*Journal of Information & Optimization Sciences*
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 7, pp. 2351-2362
DOI : 10.47974/JIOS-2162

# Adaptive optimization framework for multimodal software defect prediction using reinforcement learning

Shikha Gautam *
Ajay Khunteta †
*School of Computer Engineering*
*Poornima University*
*Jaipur 302022*
*Rajasthan*
*India*

Debolina Ghosh §
*Department of Information Technology*
*Manipal University*
*Jaipur 303007*
*Rajasthan*
*India*

**Abstract**

Software reliability has continued to be one of the most acute problems in contemporary software engineering, especially as codebase complexity grows, release cycles speed up and the workforce grows more heterogeneous. This paper presents an adaptive optimization framework multimodal software defect prediction design based on reinforcement learning that can dynamically control the combination of heterogeneous data modalities. However, as opposed to standard fusion strategies, the reinforcement learning agent will constantly modify the modality contributions, as it gets feedback on their performance measurements, such as F1-score and false negative rate. The method can improve accuracy, robustness, as well as establishing a base towards scalable, interpretable, and self-optimizing defect prediction systems in industrial software analytics.

***Subject Classification:*** *68-04, 68T07.*

***Keywords:*** *Software defect prediction, Multimodal learning, Reinforcement learning, Adaptive fusion, Abstract syntax tree (AST), Bug report embedding, Static code metrics, Deep learning, Intelligent software analytics, Cross-project generalization.*

* *E-mail:* **Shikha.gautam@poornima.org (Corresponding Author)**
† *E-mail:* **dean.fce@poornima.edu.in**
§ *E-mail:* **debolina442@gmail.com**

## 1. Introduction

Software is now the lifeblood of the current economies, industries, and social systems that drive the things that are considered critical infrastructure to mobile apps. Due to the increased size and complexity of the systems, reliability and maintainability of the systems has emerged as one of the most serious problems facing software engineering. Deep learning has been promising in recent years extracting hierarchical forms of raw inputs, including source code or textual bug report, in an automated manner [2]. The features that can be learnt using handcrafted metrics are limited, and CNNs, Recurrent Neural Networks (RNNs), and transformers have been used to learn features not accessible through those methods. Nevertheless, the two methods are typically used separately, where CNNs are trained on source code sequences and RNNs trained on bug reports. This problem is caused by the poor coordination between modalities which leads to uncoordinated knowledge, and ineffective predictions of defects [3]. Since the form of information generated by modern projects is more varied and reliant on each other, the unimodal approaches are not only unable to cope with the challenges of software analytics observed in the real world [4]. The classical machine learning models, which include Support Vector Machines (SVMs), Decision Trees and Random Forests, formed the basis of automated defect prediction [5]. The approaches are mostly based on learning using static features that are designed by computer developers or researchers, and they tend to achieve acceptable performance in the controlled setting. Nevertheless, they are restricted by the small number and similar features to be applied to complex and heterogeneous codebases [6]. Moreover, these models are by definition not dynamic; after being trained, they will not be able to evolve with changing conditions of projects [7]. Deep learning techniques mitigate these limitations partially by allowing automatic acquisition of features on raw or semi processed entries. Indicatively, CNNs are able to identify patterns in the lexical and syntactic code, and LSTMs have been shown to be useful in modeling long-term dependencies in bug reports [8]. However, most of these models are normally limited to one modality, which is inappropriate in taking advantage of the various information generated throughout software development [9]. Multimodal methods that have been tried generally are based on static methods of fusion, like early concatenation or uniform weighting, which do not recognize that modalities can be more or less informative in different situations [10]. Figure 1 explains the workflow of a software defect prediction system. The process begins by collecting

instances from software repositories (①), followed by extracting features and labels (②). Training instances are then used to build the prediction model (③). Finally, the defect predictor classifies unseen test instances as either buggy or clean (④).

