Mobile Application Defect Prediction using Ensemble-Deep Learning Approaches: A Comprehensive Review

Divya Sharma
Department of Computer Science & Informatics, University of Kota, Kota (Raj.)

Reena Dadhich
Head of CSI, Department of Computer Science & Informatics, University of Kota, Kota (Raj.)

ABSTRACT

Mobile applications are playing a crucial role in today's digital era, across various sectors, including communication, e-commerce, healthcare, and governance. One of the most important problems that software engineers face when deploying these applications is to ensure their quality and reliability. Mobile Application Defect Prediction (MADP) is a new research area that focuses on predicting potential defects in mobile applications before it is released by applying computational intelligence techniques. The paper provides an extensive literature survey of SDP, especially the use of Machine Learning (ML), Deep Learning (DL) and Ensemble Learning techniques for mobile applications. This paper explores how SDP has developed from traditional statistical models to the more sophisticated models based on Artificial Intelligence (AI), and why hybrid deep learning and ensemble models are superior. Important algorithms like Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) and ensemble methods like Bagging, Boosting and Stacking are discussed. Existing research gaps are discussed — notably, the lack of any dedicated mobile application defect prediction model based on hybrid ensembles and deep learning. Finally, the paper presents the motivation and design rationale for a proposed Hybrid MADP-Model which combines CNN-LSTM and ensemble approaches to enhance the predictive accuracy, generalizability and robustness in various mobile application datasets.

General Terms

Software Testing, Artificial Intelligence

Keywords

Mobile Application Defect Prediction (MADP), Software Defect Prediction (SDP), Machine Learning (ML), Deep Learning (DL), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Ensemble Learning, Android Applications.

1. INTRODUCTION

Mobile applications are a necessity in today's life and people all over the world are using millions of mobile applications for communication, education, entertainment, ecommerce, governance and more on their mobile devices, smartphones and androids. With the number of mobile application ecosystems rapidly expanding, and the complexity of individual applications increasing by the day, the need for bug-free software prior to deployment to end users has never been greater. A minor functional defect or app crash could bring down the reputation of a business or government organization, leading to loss of users and costs [1].

Software Defect Prediction (SDP) is the technique of predicting the modules or components of software with history data and computational model that contain defects. The main goal of SDP is to have software development teams concentrate their testing resources on the most defect-prone components, minimizing the total amount of software development effort and increasing the quality of the software. Researchers estimate that almost 70% of software development costs are spent on software testing [2] and that predictive defect identification is an economically significant software testing practice.

Although the amount of research in SDP has been done for desktop and web-based applications, few studies have been targeted specifically for mobile applications. Mobile apps are unique in a number of ways compared to desktop apps: They are updated more often, run in a much wider variety of and less predictable environments, and receive feedback more quickly, in the form of app store reviews and ratings. These characteristics bring specific problems to the defect prediction, such as concept drift, class imbalance in defect databases, and diversity of mobile platforms [3].

Traditional SDP methods, including static code analysis, black-box test, glass-box test, and statistical regression model, have not been effective in dealing with the size and complexity of current mobile applications. Conventional approaches are time-consuming, less adaptive and are less efficient with large datasets, thus, the shift towards Artificial Intelligence (AI) based approaches. Machine Learning (ML) and Deep Learning (DL) techniques have shown great potential to overcome these limitations and provide more accurate predictions, better scalability and adaptability to changing software environments [4].

Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks are some of the successful architectures employed in SDP among deep learning techniques. CNN is particularly good at learning local spatial features from code representations, whereas LSTM learns the long-range temporal dependencies in a sequence of data. CNN is especially effective at learning local spatial features from code representations, while LSTM learns long-range temporal dependencies in a sequence of data. If used together in a hybrid architecture, models can make use of both spatial and sequential patterns in software metrics and software features. Moreover, ensemble methods like Bagging, Boosting, and Stacking could be used to increase the predictive power and generalization of the base models [5].

The present paper is a comprehensive review of current literature on SDP, where we have focused on mobile applications. Based on the analysis of the evolution of SDP methodologies from traditional statistical methods to

cutting-edge ensemble-deep learning methods, the paper points to the key research gaps and inspires the design of a novel Hybrid MADP-Model that integrates CNN-LSTM and ensemble learning for mobile application defect prediction.

