3rd International Engineering Research Symposium 2025 (IERS 2025)
Colombo, Sri Lanka

Comparative Analysis of Deep Learning Models for Software Defect Prediction in Agile Environments

P. U. N. Pathirana[1] and W. V. S. K. Wasalthilaka[2]

[1]Department of Computing and Information Systems, Faculty of Computing, Sabaragamuwa University of Sri Lanka, Sri Lanka.
[2]Department of Software Engineering, Faculty of Computing, Sabaragamuwa University of Sri Lanka, Sri Lanka.
uthpalanp@gmail.com

Abstract. Software defect prediction is a critical part of development, identifying mismatches between expected and actual outcomes as detected by developers or end users. The main purpose of agile defect prediction is identifying defects in timely manner. But the iterative and fast paced nature of agile environments raises several challenges for defect prediction such as handling code changes, managing limited development time and addressing dynamic and evolving project requirements. So, there is a notable gap related to the research studies of agile defect prediction. Traditional defect prediction methods frequently struggle to identify dynamic and complex data patterns. This study performed a comparative analysis between deep learning models to identify the most effective model with the highest defect prediction accuracy. For the research, Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM) and Deep Belief Network (DBN) models are used which can identify complex data patterns and relationships by extracting the meaningful features automatically. Jira defect dataset was used and cleaned using data pre-processing and feature selection techniques, and processed dataset was divided into training and testing sets. The trained models were evaluated using metrics like accuracy, precision, recall and f1-score. The study exposes RNN outperforms other models with 80.78% accuracy, processing sequential data and predicting future risks by analyzing past defects effectively. The findings of the research emphasize the effectiveness of using deep learning models in agile software defect prediction for high-quality, reliable real-world agile development practices.

Keywords: Agile, Deep Learning, Software Defect Prediction, Software Development, Software Quality

1 Introduction

a. Background

Software systems have now become an indispensable part of our day-to-day lives. Almost every field such as education, health care, communication, transportation and also entertainment, is getting the benefits from using software systems. Due to the strong dependency on the software, the quality of the software is very crucial part to be necessarily considered. With its usage, growing size and complexity, it has become a major problem to be always concerned about how to improve the quality of software by reducing the software defects as much as possible [1].

Quality of the software is one of the crucial concerns of the software development process. Software quality is the process of measure how it is designed and how well the software conforms to that design. To build a standard quality software, testing is essential. If there is not proper testing of the software, it can include many defects which causes the system fails to perform its required functions.

IEEE standard defines a defect as "the lack to perform the tasks and system requirements that are provided by developers and customers" [2]. For the question of why the defects should be identified in the software projects, the answer can be provided with three major considerations: quality, performance and costs. A project with defects is normally a low-quality software due to poor performance and high maintenance costs. But if the project teams can develop a defect-free software, it can always increase the quality, improve its performance and reduce the maintain costs. Therefore, software defect prediction has become one of the most important part of the software engineering field.

Agile methodology is now gained more attention in many industries as most of the organizations are moving towards the adaptation of agile methodologies. Among all other software development methodologies, agile has become most effective and useful since it focuses to develop high quality software products with minimum defects and the defect management of agile environment should be involved in every development stage to gain the full confidence of the customer satisfaction [3].

Deep learning (DL) models are known for their specialties in software defect predication (SDP). They can automatically extract the features for the source code effectively by reducing the reliance on the manual feature engineering. And they can improve the prediction accuracy by outperforming the traditional machine learning methods. Deep learning models are famous for capturing the structural and semantic information from the code to handle complex data. Also, they are successfully address the challenges in cross project defect prediction by learning the transferable features efficiently.

b. Problem Statement and Objectives

Even though, different types of traditional techniques and methods have been used for SDP, there are still lack to handle complex data and frequent code changes which the DL models can easily do. Due to the iterative and fast-moving nature in agile environments, rapid changes in the system is done frequently, so it brings a significant challenge to test the system and find the defects in timely manner.