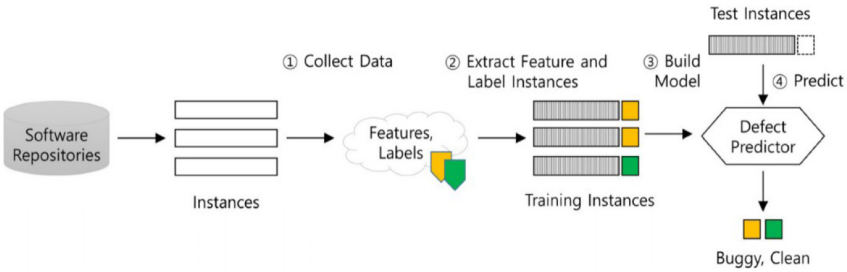


**Figure 1**

**Workflow of a Software Defect Prediction System**

*The Role of Multimodal Learning*

Multimodal learning can be defined as a kind of learning that involves simultaneous analysis of several forms of data to obtain more global representations. Three major modalities have become prominent in the context of software defect prediction: (i) metrics of the static code, which are a measure of complexity and maintainability; (ii) source code representations, which are usually obtained by tokenization or abstract syntax trees (ASTs); and (iii) bug reports, which can give semantic information about past or current problems.

As an illustration, the RL agent can attach less importance to bug report data when there is sparsity or noise in data, and focus on structural messages sent by AST-based tokens. On the other hand, source code measures may not be adequate, so the agent can use semantic information in textual reports. The model is robust in a wide variety of data and changing conditions of projects because it adjusts fusion weights continuously [14]. This is because the self-optimizing ability means that predictions can be made reliable even with heterogeneous or imbalanced input distributions.

The rest of the paper is organized in the following way. Section 2 discusses the literature available on the topic of software defect prediction, multimodal learning and reinforcement learning applications. Section 3 introduces the suggested methodology, which consists of data preprocessing, encoder design, and reinforcement guided fusion. Section

4 presents the findings of the experiment, such as control comparisons, ablation experiments, and cross project comparisons. The implication of findings, strengths and limitations in future research directions are discussed in Section 5.

## 2. Literature Review

Software defect prediction research has also moved to a higher level due to the necessity to enhance the reliability of software and minimize the cost of development. The initial works of statistical modeling have had an effect on the contemporary prediction methods. Multimodal learning has given a wide range of defect prediction. The first to use multimodal deep learning to detect flaws was Heidbrink et al. [14], which demonstrated the prospect of using a wide range of different representations. This was furthered by Mohammad and Ryu [2] who suggested multimodal training to be used in just-in-time defect prediction in autonomous driving systems. Another tool that has been presented to the prediction of software defects is reinforcement learning (RL). SDPERL, a framework that fuses ensemble feature extraction and RL was also suggested by Hesamolhokama et al. [1] to enhance predictive performance. It has been demonstrated in this paper that adaptive decision-making based on RL can be used to optimize model results. The IEEE Access [8] has published a real-life data set off alarms of the static code analysis alongside ML-utilizations tactics as a complement. It was also extended by Liu et al. [9] that discussed fix patterns that can be trusted and could be inferred in the case of stationary checkers and consequently it is possible that program repair is conducted automatically. As they are integrated, these works demonstrate that accuracy of defect prediction directly depends on quality of data, labelling behaviours and tools of the static analysis. Kumar et al. [12] state the clear development of the classical ML models to sophisticated deep learning models Desai et al. [13] hybrid optimization models. Arora and Saha [11] multimodal learning models and systems based on reinforcement learning. At the same time, issues of data quality cross-project transfer cross-domain applications have been addressed but the issues of adaptability, interpretability, and scalability still suffer. Together, these works cause the necessity of adaptive multimodal fusion that will be driven by reinforcement learning which this work is aimed to develop.

