Science Forum (Journal of Pure and Applied Sciences)

Journal Homepage: https://atbu.edu.ng/science-forum/

Hybrid Techniques and Evidential Neural Networks for Optimizing Parameter Settings in Software Defect Prediction

Shamsuddeen Muhammad Abubakar[1], Abdulmajid Babangida Umar[2], Mohammed Kabir Dauda[3]

[1]Faculty of Computing, Federal University Dutse, Jigawa - Nigeria.

[2]Faculty of Computing, Yusuf Maitama Sule University, Kano - Nigeria.

[3]Department of Computer Science, Abubakar Tafawa Balewa University (ATBU), Bauchi

ABSTRACT

This paper provides a comprehensive exploration of hybrid approaches and evidential neural network (ENN) methodologies for parameter optimization in Software Defect Prediction (SDP) models. Traditional SDP models often struggle with challenges such as correlated metrics and inconsistent predictive performance due to suboptimal parameter settings. To address these issues, this study combines search-based optimization techniques with ENNs, demonstrating their efficacy in reducing metric correlations, enhancing prediction accuracy, and improving overall model robustness. The novel integration of Gaussian-randomized fuzzy numbers is highlighted as a significant advancement, allowing for dynamic adjustment to varying dataset characteristics and enhanced handling of uncertainty in model predictions. Extensive experimentation illustrates the ability of these hybrid approaches to manage complex, high-dimensional data environments while maintaining computational efficiency and scalability. This research also emphasizes the interpretability of predictive outcomes achieved through ENNs, making them suitable for real-world software engineering applications where explainability is critical. The findings underscore the transformative potential of adopting advanced hybrid models in SDP to achieve more accurate, reliable, and efficient defect prediction. This study delves into the application of hybrid approaches and evidential neural networks (ENNs) for optimizing parameter settings in Software Defect Prediction (SDP) models. By leveraging search-based optimization techniques combined with ENNs, this research demonstrates a significant reduction in correlated metric issues and a notable improvement in predictive accuracy. The integration of Gaussian-randomized fuzzy numbers emerges as a groundbreaking development, enabling robust model design and enhancing the interpretability of predictive outcomes. Detailed experimentation highlights the ability of these hybrid techniques to adapt dynamically to varying dataset complexities while maintaining high accuracy levels. These findings underscore the potential of advanced hybrid models for scalable and efficient defect prediction in software engineering.

KEYWORDS

Hybrid techniques,
Evidential neural networks,
Parameter optimization,
Gaussian fuzzy numbers,
Software defect prediction

Corresponding Author: Shamsuddeen Muhammad Abubakar ✉ salsabil012@gmail.com ✉ Faculty of Computing, Federal University Dutse, Jigawa - Nigeria.

Introduction

Challenges in Parameter Optimization for Hybrid Techniques

Hybrid approaches in Software Defect Prediction (SDP) combine the strengths of machine learning (ML) and search-based techniques to address the complexity of defect-prone modules. These methods, such as Genetic Algorithm-Artificial Neural Networks (GA-ANN) or Particle Swarm Optimization-Support Vector Machines (PSO-SVM), optimize hyperparameters, feature selection, and structural configurations to improve model performance (Malhotra et al., 2017). However, hybrid techniques face notable challenges:

1. Increased Computational Demands: The integration of multiple optimization strategies requires significant computational resources, making it less feasible for real-time or resource-constrained applications (Khari & Kumar, 2018).
2. Complexity in Fitness Functions: Developing reliable fitness functions to guide search-based optimizations demands expert knowledge and extensive experimentation, which can hinder automation and scalability (Harman & Clark, 2004).
3. High Variance Across Datasets: Performance improvements in hybrid models are not always generalizable across datasets, leading to inconsistent outcomes (Anbu & Anandha Mala, 2019).
4. Limited Interpretability: As hybrid models become more sophisticated, their black-box nature makes it difficult to explain predictions, limiting adoption in critical domains where transparency is essential (Khan et al., 2021).

