected Layer is the number of labels which is the number of wolves to be identified in this study.

Table 3 shows the parameters settings for the proposed algorithm. In this work, the LSTM layer is used to analyze the sequence both forward and backward. The XLS file was opened. With a total of 600 iterations, the number of

[1] Sayyad Shirabad J. and Menzies T.J. PROMISE Repository of Software Engineering Databases, School of Information Technology and Engineering, University of Ottawa, Canada. 2005. Available at: http://promise.site.uottawa.ca/SERepository (accessed: 09.02.2024).

Table 1. Statistics of Dataset

D.ID	Datasets	LOC	Instance/Defect
D01	log4j-1.0	21,549	135 / 34
D02	log4j-1.2	38,191	205 / 189
D03	lucene-2.0	50,596	195 / 91
D04	lucene-2.2	63,571	247 / 144
D05	lucene-2.4	102,859	340 / 203
D06	poi-1.5	55,428	237 / 141
D07	poi-2.0	93,171	314 / 37
D08	poi-2.5	119,731	385 / 248
D09	poi-3.0	129,327	442 / 281
D10	synapse-1.0	159,254	440 / 71
D11	synapse-1.1	42,302	222 / 60
D12	synapse-1.2	53,500	256 / 86
D13	velocity-1.4	51,713	196 / 147
D14	velocity-1.6	57,012	229 / 78
D15	xalan-2.4	225,088	723 / 110
D16	xalan-2.5	304,860	803 / 387
D17	xalan-2.6	411,737	885 / 411
D18	xerces-1.2	159,254	440 / 71
D19	xerces-1.3	167,095	453 / 69
—	Total	2,306,238	7,147 / 2,858

Table 2. Features in the datasets

F.ID	Features name	F.ID	Features name
1	wmc	11	moa
2	dit	12	mfa
3	noc	13	cam
4	cbo	14	ic
5	rfc	15	cbm
6	lcom	16	amc
7	lcom3	17	ca
8	npm	18	ce
9	loc	19	max_cc
10	dam	20	avg_cc

search agents was limited to 40. The details of the chosen benchmark function were loaded.

Table 4 defined all the layers of the network. We specify the classifier' training options. Set Max Epochs to 7, and the network will iteratively run over the training data seven times. We chose a batch size of 27 to allow the network to evaluate 13 training signals at the same time. Plots can be set to "training-progress" to show training progress as the number of iterations grows. We set verbose to false to avoid producing the table output that matches the data given in the visual. We use the same mini-batch size as for training to categorize the test data, which is then used to determine the prediction accuracy. The WOA strategy is used in this study to maximize the network optimal weight by searching through the identified agents as indicated during design.

Table 3. Parameters Settings for the proposed algorithm

Parameter	Settings
Sequence Input Layer	4
LSTM Layer	100
Fully Connected Layer	2
SoftMax Layer	1
Classification Layer	1
Max Epochs	7
Mini Batch Size	27
Verbose	False
Learn Rate Schedule	Piecewise
Maximum Iterations	600
input Size	5
Num Hidden Units	200
Num Classes	2

Table 4. Parameter Settings

Parameter	Setting
Layer of Sequence Input	Input size
The LSTM Layer	2
Layer Completely Connected	1
Softmax Layer	1
Classification Layer	1
Max Epochs	7
Mini Batch Size	27
Gradient Threshold	1
Verbose	False
Execution Environment	CPU

Evaluation Parameters and Performance Metrics

Following implementation, the suggested system will be evaluated based on its performance. The performance parameters for this work include accuracy, convergence speed, and precision score. These parameters are computed mathematically as follows:

Accuracy: This performance metric is concerned with the model correct prediction, and it may be stated as:

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

Precision: Precisions tell you how exact or accurate your model is in terms of anticipated positives and how many of them are true positives. When the cost of false positives is substantial, precision is a good metric to use. It can be stated numerically as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall: Recall seeks to determine what percentage of true positives was accurately detected. It is written mathematically as:

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

F1: is a function of accuracy and recall; it may be more appropriate to use when we want to strike a balance between precision and memory and there is an even class distribution (a high number of genuine negatives). It can be stated numerically as:

$$\text{F1} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Where TP (True Positive) shows whether instances where the actual class of the data point was 1 (true) and the projected class was also 1 (true)?

TN (True Negative) shows whether instances where the actual class of the data point was 0 (false) and the projected class was also 0 (false)?