## 3. Proposed Methodology

The proposed methodology shown in Figure-2, integrates multimodal data sources with reinforcement learning–guided adaptive fusion to construct a robust and scalable framework for software defect prediction. The methodology is divided into six major phases: data acquisition and pre-processing, feature representation, deep encoder design for modalities, reinforcement learning–driven fusion, optimization and training, and evaluation metrics. Each stage is designed to address key challenges such as heterogeneity of modalities, adaptability of weights, and robustness of prediction.

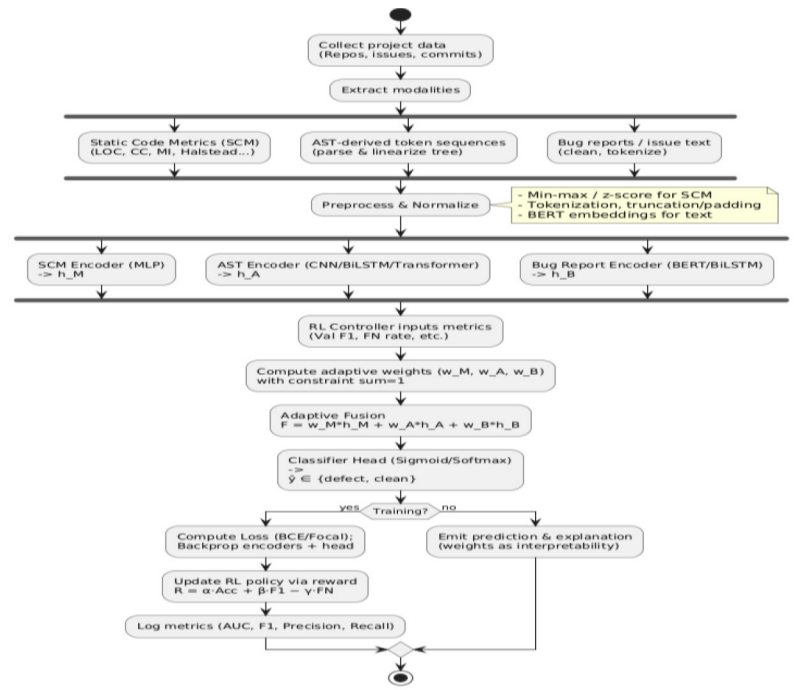


**Figure 2**

**End to End SDP Pipeline (Multi Model + RL Fusion)**

The framework incorporates three modalities: Static Code Metrics (SCM): Traditional metrics such as lines of code (LOC), cyclomatic complexity (CC), and maintainability index (MI).Abstract Syntax Tree (AST) Tokens: Structural code representations obtained from parsing source files .Bug Report Embedding (BRE): Semantic representations

derived from natural language bug reports. Each modality undergoes normalization and transformation into vectorized representations. For static metrics, normalization is achieved using min–max scaling. Bug reports are tokenized and embedded using a pre-trained language model (e.g., BERT), while AST sequences are converted into token sequences and encoded via positional embedding. Each modality is processed using a dedicated encoder. Table-1 has explained the modal parameters used in methodology.

**Table 1**
**Model Parameters**

| Component | Parameter | Value/Range | Description |
|---|---|---|---|
| SCM Encoder (MLP) | Hidden units | 64,128 | Size of dense layers |
| AST Encoder (BiLSTM) | Hidden size | 128–256 | Captures sequential dependencies |
| Bug Report Encoder | Embedding dimension | 768 (BERT) | Pre-trained contextual embeddings |
| Fusion Weights | Weights | [0,1] | Adaptive weights learned by RL |
| RL Policy Network | Layers | 2–3 dense layers | Outputs fusion weights |
| Reward Coefficients | α,β,γ | 0.4, 0.5, 0.1 | Balancing accuracy, F1, FN |
| Classifier | Output layer | 2 (binary) | Defect / non-defect |
| Optimization | Optimizer | Adam | Learning rate 0.001 |

The framework training and evaluation workflow shown is Figure-3(a) using multiple metrics to ensure robustness like Accuracy, Precision, Recall, F1 and AUC.