Therefore, an effective SDP is crucial point to maintain the quality of the software and reducing the cost which are associated to fix the defects late in the development life cycle. Even though, different types of DL approaches have been proposed, there is a lack of comprehensive comparative studies evaluating the performance of deep learning models in agile software defect prediction contexts. Therefore, the question of the research to address is which DL model for the SDP will be provided high accuracy in agile environments are arisen.

So, the main objective of the research is to compare different DL models to find which model is high accurately predicted the software defects in agile environments. To achieve the main objective, there are sub-objectives which are made to organize the research path in step-by-step manner. The first sub objective is, analyzing the effectiveness of DL models and traditional approaches in agile SDP by comparing both under same criteria. Second one is developing DL models using, CNN, RNN, DBN and LSTM algorithms to predict software defects. And third one is evaluating and comparing the accuracy of DL models and select the best performance model.

c. Significance of the study

Agile development is an environment where the frequent releases, quick iterations, and continuous integrations are happened. Because of these characteristics, it is crucial to identify software defects in timely manner to maintain the software quality and efficiency. This early identification of defects gives more benefits to the agile team such as, enhance task prioritization, efficient resource allocation, strengthen continuous delivery etc.

For the agile SDP, DL models contribute with the real-time defect prediction which accomplished the purpose of the agile defect prediction. Using DL can benefit in agile SDP as: early and accurate defect detection, reduce the manual effort, adaptability to evolving codebases etc. The research study is brought out the importance of the agile SDP by emphasizing the benefits of using DL for an effective prediction process.

2 Literature Review

Software defect prediction is one of the most popular research areas in software engineering field. Various number of research studies have been done by the researchers all around the world as it is now getting a more considerable and valuable area.

Hoang et al. [4] described that the software systems as the backbone of the economy and society, so the defects of those systems may substantially affect the businesses and people's lives in different ways. They suggested that it is best to identify and fix the defects early as possible because it would be difficult and costly as software grows in both size and complexity. Moreover, P. Deep Singh and A. Chug [5] emphasized that the quality is the most important aspect of a software as SDP can directly affect to the software quality which can glorify or ruin of the brand image of a company and deviation of time, cost and budget.

Recent research studies explore the different DL approaches for the SDP. According to S. Omri and C. Sinz [6], Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN) and Long Short-Term Memory (LSTM) based models are widely used and outperform traditional methods by which automates feature generation from source code. They emphasized that those models outperform traditional models by learning specific characteristics straight from the code.

M. Nevendra and P. Singh [7] proposed enhanced CNN model for SDP to enhance prediction accuracy and reduce overfitting compared to the traditional Machine Learning models, by emphasizing that CNN captures non-linear patterns and automates feature extraction. And Qiu et al. [8] also addressed the challenge of traditional cross project defect prediction (CPDP) methods neglect the semantic code patterns in favor of handcraft features. Therefore, they used the Transfer CNN which extracts the semantic feature.

E. Borandag [9] emphasized that RNNs are highly effective for the software fault prediction due to its ability to capture sequential dependencies in the code to learn from both structured and unstructured patterns, and to perform well on large datasets. And Fan et al. [10] proposed a DL framework to enhance the defect prediction by leveraging semantic and syntactic features from the source code.

Wang et al. [11] proposed a DL-based approach, to learn semantic features from the source code automatically to improve software defect prediction. They used DBN to extract semantic features from Abstract Syntax Trees (ASTs) and code changes. Because traditional defect prediction features fail to capture semantic differences between the programs, which leads to suboptimal performance.

Liang et al. [12] used a semantic LSTM model for defect prediction which processes token sequences in order to predict software defects, leveraging pooling layer to handle the variable-length inputs. Timperley et al. [13] also found that temporal dynamics of code

changes are omitted by the traditional approaches. They gave a solution as "AttentiveLSTM" which is an enhanced LSTM model that can learn long-term dependencies and automatically focus on the defect-prone commits.