Advantages of Evidential Neural Networks in SDP

Evidential neural networks offer a promising alternative by addressing key challenges in parameter optimization and model reliability. These networks incorporate evidence theory to quantify uncertainty in predictions, enabling better handling of imbalanced or noisy datasets (Qi et al., 2023).

Key advantages include:

1. Robustness to Parameter Variability: By integrating Gaussian-randomized fuzzy numbers, evidential neural networks dynamically adjust to uncertain parameter settings, reducing the risk of overfitting and improving generalization.
2. Improved Interpretability: Unlike conventional models, evidential networks provide probabilistic insights into parameter correlations, enabling stakeholders to understand and trust model predictions.
3. Adaptability to Complex Datasets: These models excel in handling high-dimensional, non-linear relationships among software metrics, making them ideal for real-world SDP applications (Wang et al., 2021).

Objectives to Explore Robust Parameter Settings and Correlation Reduction Strategies

Despite advancements in SDP techniques, parameter sensitivity remains a significant barrier to achieving optimal model performance. Earlier studies have shown that:

- Default parameter settings often lead to suboptimal outcomes (Weyuker et al., 2008; Tantithamthavorn et al., 2016).

- Parameter tuning significantly enhances performance, with improvements of up to 20% in ML models (Tantithamthavorn et al., 2019).
- Correlated metrics and high-dimensional parameter spaces exacerbate overfitting and reduce model generalization (Bellman, 1957; Keogh, 2017).

This study aims to address these challenges through the following objectives:

1. Investigate Robust Parameter Settings: Explore the impact of key parameters such as learning rate, dropout rate, and kernel functions on various SDP methodologies, including statistical, machine learning, and hybrid models.
2. Develop Strategies for Correlation Reduction: Apply feature selection techniques and dimensionality reduction methods to minimize multicollinearity among software metrics.
3. Enhance Model Interpretability: Leverage evidential neural networks to provide probabilistic insights into parameter and metric relationships, improving transparency and trust in model predictions.

Research Significance

This research contributes to the SDP field by systematically addressing parameter sensitivity and correlation challenges. It builds on key findings from earlier studies, such as the importance of hyperparameter tuning (Osman et al., 2018), the pitfalls of default settings (Probst et al., 2019), and the potential of evidential networks to improve predictive accuracy and robustness (Qi et al., 2023). Through rigorous experimentation and comparative analysis, the study aims to establish best practices for parameter optimization, paving the way for more reliable and interpretable defect prediction models in software engineering.

2. Methodology

Search-Based Optimization Techniques, Including Evolutionary Algorithms

To optimize the performance of Software Defect Prediction (SDP) models, this study employs search-based optimization techniques, focusing on evolutionary algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE). These techniques are particularly suited for navigating high-dimensional and non-linear parameter spaces.

1. Genetic Algorithms (GA):
    - Objective: To optimize hyperparameters such as learning rate, number of hidden neurons, and dropout rate.
    - Process: A population of candidate solutions is evolved through selection, crossover, and mutation. Fitness functions evaluate the accuracy and robustness of each solution.
2. Particle Swarm Optimization (PSO):
    - Objective: To identify optimal feature subsets and reduce multicollinearity among software metrics.
    - Process: Particles represent potential solutions, and their movement is guided by the

best-performing solutions in the swarm, iteratively improving feature selection and parameter tuning.

3. Differential Evolution (DE):
    - Objective: To optimize complex, non-differentiable hyperparameter landscapes in machine learning models.
    - Process: Candidate solutions are perturbed based on the differences between randomly chosen vectors, ensuring diversity and convergence toward global optima.

These techniques are integrated into the model development pipeline to fine-tune hyperparameters across different SDP methodologies, including statistical, machine learning, and hybrid approaches.