2. RELATED WORK

Over the last 20 years, numerous studies have been carried out to investigate software defect prediction, which include a broad range of methods from classic statistical models to sophisticated deep learning architectures. This section provides a systematic overview of the important contributions in the field.

Catal and Diri [1] were one of the first systematic review of software fault prediction (SFP) studies, in which they analyzed research published in conference proceedings and journals. The study suggested that it is advisable that at least 80% of SDP studies should make use of public available datasets to make results reproducible and comparable. This process was the foundation of the SDP research standardized benchmarking.

Wang and Yao [2] investigated how Class Imbalance Learning (CIL) can be used to enhance SDP performance. They tested five imbalance handling approaches (under-sampling, threshold adjustment and Boosting-based ensembles) on ten real-world SDP datasets against two well-known classifiers (Naive Bayes and Random Forest). Five performance indicators, Probability of Detection, Probability of False Alarm, Balance, G-mean, and AUC were used in the study. The results showed that the CIL methods showing significant performance improvement in defect detection in highly imbalanced datasets.

However, Wahono [3] did a systematic literature review of SDP research from the year 2000 to 2013 and studied the trends in methodology, feature selection and algorithm used in SDP. The survey showed that Support Vector Machine (SVM) and Naive Bayes were top classifiers used and PROMISE repository datasets were the most widely used benchmarks. Ensemble techniques were also pointed out as being increasingly important in enhancing the accuracy of the predictions and was emphasized in the study.

Laradji et al. [4] showed that the performance of defect classification can be improved considerably by using feature selection along with ensemble learning. They investigated the data imbalance and feature redundancy problems and successfully identified the most informative features from the software data, resulting in a high classification accuracy. The key point of this study is to explain the importance of feature engineering as well as model selection in SDP.

In an interesting approach, Mehmood et al. [5] used feature selection along with multiple classifiers namely Random Forest, Logistic Regression, and Neural Networks on NASA datasets titled as CM1, JM1, KC1, KC2, and PC1 to enhance the accuracy with which defects are predicted. This was one of the first studies to systematically use feature selection as a pre-processing technique, which was specifically used to improve the performance of ML classifiers in the field of defect prediction.

Dam et al. [6] created a Tree-LSTM based defect prediction model which uses Abstract Syntax Trees (ASTs) of the source code to utilize both syntactic and structural information which is important for accurate defect prediction. Both within project and cross project prediction scenarios proved to be successful with data from the PROMISE repository and Samsung's open-source project.

In this study, Giray et al. [7] have carried out a systematic literature survey on 102 peer reviewed papers on the application of deep learning for SDP. They conducted qualitative and quantitative analysis and found CNN and LSTM architectures to be the most common DL models; and cross-project defect prediction is still a big open challenge, as domain variability and distribution shift across projects influence their performance.

The authors of [8] introduced the Deep learning-based defect prediction model for Android mobile applications, where they compared ANN, CNN and LSTM networks in a cross-project and within-project analysis setups, on 14 datasets. CNN obtained the best performance with AUC of 0.93, which provides a good baseline for deep learning defect prediction for mobile applications.

Odeh and Abu Taleb [9] proposed an ensemble-based deep learning system for intrusion detection in IoT networks, which fuses CNN, LSTM and Gated Recurrent Unit (GRU) models. CNN-LSTM hybrid model obtained 99.7% accuracy (F1 0.998), and CNN-GRU model got 99.6% accuracy. The present work showed the high potential of ensemble DL models for complex, high-dimensional data classification task.

To tackle this, Li et al. [10] proposed a deep learning-based framework called SYSEVR to detect software vulnerabilities, which was tested on two software vulnerability databases: the National Vulnerability Database (NVD) and the Software Assurance Reference Dataset (SARD). The framework was able to identify 15 vulnerabilities which were not previously found, of which 7 were unknown and reported to the vendors, confirming that DL models can be used in SWE.

Dong et al. [11] performed a comparative study on the performance of five ML algorithms, two DL approaches, and three ensemble techniques (Bagging, Boosting and Stacking) for software defect prediction. Boosting is the most efficient method with the highest AUC (0.99), G-Mean (0.96) and Average F1 Score (0.97) and the shortest prediction time as well.