Louis F and Saleh M [14] emphasized the nature of agile development environment as the iterative and fast-paced, so it often leads rapid changes in the code, making it challenging to find and address defects in timely manner. They described that the agile defect prediction becomes very crucial as it helps teams to anticipate the potential issues early in the development process. They also exposed that the agile teams can allocate resources more efficiently and streamline their testing efforts to deliver high quality software product.

This review of related research studies emphasizes the significance of SDP, precisely in agile environments. Many studies expose the importance of agile SDP and how DL models outperform traditional SDP techniques by adopting to special nature of agile. Therefore, this comprehensive literature review helped to identify the gap of the research area of lack of agile SDP studies using DL models which can be more beneficial in the software development industry.

3 Methodology

The research methodology includes several steps as it is represented through the following diagram (see Fig. 1).

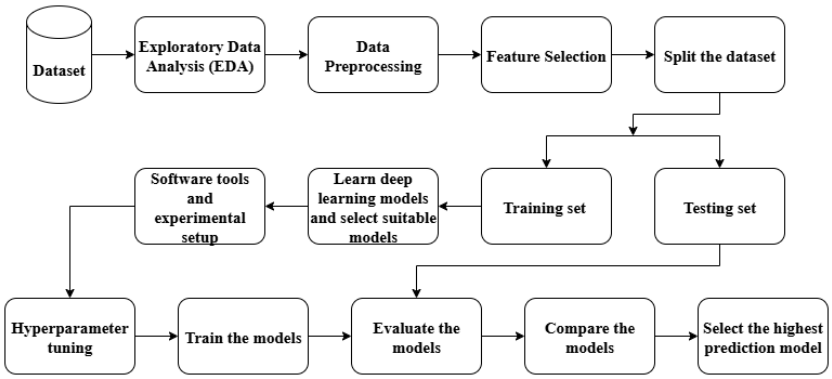

Fig. 1. Research Methodology

1) **Dataset:** A publicly available "Jira Defect Dataset" dataset from "Rnalytica" repository which can access via GitHub was used for the research study.

2) **Exploratory Data Analysis (EDA):** An extensive EDA was conducted to understand the structure, quality and underlying patterns of the dataset. For this, data size, defect distribution and the feature types were analyzed. Each data point includes code complexity metrics, code volume metrics, code churn metrics, commit activity metrics and defect labels. Each data record is labeled as either defect (1) or non-defect (0). A significant class imbalance was identified and, boxplots were used to identify the outliers and trends.
3) **Data Preprocessing:** Cleaned the dataset to get a high accurate prediction accuracy, data preprocessing part was done with several techniques such as handling missing values and duplicate data, handling outliers and reshaping the data for the models.
4) **Feature Selection:** Among 70 total columns of the raw dataset, 15 most relevant attributes were selected for prediction process (like AvgCyclomatic, CountLineCode, Added_lines, COMM, RealBug etc.) to reduce the redundancy of features and improve the performance.
5) **Split the dataset:** Dataset was divided into 2 sets as; training set and testing set.
6) **Learn the DL models and select suitable models:** By reviewing existing related studies, CNN, RNN, LSTM and DBN models were selected for the research.
7) **Software tools and experimental setup:** All the DL models were implemented using Python in Google Colab environment with TensorFlow and Keras libraries. Data preprocessing and analysis were performed using Pandas and NumPy. Matplotlib and Seaborn were used to generate visualizations. Evaluation metrics were computed using scikit-learn. Each model was designed with relevant standard setups: CNN with 1D Convolutional layers (ReLU and Sigmoid activations), DBN with Pretraining RBMs (Sigmoid activation), LSTM with two LSTM layers (ReLU and Sigmoid activations), and RNN with two SimpleRNN layers (tanh, ReLU and Sigmoid activations).
8) **Hyperparameter tuning:** To train the models, key hyperparameters were tuned as following to optimized the model performance:
    - Data Split (Train, Test): (85%, 15%), (80%, 20), (75%, 25%), (70%, 30%), (65%, 35%), (60%, 40%)
    - Batch sizes: 16, 32, 64, 128
    - Epoch count: 50, 100, 150, 200
    - Learning rate was tuned as 0.001, 0.0001, 0.0003 and 0.0005
9) **Train models:** All selected DL models were trained using training dataset.
10) **Evaluate models:** All the trained DL models were evaluated using testing dataset with accuracy, precision, recall and f1-score metrics.
11) **Compare the models:** All the models were compared with evaluation metrics wise.