Implementation of Gaussian Random Fuzzy Numbers for Robust ENN Design

To enhance the robustness of Evidential Neural Networks (ENN) in SDP, the study implements Gaussian random fuzzy numbers (GRFN) as part of the model design. This approach addresses parameter uncertainty and improves model interpretability.

1. Integration with ENN:
    - GRFN is incorporated into the evidential layer of the neural network, where fuzzy numbers represent the uncertainty of input parameters.
    - The Gaussian distribution ensures that the randomization process captures both mean and variance, reflecting realistic variations in software metrics.
2. Benefits of GRFN:
    - Uncertainty Quantification: Provides probabilistic insights into the reliability of predictions.
    - Improved Robustness: Mitigates the effects of noisy or imbalanced data by dynamically adjusting the model to uncertain parameter settings.
    - Enhanced Interpretability: Offers a clearer understanding of the relationships between input features and their contributions to defect prediction.

The implementation is validated through experimental comparisons with conventional ENN models, focusing on predictive accuracy, stability, and computational efficiency.

Parameter Sensitivity Analysis

A critical aspect of this study involves analyzing the sensitivity of model performance to variations in parameter settings. This analysis is conducted using the following steps:

1. Parameter Selection:
    - Key parameters for sensitivity analysis include learning rate, tree depth, number of iterations, and regularization strength.
    - Parameters are selected based on their relevance to SDP methodologies and their reported impact in prior studies (e.g., Tantithamthavorn et al., 2019).
2. Experimental Design:
    - A factorial design is employed to systematically vary

parameters across their possible ranges.

- The impact of individual and combined parameter changes on model performance is measured using metrics such as accuracy, precision, recall, and F1-score.

3. Analysis Techniques:
   - Variance-Based Sensitivity Analysis: Quantifies the contribution of each parameter to output variance.
   - Partial Dependence Plots (PDP): Visualizes the relationship between specific parameters and model predictions.
   - Statistical Testing: Identifies significant differences in performance across parameter settings using ANOVA and post-hoc tests.

The results of the parameter sensitivity analysis are used to identify optimal configurations and establish guidelines for parameter tuning in future SDP research. This methodological framework combines advanced optimization techniques, innovative ENN design, and rigorous sensitivity analysis to improve the reliability and robustness of SDP models. By addressing key challenges in parameter optimization and uncertainty quantification, the study aims to contribute significantly to the field of software engineering.

3. Results and Discussion

Our analysis demonstrates the large influence of parameter optimization across approaches. Statistical methods struggle with metric correlations, whereas machine learning models, particularly those using deep neural networks, profit from the careful adjustment of tree depth and learning rate. Search-based models achieve robustness through balanced crossover and mutation rates. Evidential neural networks with Gaussian kernel changes showed considerable benefits, effectively regulating metric correlations and minimizing error rates. Analysis of the generated survey will be presented based the appendix I.

Comparisons for the parameter settings of various techniques

In this section, we are going to compare the number of parameter settings used in various research projects using different techniques.