To predict the stock price, Shi et al. [12] suggested a hybrid model of CNN-LSTM and XGBoost that incorporates attention mechanism. The attention mechanism embedded in CNN-LSTM structure also showed excellent performance in capturing the non-linear temporal relationship between them, which is a design concept that is directly applicable to the sequential software metric analysis in MADP.

Aggarwal et al. [13] proposed a voting ensemble of several pre-trained CNN models with LSTM model for human activity recognition from STAIR Actions dataset. Aggarwal et al. [13] proposed a multi-weighted CNN model combined with LSTM model for human activity recognition using STAIR Actions dataset. Ensemble model approach was superior to the individual models and demonstrated the wide-ranging applicability of getting the combination of CNN-LSTM ensemble models for sequential pattern recognition problems like Software Defect Prediction.

3. MACHINE LEARNING TECHNIQUES FOR SOFTWARE DEFECT PREDICTION

Machine Learning techniques have been at the forefront of software defect prediction research for over two decades. By

learning patterns from historical software metrics and defect data, ML models can predict whether a software module is defect-prone without requiring manual inspection. Table 1. displays the summary of key ML approaches studied in the literature.

Table 1. Summary of Machine Learning Techniques

Author(s) & Year	Technique Used	Dataset	Key Finding
Catal & Diri [1] (2009)	Naive Bayes, KNN	NASA MDP, PROMISE	80% studies should use public datasets
Wang & Yao [2] (2013)	Random Forest, Naive Bayes + CIL	10 SDP datasets	Ensemble + CIL improves AUC & G-mean
Wahono [3] (2015)	SLR of multiple ML models	PROMISE repository	SVM & Random Forest most commonly studied
Laradji et al. [4] (2015)	Ensemble + Feature Selection	Public SDP datasets	Feature selection + ensemble improves accuracy
Mehmood et al. [5] (2023)	RF, LR, Neural Networks	NASA: CM1, JM1, KC1, KC2, PC1	Feature selection boosts ML classifier accuracy

3.1 Classical ML Classifiers

The traditional machine learning techniques have widely been used in SDP including Naive Bayes, Support Vector Machine (SVM), k-Nearest Neighbors (KNN), Decision Trees, and Random Forest. In this group, Random Forest has consistently performed best in terms of accuracy as it is an inherently ensemble learning method, is less prone to overfitting and can be used in high dimensional feature spaces. One drawback of the Naive Bayes approach is that it assumes independence among the features, which is not true for software metrics data sets [1].

3.2 Class Imbalance Handling

One of the main problems in SDP with ML techniques is the class imbalance issue, in which the number of defective software modules is relatively low in the set of data but still low. Wang and Yao [2] showed that in the case of class-imbalanced databases, using any standard ML algorithms trained on the databases shows higher false negative rates. To overcome this problem, various techniques have been suggested: Synthetic Minority Over-sampling Technique (SMOTE), under-sampling and cost-sensitive learning are among the methods proposed to improve the detection of defective modules [2][4].

3.3 Feature Selection in ML

The selection of features is a crucial step to enhance the accuracy and effectiveness of a defect prediction model based on ML. Features may be redundant or irrelevant, adding to the computational load and affecting the performance of the model. Laradji et al. [4] demonstrated that feature selection combined with ensemble learning is crucial to enhancing the classification of defects. Further, Mehmood et al. [5] applied feature selection as a pre-processing scheme for various ML classifiers for NASA datasets with significant gains in the prediction rate.

4. DEEP LEARNING TECHNIQUES FOR SOFTWARE DEFECT PREDICTION

A recent paradigm that has had great success in SDP is Deep Learning (DL) which is able to learn high-level feature representations automatically from raw source code and software measures, without the need for handcrafted feature engineering. Unlike the conventional ML models, DL models can learn complex non-linear relationships. Table 2. presents the deep learning techniques used in SDP.

4.1 Convolutional Neural Networks (CNN)

The Convolutional Neural Networks have achieved good results on SDP, by making the source code and software metric vectors structured inputs that can be convolved to extract local patterns. Jorayeva et al. [8] presented the results of a comparison of the performance of 14 Android application datasets for the prediction of defects using various deep learning models; CNN was the best model used, with an AUC value of 0.93. CNN has the capability to learn the local feature patterns of codes, which is very suitable for defect indicator code pattern recognition.