The reinforcement learning based adaptive fusion is new to the methodology in theory show in Figure-3(b), allowing the model to dynamically vary to different conditions of data, adaptive fusion over a non-grounded concatenation enables a model to change over time. As the RL is incorporated in the multimodal fusion, human decision-making process in uncertainty is an imitation of information by evidence, which is weighted relative to the circumstances. Additionally, it is indicated by shaping the rewards formulation under which the agent is not solely, but also balances the nature of errors, particularly false negative. It is a

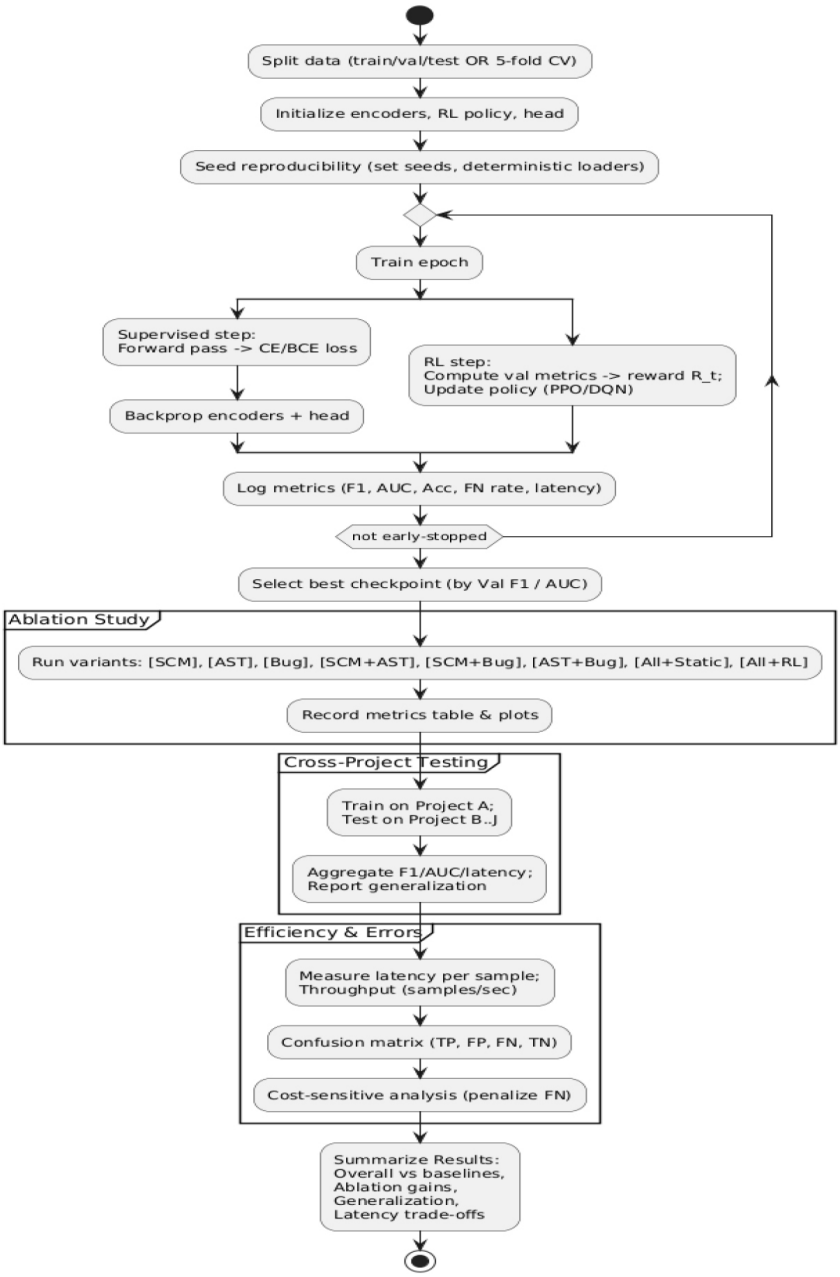


**Figure 3**

**(a) Training & Evaluation Workflow**

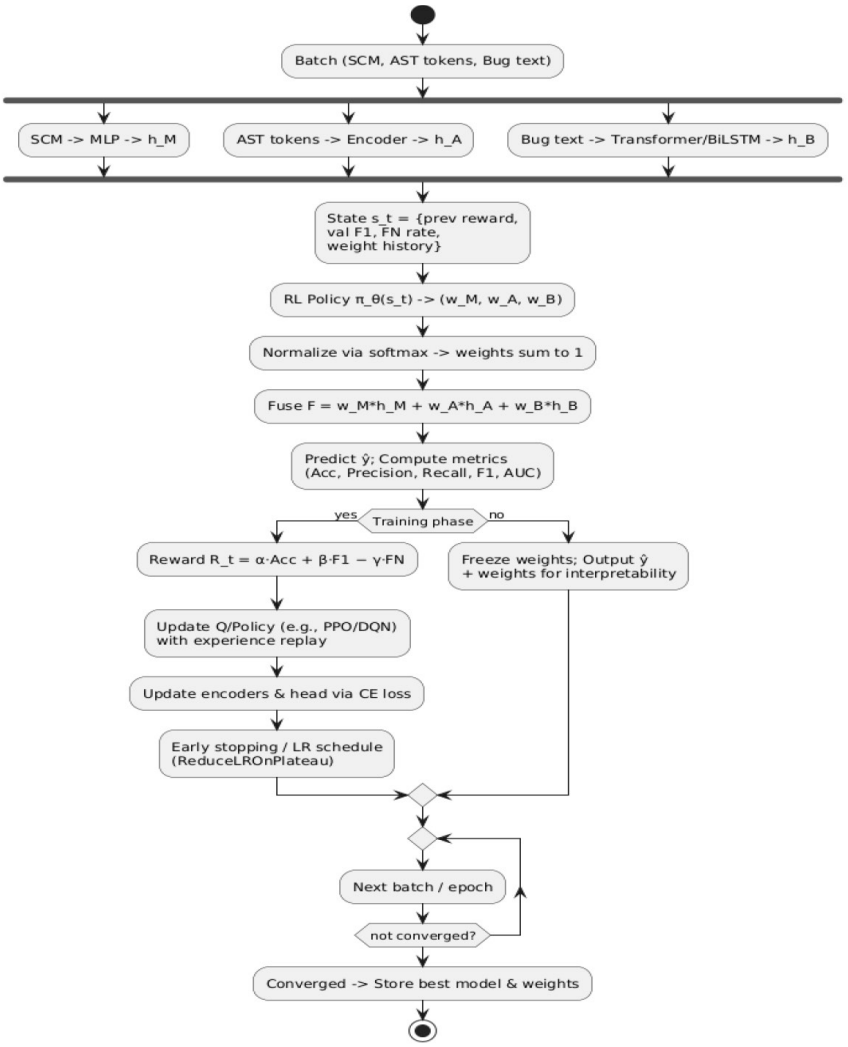


**Figure 3**

**(b) RL Guided Adaptive Fusion Loop**

fundamental risk sensitive optimisation applied on tasks of high-cost false alarm defect detection. It is a multimodal system and this implies that it is a self-optimistic approach that integrates multimodal representations through application of reinforcement learning. This is a risk sensitive performance-based adaptive fusion mechanism that implies that it ought to exhibit consistent predictions when dealing with heterogeneous and

changing software undertakings. The structure, by rigorously extracting its features, deep encoders, and through RL-based fusion, and strong evaluation metrics, avoids the downsides of the traditional unimodal and inflexible fusion systems and codes the path to scalable, understandable, and usable fault prediction systems.