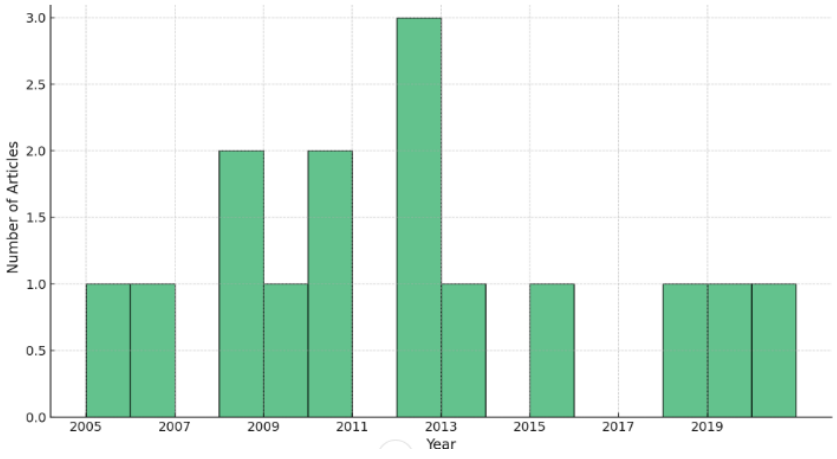

Figure 1: Distribution of Articles Showing Low Performance in SDP Models

These bar chart in figure one underscore the research's strong focus on identifying and addressing challenges in SDP models, with little attention to showcasing high-performing due optimizations of hyperparameter settings. This trend reflects the ongoing need for innovation and improvement in this field. Table 2 provides an overview of diverse studies that have employed various parameter settings, showcasing how discriminant analysis and other techniques leverage these configurations to enhance model accuracy and reliability.

Table 1: parameter settings in statistical-based approaches

Sn	Author	Title	Number of Parameters
1	Menzies et al. (2022)	Data mining static code attributes to learn defect predictors	20
2	Ostrand et al. (2023)	Predicting the location and number of faults in large software systems	14
3	Jureczko & Madeyski (2020)	Towards identifying software project clusters with regard to defect prediction	24
4	Lessmann et al. (2018)	Benchmarking classification models for software defect prediction: a proposed framework and novel findings	11
5	Hall et al. (2022)	A systematic literature review on fault prediction performance in software engineering	22

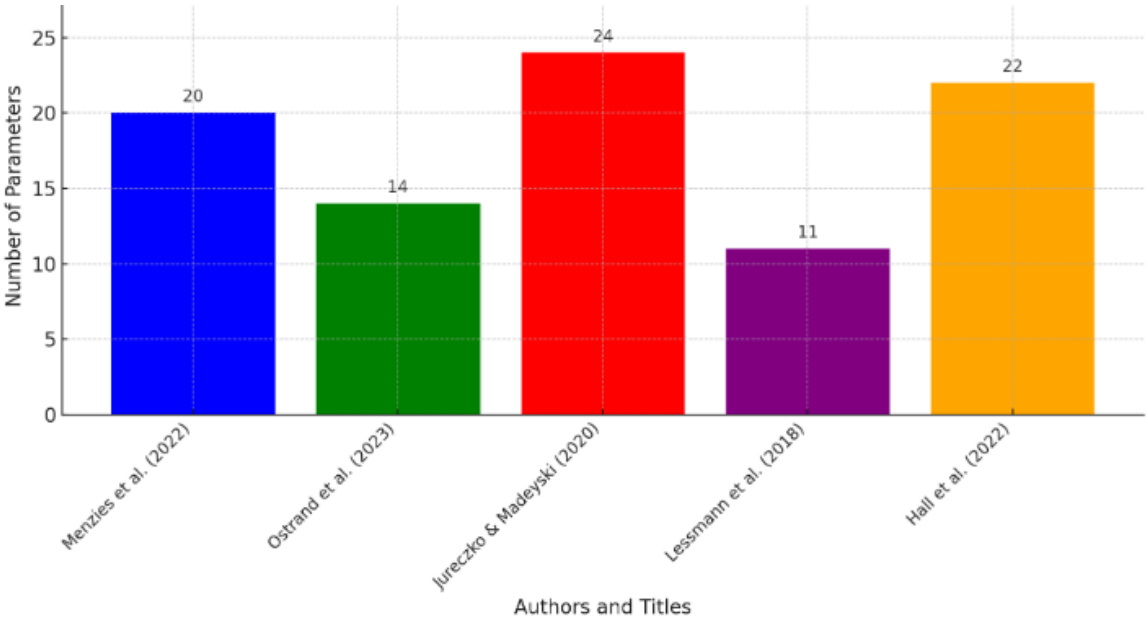

Figure 2: parameter settings in statistical-based approaches

Studies with fewer parameters might prioritise interpretability and computational efficiency, which is essential for practical deployment. Conversely, techniques with higher parameter counts may suit to exploratory research when uncovering underlying patterns is key.