12) **Select the highest prediction model:** After comparing the models, the model with highest accuracy was selected as the best prediction model.

4 Results and Findings

a. Effectiveness of Deep Learning vs Traditional approaches in Agile SDP

The research study compared the effectiveness of DL models with traditional SDP methods, as pointed out in the literature review part. This comparison was followed by the criteria which were derived from the categories mentioned in the existing related studies such as; challenges for agile SDP, the limitations of the traditional methods and the strengths of DL models. By analyzing these categories, four criteria for the comparison was identified. The following Table 1 gives the summary of the comparison.

Table 1. Comparison between the effectiveness of DL models vs Traditional models

Criteria	Deep Learning Approaches	Traditional Approaches
Feature Engineering	Automatic (learns raw code/commit data)	Manual (require expertise and time consuming)
Adaptability	Handles frequent changes via incremental learning	Struggle with dynamic requirements of agile projects
Sequential data	Specializes in temporal pattern recognition	Cannot model temporal dependencies effectively
Class imbalance mitigation	Provides data augmentation techniques to improve the prediction of minority (defective) class	Often resulting in poor minority class prediction due to bias towards the majority (non-defective) class

b. Model Performance Evaluation

The performance of each DL model can be presented according to the train-test split, in the Table 2 to Table 5.

The analysis shows that CNN model achieves its highest accuracy as 79.20% with an

60%-40% train-test split, in batch size 128 and 200 epochs using ReLU and Sigmoid activation functions (learning rate 0.0003).

Table 2. CNN Model's Performance

Train - Test Split	Batch Size	Epochs	Accuracy	Precision	Recall	F1-Score
85%-15%	16	150	74.80%	74.10%	76.27%	75.17%
80%-20%	64	200	77.73%	76.90%	79.29%	8.08%
75%-25%	64	200	78.58%	76.56%	82.38%	79.37%
70%-30%	128	200	78.26%	76.75%	81.07%	78.85%
65%-35%	64	200	77.70%	76.57%	79.82%	78.16%
60%-40%	128	200	**79.20%**	76.43%	84.45%	80.24%

Table 3. DBN Model's Performance

Train - Test Split	Batch Size	Epochs	Accuracy	Precision	Recall	F1-Score
85%-15%	16	100	**74.22%**	73.61%	75.51%	74.55%
80%-20%	64	150	73.19%	78.44%	63.96%	70.47%
75%-25%	64	50	73.84%	78.71%	65.38%	71.43%
70%-30%	128	150	73.53%	78.99%	64.13%	70.79%
65%-35%	64	200	73.44%	73.45%	73.42%	73.43%
60%-40%	64	200	73.83%	73.82%	73.82%	73.82%

The analysis shows that DBN model achieves its highest accuracy as 74.22% with an 85%-15% train-test split, in batch size 16 and 100 epochs using Sigmoid activation (learning rate 0.0005).

Table 4. LSTM Model's Performance

Train - Test Split	Batch Size	Epochs	Accuracy	Precision	Recall	F1-Score
85%-15%	128	200	73.48%	73.41%	73.65%	73.53%
80%-20%	128	100	75.06%	75.01%	75.15%	75.08%
75%-25%	64	200	74.63%	75.13%	73.65%	74.38%
70%-30%	128	200	75.30%	74.99%	75.92%	75.46%
65%-35%	16	200	76.61%	74.87%	80.11%	77.40%
60%-40%	128	200	**79.42%**	79.38%	79.48%	79.43%

The analysis shows that LSTM model achieves its highest accuracy as 79.42% with an 60%-40% train-test split, in batch size 128 and 200 epochs using ReLU and Sigmoid activation functions (learning rate 0.0005).