FP (False Positive) shows whether instances where the actual class of the data point was 0 (false) and the expected class was 1 (true)?

FN (False Negative) shows whether instances when the actual class of the data point was 1 (true) and the predicted class was 0 (false)?

Result and Discussion

We use a static analysis-based evaluation sub-module to benchmark the DL architectures. On publicly available datasets containing collected samples, the performance of numerous traditional ML and DL methods for SDP is studied. In this part, the proposed methodologies were tested against the most extensively used software defect classification approach (CNN) on benchmark datasets. Precision, recall, accuracy, and F1 were all recorded for each one. The study was carried out on a system powered by an Intel Core i7 processor. The simulation output is evaluated and compared in three stages, beginning with accuracy and progressing to training performance.

From Table 5, each of the evaluation metrics (accuracy, precision, recall and F1) was reported between 0 and 100. In each case, the higher is the value the better is the model performance. It is quite obvious that the proposed system achieved the best performance in terms of accuracy, precision, recall and F1.

Classification Accuracy

The three DL algorithms used in this study had the highest accuracy. In addition, the suggested system produced a higher classification accuracy of 0.975 on the DO2 software defect data than the other classical CNN systems utilized by [6] and [10], which reached 0.922 and 0.942, respectively. This means that the suggested model outperformed the most recent DL approach in accurately predicting the presence of faults. Fig. 2 compares the suggested model performance to that of the existing CNN approach.

Precision

On the DO2 software defect data, the suggested system achieved the highest precision score of 0.968 when compared to the other classical CNN algorithms utilized by [6] and [10], which achieved accuracy of 0.952 and 0.902, respectively.

From the Fig. 3, we can see that the proposed system achieved the best precision score of 0.968 on the DO2 software defect data compare to the other classical CNN approaches used by [6] and [10] which achieved accuracy of 0.952 and 0.902, respectively.

Recall

The suggested method had the highest precision score of 0.989 on the DO2 software defect data when compared to the other classical CNN systems utilized by [6] and [10] which had accuracy of 0.962 and 0.910, respectively. However, precision and recall are sometimes merged into a single statistic known as the F1 which strikes a compromise between the two.

As seen from the Fig. 1, the proposed system achieved the best precision score of 0.989 on the DO2 software defect data compare to the other classical CNN approaches

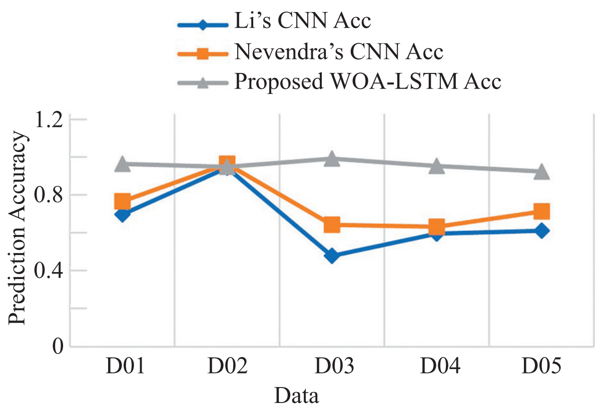

Fig. 2. Classification accuracy for all methods

Table 5. Performance comparison with classical CNN architectures on the same dataset

Data	Li's CNN				Nevendra's CNN				Proposed WOA-LSTM			
	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1	Acc	Pre	Rec	F1
D01	0.682	0.623	0.533	0.574	0.748	0.756	0.778	0.767	**0.942**	**0.951**	**0.936**	**0.923**
D02	**0.922**	**0.902**	**0.910**	**0.906**	**0.942**	**0.952**	**0.962**	**0.957**	**0.975**	**0.968**	**0.989**	**0.978**
D03	0.468	0.528	0.570	0.548	0.628	0.638	0.646	0.642	**0.969**	**0.927**	**0.932**	**0.930**
D04	0.583	0.573	0.568	0.570	0.618	0.658	0.648	0.653	**0.931**	**0.941**	**0.936**	**0.925**
D05	0.597	0.586	0.592	0.589	0.697	0.721	0.732	0.726	**0.903**	**0.974**	**0.932**	**0.926**
Average	0.666	0.664	0.670	0.666	0.775	0.782	0.790	0.786	**0.934**	**0.952**	**0.935**	**0.926**