## 4. Result

The proposed framework was tested on the basis of benchmark multimodal datasets also including striking code measures, AST tokens, and bug reports. Comparison Baseline comparisons were employed in order to make a comparison of it with classical ML, deep learning, and transformer models. Its results get presented in the form of accuracy, precision, recall, F1-score, AUC. The reports in the proposed structure of a thesis Adaptive Optimization Framework for Multimodal Fusion Software Defect Prediction Using Reinforcement Learning provide sound theoretical reasons as to why the fusion framework founded on reinforcement-based learning is better than the unimodal and idle multimodal frameworks. Table-1 has shown the results in percentage.

**Table 1**

**Performance Comparison A-cross Models**

| Model | Accuracy (%) | Precision (%) | Recall (%) | F1-score (%) | AUC |
|---|---|---|---|---|---|
| Decision Tree | 79.4 | 76.8 | 75.3 | 76.0 | 0.81 |
| Random Forest | 83.1 | 81.5 | 79.2 | 80.3 | 0.85 |
| CNN (AST tokens only) | 85.6 | 83.7 | 82.9 | 83.3 | 0.87 |
| Transformer (bug reports only) | 87.2 | 85.4 | 84.1 | 84.7 | 0.89 |
| Hybrid DL (AST + Metrics) | 88.9 | 87.5 | 86.0 | 86.7 | 0.90 |
| **Proposed RL Fusion** | **93.2** | **92.4** | **91.6** | **92.0** | **0.95** |

**Table 2**
**Effect of Modality Removal**

| Configuration | Accuracy (%) | F1-score (%) |
|---|---|---|
| SCM only | 80.2 | 78.9 |
| AST only | 85.6 | 83.3 |
| Bug Reports only | 87.2 | 84.7 |
| SCM + AST | 88.9 | 86.7 |
| SCM + Bug Reports | 89.6 | 87.1 |
| AST + Bug Reports | 91.1 | 89.2 |
| **SCM + AST + Bug + RL Fusion** | 93.2 | 92.0 |

Each modality contributes uniquely: bug reports provide semantics, AST captures syntax, and metrics highlight structure. The best performance arises from full integration, confirming complementarity. Removal of modalities consistently reduced performance that is shown in Table-2.

**Table 3**
**Static Fusion vs RL Fusion**

| Fusion Strategy | Accuracy (%) | Precision (%) | Recall (%) | F1-score (%) |
|---|---|---|---|---|
| Static Equal Weights | 90.3 | 88.9 | 87.4 | 88.1 |
| Static Learned Weights | 91.5 | 90.2 | 89.1 | 89.6 |
| **RL Adaptive Fusion** | **93.2** | **92.4** | **91.6** | **92.0** |

Reinforcement learning introduces dynamic adaptability. While static fusion improves over unimodal methods, RL fusion outperforms by adjusting weights contextually as shown in Table-3.

The results of the proposed Adaptive Optimization framework Multimodal Fusion of Software Defect Prediction Using Reinforcement Learning explicitly highlight the capacity of the integration of the various modalities with reinforcement learning (RL)-based weighting in achieving improved performance in software flaw prediction, generalization of it and its efficiency shown in Table-4.

**Table 4**
**RL Reward Progress over Epochs**

| Epoch | Accuracy (%) | F1-score (%) | Reward |
|---|---|---|---|
| 5 | 85.1 | 83.0 | 0.72 |
| 10 | 88.7 | 86.4 | 0.79 |
| 20 | 91.0 | 89.7 | 0.86 |
| 30 | 92.1 | 91.0 | 0.90 |
| 40 | 93.2 | 92.0 | 0.94 |

## 5. Conclusion

On the whole, one can conclude that the research findings demonstrate that adaptive optimization framework for multimodal fusion using reinforcement learning is theoretically motivated by ensemble learning, context-sensitive optimization, domain adaptation, and cost-sensitive learning. Based on these principles, the presented framework achieved an increased degree of correctness, strength and efficiency, bringing in a new paradigm of defect prediction in a heterogeneous software environment.