Table 5. RNN Model's Performance

Train - Test Split	Batch Size	Epochs	Accuracy	Precision	Recall	F1-Score
85%-15%	64	200	73.86%	72.93%	75.88%	74.38%
80%-20%	64	200	78.83%	78.32%	79.71%	79.01%
75%-25%	32	50	79.08%	77.56%	81.84%	79.64%
70%-30%	128	200	80.68%	79.25%	83.13%	81.14%
65%-35%	128	200	79.33%	79.17%	79.63%	79.39%
60%-40%	32	200	**80.78%**	78.51%	84.77%	81.52%

The analysis shows that RNN model achieves its highest accuracy as 80.78% with an 60%-40% train-test split, in batch size 32 and 200 epochs using tanh, ReLU and Sigmoid activation functions (learning rate 0.0003).

c. Comparison of the models

In order to provide a comprehensive assessment of the established DL models, performance of above DL model architectures was evaluated. By analyzing each performance, following graph shows the comparison of highest accuracy of each model (see Fig. 2.). **RNN model achieves an accuracy of 80.78%, outperforming all other models.** Following that, LSTM gave 79.42% accuracy, and CNN and DBN models followed 79.20% and 74.22% accuracy respectively.

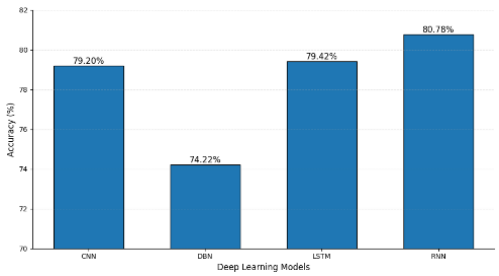

Fig. 2. Accuracy Comparison

The Fig. 3 to Fig. 5 show the Precision, Recall and F1-Score comparison between each model. It displays that LSTM achieve the highest precision value, and RNN achieve highest values for both recall and f1-score metrics.

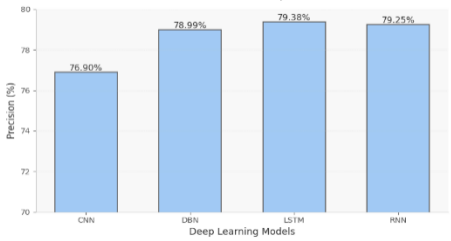

Fig. 3. Precision Comparison

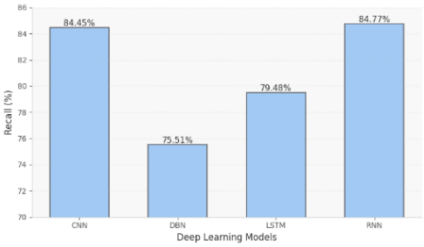

Fig. 4. Recall Comparison

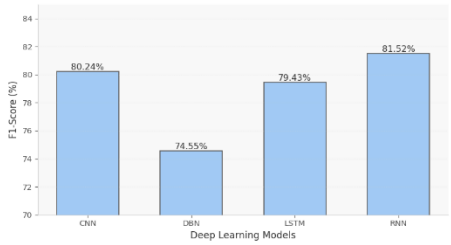

Fig. 5. F-Score Comparison

5 Discussion and Conclusions

The research study is conducted for a comparative analysis of DL models to identify their performance at SDP in agile environments. The research exposed the significance of using DL approaches to bridge the gap of lack of studies related to SDP in agile environments. During the research, class imbalance challenge was prevented by using data augmentation methods and overfitting was reduced by hyperparameter tuning over the models.

Research findings exposes that DL models performed noticeably better in agile environments compared to the traditional defect prediction methods. DL models have the ability to automatic feature engineering and they specialize in temporal pattern recognition.

They can handle frequent changes via incremental learning, and also handle class imbalance issues using data augmentation techniques. It reveals that these are hard to perform by traditional SDP methods in agile environment. Parallel to that, RNN model achieved 80.78% of its highest accuracy among all other models, emphasizing its outstanding ability to analyze the sequential data and predict the future defects risks by using historical patterns, which is crucial in fast-paced nature of agile environments.