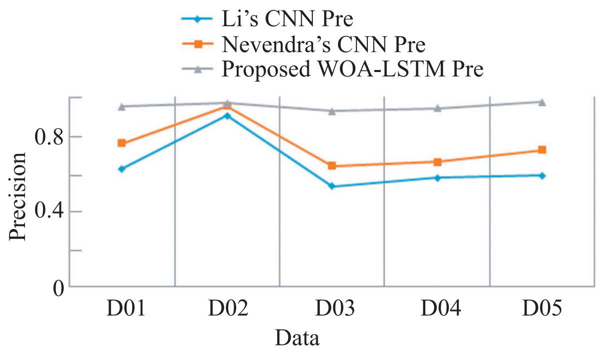

Fig. 3. Precision for all methods

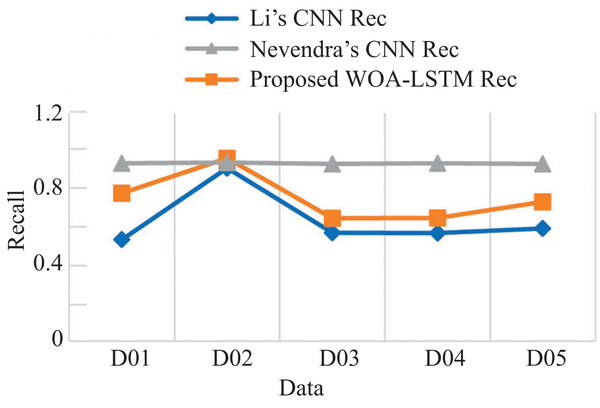

Fig. 4. Recall for all method

used by [6] and [10] which achieved accuracy of 0.962 and 0.910, respectively.

F1

On the DO2 software defect data, the suggested system achieved the highest F1 of 0.978 when compared to the other classical CNN systems utilized by [6] and [10], which achieved 0.957 and 0.906, respectively.

From Fig. 5, the proposed system achieved the best F1 of 0.978 on the DO2 software defect data compare to the other classical CNN approaches used by [6] and [10] which achieved accuracy of 0.957 and 0.906, respectively.

In general, it can be clearly noticed that our proposed method (WOA-LSTM in Fig. 6) achieved the highest accuracy with 0.934, precision 0.952, recall 0.935, and F1 with 0.926, irrespective of the software defect datasets used. Compared to the existing CNN in [6], which achieved

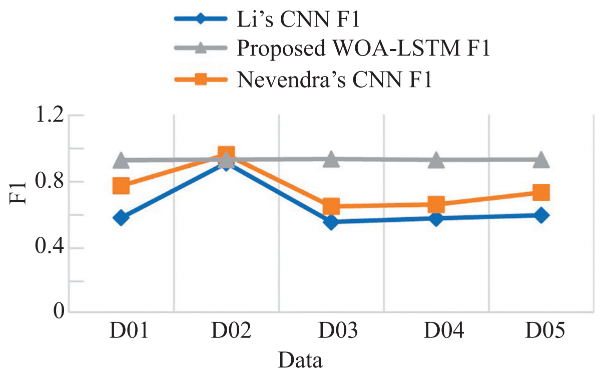

Fig. 5. F1 for all methods

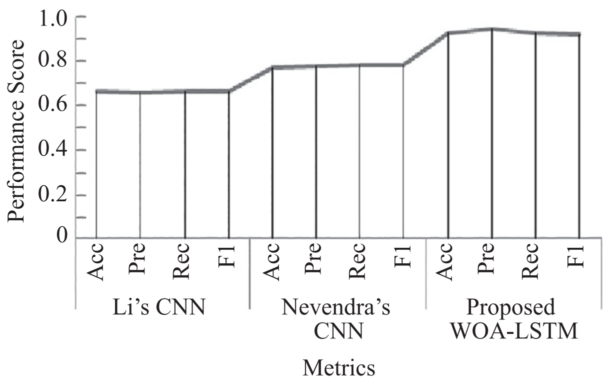

Fig. 6. Average performance score for all methods

the values: accuracy 0.666, precision 0.664, recall 0.67, and F1 0.666 (Li's CNN in Fig. 6); and existing CNN in [10] which achieved the values: accuracy 0.775, precision 0.782, recall 0.79, and F1 0.786 (Nevendra's CNN in Fig. 6).