Table 2. Summary of Deep Learning Techniques for SDP

Author(s) & Year	DL Model	Dataset/Domain	Result
Dam et al. [6] (2018)	Tree-LSTM	PROMISE, Samsung OSS	Syntactic + semantic info from ASTs improves SDP
Giray et al. [7] (2023)	SLR of DL models	102 peer-reviewed studies	CNN and LSTM dominant; cross-project SDP remains challenging
Jorayeva et al. [8] (2022)	ANN, CNN, LSTM	Android Applications (14 datasets)	CNN best: AUC = 0.93 for mobile defect prediction

Li et al. [9] (2021)	SYSEVR (DL-based)	NVD, SARD	Detected 15 previously unreported software vulnerabilities
Qasem et al. [16] (2020)	DL algorithms (LSTM, CNN, DNN)	PROMISE benchmark datasets	DL factors (layers, neurons) significantly affect SDP

4.2 Long Short-Term Memory Networks (LSTM)

A particular type of Recurrent Neural Network (RNN) is LSTM network, which is used to model long-range temporal patterns and sequential dependencies in data. LSTM is quite useful for SDP, as it is suitable to analyze the evolution of software metrics over time and code commitment histories. Qasem et al. [16] showed that the number of hidden units and the depth of LSTM layers can greatly affect the accuracy of predictions in SDP tasks. LSTM models have been able to attain accuracy rates of up to 87% on various benchmark sets.

4.3 Tree-LSTM and Structural Models

To incorporate tree structure, Dam et al. [6] adapted the LSTM architecture to form a Tree-LSTM model that uses Abstract Syntax Trees (ASTs) of source code. This allowed the model to take into account both the syntactic structure and semantic content of code, which are important features of defect-prone modules. Structural deep learning models for SDP were shown to be effective by results in both within-project and cross-project settings from the PROMISE repository and from Samsung's open-source dataset.

4.4 Deep Learning for Vulnerability Detection

Li et al. [10] proposed SYSEVR, a comprehensive framework for software vulnerability detection based on deep learning, which was tested on the National Vulnerability Database (NVD), and the Software Assurance Reference Dataset (SARD). SYSEVR was able to successfully report 15 previously unknown vulnerabilities by injecting slices of software code into Program Slices for Vulnerability (SyVC) representations and providing them as input to DL models. This shows the generality of the DL approach to software security assurance, and not just for SDP.

5. ENSEMBLED LEARNING TECHNIQUES FOR SDP

Ensemble learning approaches are various combinations of the predictions made by different base models to create a stronger and more accurate prediction. Such techniques are especially useful when an individual classifier can be prone to overfitting, class imbalance, or noise in its training set, as found in SDP. In the literature on the SDP, the three major ensemble strategies reviewed are Bagging, Boosting and Stacking. Table 3. gives the summary for ensembled techniques used in SDP.

Table 3. Summary of Ensemble Learning Techniques for SDP

Author(s) & Year	Ensemble Technique	Domain	Performance
Dong et al. [11] (2023)	Bagging, Boosting, Stacking	SDP datasets	Boosting: AUC 0.99, G-Mean 0.96, F1 0.97
Odeh & Abu Taleb [12] (2023)	CNN + LSTM + GRU (Ensemble DL)	IoT Intrusion Detection	CNN-LSTM: 99.7% accuracy; CNN-GRU: 99.6%
Aggarwal et al. [13] (2024)	Weighted Voting CNN-LSTM Ensemble	STAIR Actions Dataset	High accuracy in activity recognition; transferable to SDP
Shi et al. [14] (2022)	Attention CNN-LSTM + XGBoost	Stock price prediction	Hybrid attention model improves non-linear prediction accuracy

5.1 Bagging

Bagging (Bootstrap Aggregating) is a technique that trains a number of base classifiers using different random subsets of the training set and combines them using majority voting. Random Forest is a most widely-used bagging ensemble, which is highly accurate and does not suffer from overfitting. Dong et al. [11] demonstrated that the bagging-based methods outperform others at SDP on several benchmark datasets.