Due to the special nature of agile environment, using DL models brings more advantages to the agile teams to identify defects in timely manner. This helps teams to prioritize tasks, allocate resources to the riskiest parts of the code, enhance sprint planning and improve software quality by supporting effective continuous development. In practically, these findings can be implemented into the issue tracking or CI/CD workflows of agile teams to enable the early detection of defects during sprints cycles, by integrating the selected DL model. To enhance the SDP accuracy, future research may focus on integrate project-level data and applying advanced architectures such as transformer-based models.

References

1. 2019 IEEE 5th International Conference on Computer and Communications (ICCC). IEEE, 2019.
2. M. Fokrul, I. Khan, A. Kader, and M. Masum, Predictive Analytics And Machine Learning For Real-Time Detection Of Software Defects And Agile Test Management, Theory And Practice, 4, pp. 1051–1057, 2024, doi: 10.53555/kuey.v30i4.1608.
3. R. Noor and M. Fahad Khan, Defect Management in Agile Software Development, International Journal of Modern Education and Computer Science, 6(3), pp. 55–60, Mar. 2014, doi: 10.5815/ijmecs.2014.03.07.
4. T. Hoang, H. Khanh Dam, Y. Kamei, D. Lo, and N. Ubayashi, DeepJIT: An end-to-end deep learning framework for just-in-time defect prediction, in IEEE International Working Conference on Mining Software Repositories, IEEE Computer Society, May 2019, pp. 34–45. doi: 10.1109/MSR.2019.00016.
5. P. Deep Singh and A. Chug, Software Defect Prediction Analysis Using Machine Learning Algorithms. IEEE, 2017.
6. S. Omri and C. Sinz, Deep Learning for Software Defect Prediction: A Survey, Proceedings - 2020 IEEE/ACM 42nd International Conference on Software Engineering Workshops, ICSEW Association for Computing Machinery, 2020, pp. 209–214. doi: 10.1145/3387940.3391463.
7. M. Nevendra and P. Singh, Software defect prediction using deep learning, Acta Polytechnica Hungarica, 18(10), pp. 173–189, 2021, doi: 10.12700/aph.18.10.2021.10.9.

8. S. Qiu, H. Xu, J. Deng, S. Jiang, and L. Lu, Transfer convolutional neural network for cross-project defect prediction, Applied Sciences (Switzerland), 9(13), 2019, doi: 10.3390/app9132660.
9. E. Borandag, Software Fault Prediction Using an RNN-Based Deep Learning Approach and Ensemble Machine Learning Techniques, Applied Sciences (Switzerland), 13(3), Feb. 2023, doi: 10.3390/app13031639.
10. G. Fan, X. Diao, H. Yu, K. Yang, and L. Chen, Software Defect Prediction via Attention-Based Recurrent Neural Network, Sci Program, 2019, doi: 10.1155/2019/6230953.
11. S. Wang, T. Liu, J. Nam, and L. Tan, Deep Semantic Feature Learning for Software Defect Prediction, IEEE Transactions on Software Engineering, 46(12), pp. 1267–1293, Dec. 2020, doi: 10.1109/TSE.2018.2877612.
12. H. Liang, Y. Yu, L. Jiang, and Z. Xie, Seml: A Semantic LSTM Model for Software Defect Prediction, IEEE Access, 7, pp. 83812–83824, 2019, doi: 10.1109/ACCESS.2019.2925313.
13. C. S. Timperley, L. Herckis, C. Le Goues, and M. Hilton, Understanding and improving artifact sharing in software engineering research, Empir Softw Eng, 26(4), Jul. 2021, doi: 10.1007/s10664-021-09973-5.
14. L. Frank and S. Mohamed, Training Machine Learning Models for Software Defect Prediction in Agile Development, 2024. [Online]. Available: https://www.researchgate.net/publication/380288702