Computational Complexity

The recommended model, WOA-LSTM, achieved the fastest run time of 112 s, beating 195 s in [6], and CNN 203 s in [10]. This means that the proposed technique significantly accelerated network training while also providing a more efficient DL model for early detection of software flaws. After ten separate runs, the average performance for the scenario of algorithm execution time is shown. The testing of the three techniques on DO2 software data is depicted in Fig. 7.

Discussion

This study presents an optimized DL technique, specifically the LSTM employing WOA, to improve the prediction performance of software problems.

Limitation

The research was limited to a simulation technique and did not cover model upgrades, integration with development tools, or developer cooperation. As a result, real-world deployment must be thoroughly investigated. Future research will look into the practical issues of deploying LSTM-based defect prediction models in real-world software development environments. Researchers might focus on making LSTM models more interpretable in the future, allowing developers and testers to understand why particular predictions are generated. The research relied on limited or specific datasets, which may have resulted in a lack of diversity in terms of software projects,

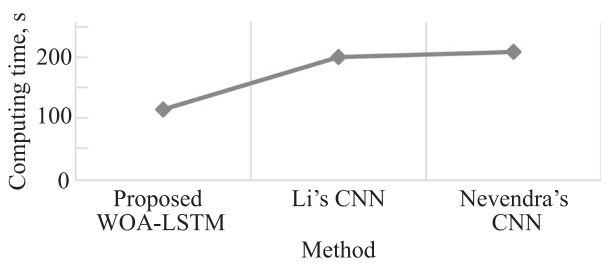

Fig. 7. Average converging time for all algorithms

programming languages, or sectors. This shortcoming may limit the suggested method applicability to numerous real-world settings. The study may have drawbacks due to the proposed method sensitivity to hyper parameters or specific setups. It may be necessary to investigate the method resilience in relation to different parameter values further.

The study provided an optimized DL methodology based on the WOA; however, it did not provide a full comparison to other state-of-the-art defect prediction models or approaches. The proposed method computing requirements may be a limitation. Future research could look at combining the proposed improved DL method with ensemble techniques. It would be possible to investigate how integrating various models could improve defect prediction accuracy and robustness. Future research should look on ways to make the model decisions more visible and intelligible, allowing software programmers to better grasp the reasoning behind defect predictions. An area of interest could be the development of a real-time defect prediction framework based on the optimal DL algorithm. LSTM could be used in SDP research to detect, mitigate, and assure fairness and ethical issues.

Recommendation

It is recommended in the future to look at transfer learning approaches that allow pretrained LSTM models (for example, on one project or domain) to be fine-tuned for defect prediction on various projects or domains. This may eliminate the requirement for large labeled datasets for each project. In the future, we should address the issue of imbalanced datasets in software prediction. Imbalanced datasets are widespread in this domain; therefore, research into how to employ LSTM networks successfully for such datasets is critical. This research also recommends investigating in the future how LSTM models may capture and use time-related features of software development. Understanding how historical data affects future defect prediction might be useful as software projects change over time. We also recommend in the future looking into incorporating diverse data modalities (for example, source code, bug reports, and version history) into LSTM-based models for defect prediction. Combining data from many sources has the potential to improve prediction accuracy.

Creating methods for estimating the uncertainty or confidence associated with LSTM-based predictions. Understanding when the model is uncertain might be essential for making decisions. The research also strongly recommends investigating ways to adapt LSTM models to changing software projects. In actuality, software projects grow, and models should be able to react to new data without retraining extensively. Look into approaches for protecting sensitive software data while utilizing LSTM models for fault prediction. It is critical to follow data protection regulations. Conduct comparative studies that compare LSTM-based approaches to other ML and classical defect prediction techniques in order to better understand the strengths and drawbacks of LSTM in various contexts. Investigate how LSTM models can be integrated into human-in-the-loop defect prediction systems, which combine machine predictions with human expertise.

Conclusion

The suggested system is an LSTM taught with the WOA to reduce training time while improving DL model efficacy and detection rate. In general, we can conclude that the suggested model runs in less than 2 minutes on the DO2 datasets. CNN was revealed to be the most sluggish [10] of all the algorithms tested. The WOA successfully limits premature convergence toward local optima and establishes the appropriate values for the LSTM weights and biases, which accounts for the technique's success. The results demonstrated that the WOA can increase convergence speed. The basic mechanism that helped this algorithm avoid the multiple local solutions to the difficulty of training DL algorithms was the random selection of prey in each selection. The property is inherited by the WOA-based trainer, which outperforms all classical CNN algorithms.