5.2 Boosting

Boosting sequentially trains weak learners with the next learner focusing more on the instances that were misclassified by the previous learners. In Dong et al. [11] comparative study Boosting got the best performance with an AUC score of 0.99, G-Mean of 0.96 and F1 score of 0.97 and also minimal computational time. Because of their excellent performance on imbalanced datasets, the AdaBoost and XGBoost variants of Boosting are widely used in SDP research.

5.3 Stacking

Stacking uses a meta-learner to fuse the predictions of several base classifiers, and trains the meta-learner with the characteristics of the data. The stacking method is more computationally expensive than both bagging and boosting, but it has the ability to learn from the strengths of different base classifiers, leading to better generalization. Laradji et al. [4] showed that ensemble learning in conjunction with feature selection is superior to individual classifiers on defect detection problems.

5.4 Deep Ensemble Models

In recent years, ensemble methods have been expanded to deep learning methods. Odeh and Abu Taleb [9] proposed an ensemble of CNN, LSTM and GRU based deep learning models for IoT IDS that attained 99.7% accuracy with

CNN-LSTM and 99.6% with CNN-GRU configurations. Aggarwal et al. [13] proposed a weighted voting approach to combine CNN-LSTM models for human activity recognition to show that the combination of multiple deep architectures significantly improves its performance as compared to any individual model. The results are very inspiring for the design of hybrid ensemble-deep learning model for mobile application defect prediction.

6. MOBILE APPLICATION DEFECT PREDICTION: A FOCUSED REVIEW

Mobile Application Defect Prediction (MADP) is a specific sub-domain of SDP with special focus on the characteristics of mobile software ecosystems. Mobile applications share a number of key distinctions from desktop applications that directly impact on the defect prediction strategies.

6.1 Characteristics of Mobile Applications

Mobile applications are characterized by:

- Frequent updates and rapid release cycles, leading to a higher occurrence of software defects over time.
- Diverse and less stable operating environments, including varying hardware configurations, OS versions, and network conditions.
- Strong user feedback sensitivity — user reviews on app stores directly reflect defect occurrences such as crashes, freezes, and functional errors.
- Significant class imbalance in mobile defect datasets, where defective modules constitute a small fraction of the total codebase.
- Limited applicability of traditional desktop SDP models due to differences in development frameworks, coding languages (Java, Kotlin, Swift), and mobile-specific APIs.

In the study on user complaints from mobile app stores by Khalid et al. [14] they discovered that they were mostly reporting functional errors and app crashes, highlighting the importance of defect prediction for mobile applications before they are deployed. Kaur et al. [15] compared the accuracy of code and process metrics for mobile application defect prediction, and concluded that mobile application process metrics like code changes and developer experience offer complementary predictive signals to traditional code metrics.

6.2 Deep Learning for MADP

In the first study of deep learning for defect prediction in mobile application [8], Jorayeva et al. [17] presented a study that performed a defect prediction study on 14 Android application projects. In the study the researchers compared the performance of ANN, CNN and LSTM models, both within and across projects. The CNN model consistently performed best with an AUC of 0.93, compared to other architectures. The study pointed out that deep learning models such as CNN are good for capturing complex feature interactions in mobile application code metrics, and cross-project prediction is still a big challenge to be explored in this field.

6.3 Comparison of MADP Approaches

As evident from Table 4, while individual ML and DL models have been evaluated for mobile application defect prediction, no study has yet proposed a dedicated hybrid ensemble-deep learning model specifically designed and validated for mobile applications across both within-project and cross-project settings on large-scale datasets.

Table 4. Comparative Analysis of MADP Approaches

Study	Technique	Dataset	Metrics	Limitation
Jorayeva et al. [8](2022)	CNN, LSTM, ANN	Android (14 sets)	AUC=0.93	No ensemble used
Kaur et al. (2015)	Code & process metrics	Mobile apps	Accuracy, Recall	Limited to metric analysis
Dong et al. (2023)	Boosting, Stacking, Bagging	General SDP	AUC=0.99, F1=0.97	Not mobile-specific
Mehmood et al. (2023)	ML + feature selection	NASA datasets	High Accuracy	Desktop-focused
Proposed MADP-Model	CNN-LSTM + Ensemble	Android (large-scale)	Accuracy, AUC, F1, Recall	Under development

7. RESEARCH GAP AND MOTIVATION

The review of the existing literature documents the following key gaps in the state of the art of SDP research, especially in relation to using mobile applications:

7.1 Non-Dedicated MADP Models

CNN and LSTM have been explored separately for mobile defect prediction, but no CNN-LSTM based models are specifically developed and tested for mobile applications which combine CNN-LSTM with ensembled models.