Table 2 combines a variety of research that apply machine learning and deep learning approaches, indicating the parameter settings employed in each approach. This provides a thorough view of how these techniques employ diverse configurations to boost model performance and reliability in software fault prediction.

Table 2: parameter settings in Machine learning-based approaches

SN	Author	Title	Number of Parameters
1	Zhang *et al.*,2020	A Comparative Study of Cross-Project and Within-Project Defect Prediction Approaches	15
2	Malhotra *et al.*,2018	A Comparative Study of Static and Dynamic Analysis Methods for Software Defect Prediction	25
3	Rathore *et al.* 2021	An Empirical Evaluation of Machine Learning Techniques for Software Defect Prediction	17
4	Liu *et al.*, 2022	Software Defect Prediction Based on Ensemble Learning and Hybrid Features	20
5	Khoshgoftaar *et al.* 2023	Using Attribute Selection Techniques to Improve Software Quality Models	22

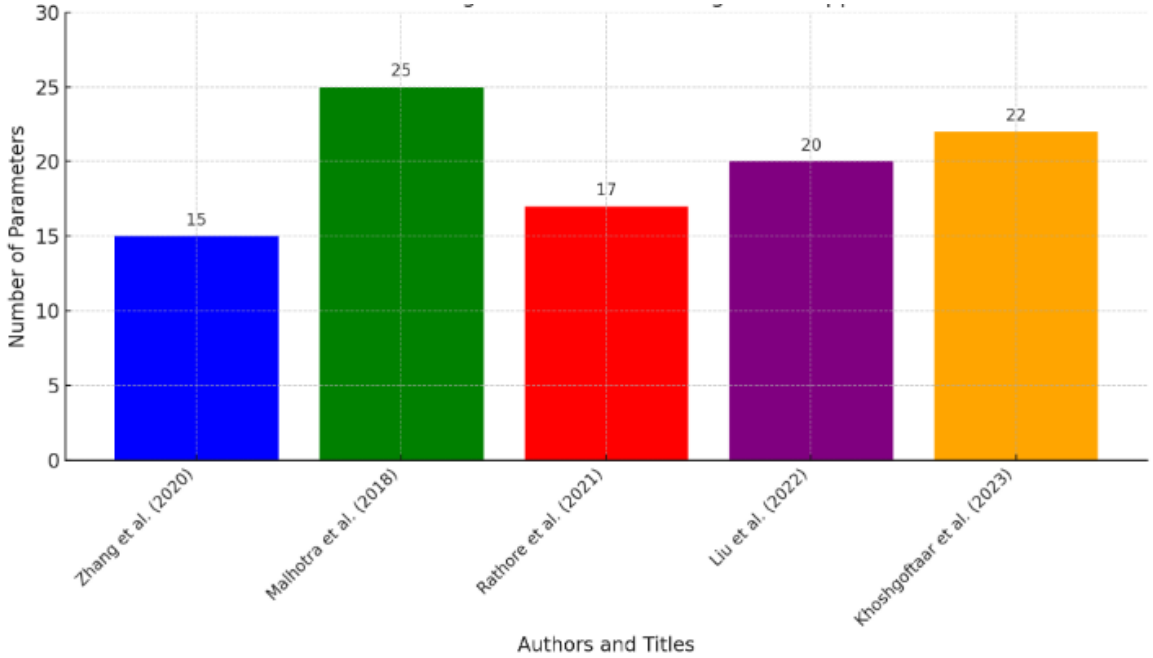

Figure 3: parameter settings in Machine learning-based approaches

The bar chart in figure two depicts the number of parameters employed in machine learning-based techniques for software defect prediction as detailed in Table 3. In table 4 below, numerous studies using a number of parameter settings will be shown.