References

1. Wunsch A., Liesch T., Broda S. Groundwater level forecasting with artificial neural networks: a comparison of long short-term memory (LSTM), convolutional neural networks (CNNs), and non-linear autoregressive networks with exogenous input (NARX). *Hydrology and Earth System Sciences*, 2021, vol. 25, no. 3, pp. 1671–1687. https://doi.org/10.5194/hess-25-1671-2021
2. Conneau A., Schwenk H., Barrault L., Lecun Y. Very deep convolutional networks for text classification. *Proc. of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Vol. 1, Long Papers*, 2017, pp. 1107–1116. https://doi.org/10.18653/v1/e17-1104
3. Aljarah I., Faris H., Mirjalili S. Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing*, 2018, vol. 22, no. 1, pp. 1–15. https://doi.org/10.1007/s00500-016-2442-1
4. Lipton Z.C., Berkowitz J., Elkan Ch. A critical review of recurrent neural networks for sequence learning. *arXiv*, 2015, arXiv:1506.00019. https://doi.org/10.48550/arXiv.1506.00019
5. Xu Z., Li S., Xu J., Liu J., Luo X., Zhang Y., Zhang T., Keung J., Tang Y. LDFR: Learning deep feature representation for software defect prediction. *Journal of Systems and Software*, 2019, vol. 158, pp. 110402. https://doi.org/10.1016/j.jss.2019.110402

Литература

1. Wunsch A., Liesch T., Broda S. Groundwater level forecasting with artificial neural networks: a comparison of long short-term memory (LSTM), convolutional neural networks (CNNs), and non-linear autoregressive networks with exogenous input (NARX) // Hydrology and Earth System Sciences. 2021. V. 25. N 3. P. 1671–1687. https://doi.org/10.5194/hess-25-1671-2021
2. Conneau A., Schwenk H., Barrault L., Lecun Y. Very deep convolutional networks for text classification // Proc. of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Vol. 1, Long Papers, 2017. P. 1107–1116. https://doi.org/10.18653/v1/e17-1104
3. Aljarah I., Faris H., Mirjalili S. Optimizing connection weights in neural networks using the whale optimization algorithm // Soft Computing. 2018. V. 22. N 1. P. 1–15. https://doi.org/10.1007/s00500-016-2442-1
4. Lipton Z.C., Berkowitz J., Elkan Ch. A critical review of recurrent neural networks for sequence learning // arXiv. 2015. arXiv:1506.00019. https://doi.org/10.48550/arXiv.1506.00019
5. Xu Z., Li S., Xu J., Liu J., Luo X., Zhang Y., Zhang T., Keung J., Tang Y. LDFR: Learning deep feature representation for software defect prediction // Journal of Systems and Software. 2019. V. 158. P. 110402. https://doi.org/10.1016/j.jss.2019.110402

6. Li Z., Jing X.Y., Zhu X. Progress on approaches to software defect prediction. *IET Software*, 2018, vol. 12, no. 3, pp. 161–175. https://doi.org/10.1049/iet-sen.2017.0148
7. Dos Santos G.E., Figueiredo E. Failure of one, fall of many: An exploratory study of software features for defect prediction. *Proc. of the IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2020, pp. 98–109. https://doi.org/10.1109/SCAM51674.2020.00016
8. Zain Z.M., Sakri S., Ismail N.H.A., Parizi R.M. Software defect prediction harnessing on multi 1-dimensional convolutional neural network structure. *Computers, Materials and Continua*, 2022, vol. 71, no. 1, pp. 1521. https://doi.org/10.32604/cmc.2022.022085
9. Chen L., Fang B., Shang Z., Tang Y. Tackling class overlap and imbalance problems in software defect prediction. *Software Quality Journal*, 2018, vol. 26, no. 1, pp. 97–125. https://doi.org/10.1007/s11219-016-9342-6
10. Nevendra M., Singh P. Software defect prediction using deep learning. *Acta Polytechnica Hungarica*, 2021, vol. 18, no. 10, pp. 173–189. https://doi.org/10.12700/aph.18.10.2021.10.9
11. Ahmad A., Musa K.I., Zambuk F.U., Lawal M.A. Optimizing connection weights in a Long Short-Term Memory (LSTM) using Whale Optimization Algorithm (WOA): A review. *Journal of Science, Technology and Education*, 2022, vol. 10, no. 3, pp. 362–373.
12. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural computation*, 1997, vol. 9, no. 8, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