7.2 Ensemble Techniques

Ensemble methods like Bagging, Boosting and Stacking are widely used in desktop SDP, but have been seldom applied in the case of mobile application defect prediction in research papers.

7.3 Limited Generalizability

Most of the current mobile SDP research is conducted on project specific, small datasets, which restricts the findings from the research. Large and diverse datasets are urgently needed for models to be evaluated.

7.4 CPPC

Due to distributional differences among projects, Cross-Project Prediction Challenges (CPPC) (cross-project defect prediction, where a model trained on one project is used on another project) is an open challenge in MADP. The models that already exist are not generalizable.

7.5 Class Imbalance Problem

Class Imbalance Problem is well recognized in SDP, but not much has been addressed so far in the case of mobile application datasets.

7.6 No Hybrid Application Available

There are no applications of CNN's spatial feature extraction ability and LSTM's sequential ability in the mobile application defect prediction domain within a Hybrid Architecture, even with the addition of ensemble techniques.

These identified gaps collectively motivate the development of a novel Hybrid MADP-Model that integrates CNN-LSTM deep learning with ensemble techniques, evaluated on large-scale Android application datasets in both within-project and cross-project settings. Such a model is expected to significantly advance the state of the art in mobile application defect prediction.

8. PROPOSED HYBRID MADP-MODEL: AN OVERVIEW

These identified gaps inspire creation of a new Hybrid MADP-Model combining both CNN-LSTM deep learning and ensemble techniques tested on large-scale Android application datasets within project and across projects. This model is expected to make a great leap forward in the state of the art in mobile application defect prediction.

Given the identified research gaps, a Hybrid MADP-Model that combines Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks and ensemble learning is motivated in this paper for software defect prediction in mobile applications. The proposed model is a multi-stage model, which is described below:

Stage 1 — Data collection: Data from Android application datasets are gathered from the public repositories, with the intention of getting multiple datasets ranging from small to large in number to provide generalizability for the models.

Stage 2 — Data Pre-processing: Normalization and standardization of software metrics to prepare the data for model training.

Stage 3 — Feature Selection: In this stage, features that are relevant to the problem are selected and the ones that are redundant or irrelevant are eliminated to make the model more efficient.

Stage 4 — Class Balancing: Addressing the class imbalance problem is done through the use of SMOTE (Synthetic Minority Over-Sampling Technique) to avoid bias towards the majority class (the non-defective class).

Stage 5 — Hybrid Model Development: A CNN-LSTM hybrid architecture is developed, which combines the CNN to capture the local feature patterns, and LSTM to model the sequential dependencies across software metrics.

Stage 6 — Ensemble Integration: In this stage, the CNN-LSTM model is integrated with an ensemble technique such as Bagging, Boosting or Stacking to improve the accuracy and robustness of the predictions.

Stage 7 — Training, Testing and Evaluation: The model is tested with N-fold cross-validation and metrics used are accuracy, precision, recall, f1 score and AUC.

Stage 8 — Comparison: The suggested model is compared to existing state-of-the-art SDP models to check the superiority of the proposed model.

The proposed MADP-Model is supposed to be a generic model that can be used for both within-project defect prediction and cross-project defect prediction, for all kinds of mobile applications such as Personal Device Assistants (PDAs), government mobile applications, and commercial Android applications.

9. CONCLUSION

This paper provided a review of the software defect prediction (SDP) research in general and focused on mobile application defect prediction (MPDP). The study goes through the statistical and rule-based approach to SDP and into the AI based approach and evaluates the contribution from Machine Learning, Deep Learning and Ensemble Learning paradigms.