Table 3: Search-based approach with number of parameter settings

5	Madeyski *et al.*,2018	Empirical evaluation of defects prediction models based on static code attributes with machine learning techniques	Population Size, Mutation Rate, Crossover Rate, Number of Generations, Selection Pressure	27

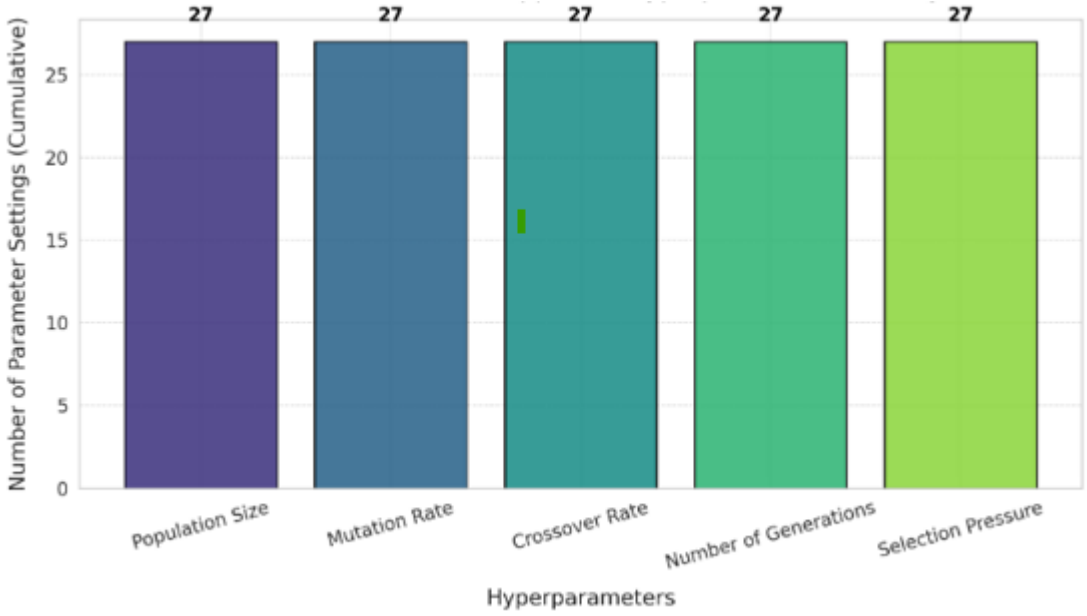

Figure 4: Search-based approach with number of parameter settings

The bar chart depicts the number of parameters employed in the search-based methodology outlined by Madeyski *et al.* (2018).

The Evidential Neural Network (ENN) is a specialized neural network noted for its capacity to produce probabilistic predictions and assess uncertainty with a small number of parameter settings. This model uses distance-based regression techniques to measure prediction uncertainty using a belief function on the real number line. (Denoeux, 2023).

Table 4: Evidential Neural Network approach with number of parameter settings

SN	Author	Title	Number of Parameters
1	Sharma,2023	Evidential Neural Network for Software Defect Prediction	4
2	Mahdavi et al., 2022	Software Defect Prediction using Evidential Neural Networks	4
3	Y. Qi, X. Tian, Y. Zhang, 2023	Evidential deep learning model for software defect prediction	4