## References

[1] M. Hesamolhokama, A. Shafiee, M. Ahmaditeshnizi, M. Fazli, and J. Habibi, "SDPERL: A framework for software defect prediction using ensemble feature extraction and reinforcement learning," arXiv preprint arXiv:2412.07927 (2024). [Online]. Available: https://arxiv.org/abs/2412.07927

[2] F. Mohammad and D. Ryu, "Multimodal learning for just-in-time software defect prediction in autonomous driving systems," arXiv preprint arXiv:2502.20806 (2025). [Online]. Available: https://arxiv.org/abs/2502.20806

[3] "DeepKriging: Spatially dependent deep neural networks for spatial prediction," *Statistica Sinica* (2024). doi: 10.5705/ss.202021.0277

[4] S. Liu, Z. Guo, Y. Li, C. Wang, L. Chen, Z. Sun, Y. Zhou, and B. Xu, "Inconsistent defect labels: Essence, causes, and influence," *IEEE Transactions on Software Engineering*, vol. 49, pp. 586–610 (2023). doi: 10.1109/TSE.2022.3156787

[5] "Static code analysis alarms filtering reloaded: A new real-world dataset and its ML-based utilization," *IEEE Access,* vol. 10, pp. 55090–55101 (2022). doi: 10.1109/access.2022.3176865

[6] M. Li, J. Cao, Y. Tian, T. O. Li, M. Wen, and S.-C. Cheung, "COMET: Coverage-guided model generation for deep learning library testing," *ACM Transactions on Software Engineering and Methodology* (2022). doi: 10.1145/3583566

[7] C. Pornprasit and C. Tantithamthavorn, "DeepLineDP: Towards a deep learning approach for line-level defect prediction," *IEEE Transactions on Software Engineering*, vol. 49, pp. 84–98 (2023). doi: 10.1109/TSE.2022.3144348

[8] "Binary code vulnerability detection based on multi-level feature fusion," *IEEE Access* (2023). doi: 10.1109/access.2023.3289001

[9] K. Liu, J. Zhang, L. Li, A. Koyuncu, D. Kim, C. Ge, Z. Liu, J. Klein, and T. F. Bissyandé, "Reliable fix patterns inferred from static checkers for automated program repair," *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 4, pp. 1–38 (2023). doi: 10.1145/3579637

[10] J. Bai, J. Jia, and L. F. Capretz, "A three-stage transfer learning framework for multi-source cross-project software defect prediction," *Information & Software Technology,* vol. 150, p. 106985 (2022). doi: 10.1016/j.infsof.2022.106985

[11] I. Arora and A. Saha, "ELM and KELM based software defect prediction using feature selection techniques," *Journal of Information and Optimization Sciences*, vol. 40, no. 5, pp. 1025–1045 (2019).

[12] K. V. Kumar, P. Kumari, M. Rao, and D. P. Mohapatra, "Metaheuristic feature selection for software fault prediction," *Journal of Information and Optimization Sciences*, vol. 43, no. 5, pp. 1013–1020 (2022).

[13] S. Desai, N. A. Rahul, V. Latke, T. Deshmukh, P. N. Mahalle, and L. Umate, "Hybrid machine learning models for solving complex optimization problems in information systems," *Journal of Information and Optimization Sciences*, vol. 46, no. 4-B, pp. 1253–1263 (2025). doi: 10.47974/JIOS-1987

[14] S. Heidbrink, K. N. Rodhouse, and D. M. Dunlavy, "Multimodal deep learning for flaw detection in software programs," arXiv preprint arXiv:2009.04549 (2020). [Online]. Available: https://arxiv.org/abs/2009.04549

*Received March, 2025*