The review validated the effectiveness of AI-driven approaches, especially deep learning models like CNN and LSTM, in terms of defect prediction accuracy, scalability, and adaptability. Comparative studies have shown that ensemble methods, particularly the Boosting-based methods, show the best performance. A critical research gap was identified, however, there is no existing study suggesting any dedicated hybrid ensemble-deep learning model that can be used specifically for defect prediction in the mobile application domain on large datasets, not only within the project, but across projects as well.

The identified gaps strongly motivate the proposed Hybrid MADP-Model that combines CNN-LSTM with ensemble learning that is expected to potentially help develop a prediction accuracy, robustness and generalizability that is better. The role of reliable, pre-deployment defect prediction in the societal and economic gains of mobile applications in government, commercial, and social sectors is invaluable, as the applications proliferate. The following research steps are planned to design, implement, experimental test and compare the proposed Hybrid MADP-Model.

10. REFERENCES

[1] C. Catal and B. Diri, "A systematic review of software fault prediction studies," Expert Systems with Applications, vol. 36, no. 4, pp. 7346–7354, 2009.

[2] S. Wang and X. Yao, "Using class imbalance learning for software defect prediction," IEEE Transactions on Reliability, vol. 62, no. 2, pp. 434–443, 2013.

[3] R. S. Wahono, "A systematic literature review of software defect prediction," Journal of Software Engineering, vol. 1, no. 1, pp. 1–16, 2015.

[4] I. H. Laradji, M. Alshayeb, and L. Ghouti, "Software defect prediction using ensemble learning on selected features," Information and Software Technology, vol. 58, pp. 388–402, 2015.

[5] I. Mehmood, S. Shahid, H. Hussain, I. Khan, S. Ahmad, S. Rahman, N. Ullah, and S. Huda, "A novel approach to improve software defect prediction accuracy using machine learning," IEEE Access, vol. 11, pp. 63579–63597, 2023.

[6] H. K. Dam, T. Pham, S. W. Ng, T. Tran, J. Grundy, A. Ghose, T. Kim, and C.-J. Kim, "A deep tree-based model for software defect prediction," arXiv preprint arXiv:1802.00921, 2018.

[7] G. Giray, K. E. Bennin, Ö. Köksal, Ö. Babur, and B. Tekinerdogan, "On the use of deep learning in software defect prediction," Journal of Systems and Software, vol. 195, p. 111537, 2023.

[8] M. Jorayeva, A. Akbulut, C. Catal, and A. Mishra, "Deep learning-based defect prediction for mobile applications," Sensors, vol. 22, no. 13, p. 4734, 2022.

[9] A. Odeh and A. Abu Taleb, "Ensemble-based deep learning models for enhancing IoT intrusion detection," Applied Sciences, vol. 13, no. 21, p. 11985, 2023.

[10] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, and Z. Chen, "SYSEVR: A framework for using deep learning to detect software vulnerabilities," IEEE Transactions on Dependable and Secure Computing, vol. 19, no. 4, pp. 2244–2258, 2021.

[11] X. Dong, Y. Liang, S. Miyamoto, and S. Yamaguchi, "Ensemble learning based software defect prediction," Journal of Engineering Research, vol. 11, no. 4, pp. 377–391, 2023.

[12] Z. Shi, Y. Hu, G. Mo, and J. Wu, "Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction," arXiv preprint arXiv:2204.02623, 2022.

[13] S. Aggarwal, G. Bhola, and D. K. Vishwakarma, "Weighted voting ensemble of hybrid CNN-LSTM models for vision-based human activity recognition," Multimedia Tools and Applications, pp. 1–39, 2024.

[14] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, "What do mobile app users complain about?," IEEE Software, vol. 32, no. 3, pp. 70–77, 2014.

[15] A. Kaur, K. Kaur, and H. Kaur, "An investigation of the accuracy of code and process metrics for defect prediction of mobile applications," in Proc. 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), IEEE, pp. 1–6, 2015.

[16] O. A. Qasem, M. Akour, and M. Alenezi, "The influence of deep learning algorithms factors in software fault prediction," IEEE Access, vol. 8, pp. 63945–63960, 2020.

[17] M. Jorayeva, A. Akbulut, C. Catal, and A. Mishra, "Machine learning-based software defect prediction for mobile applications: A systematic literature review," Sensors, 2022.