6. Li Z., Jing X.Y., Zhu X. Progress on approaches to software defect prediction // IET Software. 2018. V. 12. N 3. P. 161–175. https://doi.org/10.1049/iet-sen.2017.0148
7. Dos Santos G.E., Figueiredo E. Failure of one, fall of many: An exploratory study of software features for defect prediction // Proc. of the IEEE 20th International Working Conference on Source Code Analysis and Manipulation (SCAM). 2020. P. 98–109. https://doi.org/10.1109/SCAM51674.2020.00016
8. Zain Z.M., Sakri S., Ismail N.H.A., Parizi R.M. Software defect prediction harnessing on multi 1-dimensional convolutional neural network structure // Computers, Materials and Continua. 2022. V. 71. N 1 . P. 1521. https://doi.org/10.32604/cmc.2022.022085
9. Chen L., Fang B., Shang Z., Tang Y. Tackling class overlap and imbalance problems in software defect prediction // Software Quality Journal. 2018. V. 26. N 1. P. 97–125. https://doi.org/10.1007/s11219-016-9342-6
10. Nevendra M., Singh P. Software defect prediction using deep learning // Acta Polytechnica Hungarica. 2021. V. 18. N 10. P. 173–189. https://doi.org/10.12700/aph.18.10.2021.10.9
11. Ahmad A., Musa K.I., Zambuk F.U., Lawal M.A. Optimizing connection weights in a Long Short-Term Memory (LSTM) using Whale Optimization Algorithm (WOA): A review // Journal of Science, Technology and Education. 2022. V. 10. N 3. P. 362–373.
12. Hochreiter S., Schmidhuber J. Long short-term memory // Neural computation. 1997. V. 9. N 8. P. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Authors

Anes Aliyu Aihong — BSc, Abubakar Tafawa Balewa University (ATBU), Bauchi, 740272, Nigeria, https://orcid.org/0009-0009-5169-7593, Anesaliyu123@gmail.com
Badamasi Imam Ya'u — PhD, Senior Lecturer, Abubakar Tafawa Balewa University (ATBU), Bauchi, 740272, Nigeria, https://orcid.org/0000-0002-2710-8973, biyau@atbu.edu.ng
Usman Ali — PhD, Lecturer, Federal College of Education (Technical), Gomber, 760101, Nigeria, https://orcid.org/0000-0001-9645-3642, usmanali@fcetgombe.edu.ng
Abuzairu Ahmad — MSc, Abubakar Tafawa Balewa University (ATBU), Bauchi, 740272, Nigeria, https://orcid.org/0000-0003-0229-739X, Abuzairuahmad2020@gmail.com
Mustapha Abdulrahman Lawal — PhD, Principal Scientist, Abubakar Tafawa Balewa University (ATBU), Bauchi, 740272, Nigeria, https://orcid.org/0000-0002-1037-2022, musbaida@gmail.com

Авторы

Алию Айхонг Анес — студент, Университет Абубакара Тафавы Балева (ATBU), Баучи, 740272, Нигерия, https://orcid.org/0009-0009-5169-7593, Anesaliyu123@gmail.com
Имам Яу Бадамаси — PhD, старший преподаватель, Университет Абубакара Тафавы Балева (ATBU), Баучи, 740272, Нигерия, https://orcid.org/0000-0002-2710-8973, biyau@atbu.edu.ng
Али Усман — PhD, преподаватель, Федеральный педагогический колледж (технический), Гомбе, 760101, Нигерия, https://orcid.org/0000-0001-9645-3642, usmanali@fcetgombe.edu.ng
Ахмад Абузайру — студент, Университет Абубакара Тафавы Балева (ATBU), Баучи, 740272, Нигерия, https://orcid.org/0000-0003-0229-739X, Abuzairuahmad2020@gmail.com
Абдулрахман Лаваль Мустафа — PhD, главный научный сотрудник, Университет Абубакара Тафавы Балева (ATBU), Баучи, 740272, Нигерия, https://orcid.org/0000-0002-1037-2022, musbaida@gmail.com

Received 30.10.2023
Approved after reviewing 21.01.2024
Accepted 22.03.2024

Статья поступила в редакцию 30.10.2023
Одобрена после рецензирования 21.01.2024
Принята к печати 22.03.2024