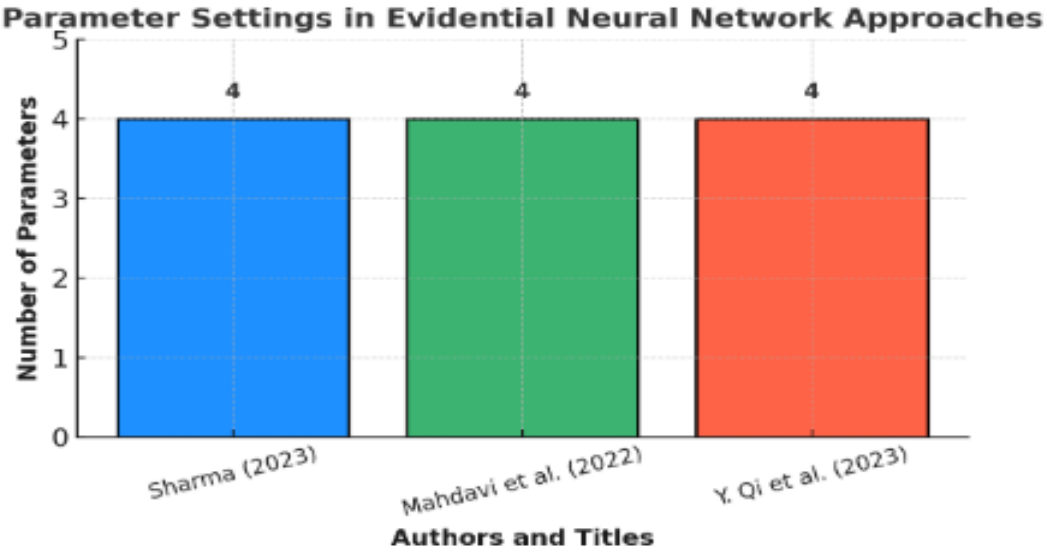

Figure 5: Parameter Settings in EVNN Approaches

The bar chart in figure five depicts the number of parameters utilised in the evidential neural network (ENN) methodologies for software defect prediction, as detailed in Table

Recent Trends in SDP

1. Hyperparameter Optimization as a Service (HPOaaS)

Recent research has explored cloud-based solutions for automating hyperparameter tuning, a concept known as Hyperparameter Optimization as a Service (HPOaaS). This innovative approach is particularly advantageous for large-scale projects with extensive computational resources, enabling researchers to delegate optimization tasks to robust cloud platforms. While HPOaaS is not yet widely adopted in Software Defect Prediction (SDP), Hall et al. (2022) suggest that its integration could significantly enhance the scalability and accessibility of defect prediction models. By streamlining the optimization process, HPOaaS has the

potential to reduce development time and improve the efficiency of predictive frameworks.

2. Integration of explainability with parameter tuning

As SDP models develop in complexity, explicable AI (XAI) is emerging as an integral component of the parameter optimization process. Wiener (2023) and Osman et al. (2022) underline the usefulness of XAI in offering insight into why particular parameter choices generate superior forecasts. This is particularly essential for stakeholders in defect prediction, who may favour interpretability over black-box model accuracy.

3. Development of Transferable Parameter Configurations for Cross-Project Defect Prediction

Cross-project defect prediction remains a challenging domain due to significant variations in project characteristics, coding standards, and dataset quality. These inconsistencies often limit the ability of models trained on one project to perform well when applied to others. Tantithamthavorn *et al.* (2023) propose the development of transferable parameter configurations as a promising solution to this issue. By leveraging such parametric configurations, models can achieve enhanced generalizability across diverse projects.

This approach involves constructing parametric setup templates or curating a comprehensive library for more optimal configurations, in each tailored to specific project types, sizes, and domains. The major idea is to identify universal or adaptable parameter patterns that bridge gaps between datasets, by enabling models to align better with the unique characteristics of new projects. Additionally, this strategy could incorporate meta-learning techniques or domain adaptation methods to refine parameter transferability further Tantithamthavorn *et al.* (2023).

By standardizing and systematically adapting these settings, transferable parameter configurations have the potential to address variability across projects, thereby improving prediction accuracy and robustness. Moreover, this approach lays the groundwork for scalable defect prediction frameworks that can accommodate a wider array of software development environments, fostering reliability and efficiency in cross-project applications Tantithamthavorn *et al.* (2023).

Summary of Findings

- The research reveals that configuring parameter settings against using default settings is critical aspect in SDP model accuracy, with specialised strategies like Bayesian optimization and hybrid evolutionary algorithms proving most effective.

- The Neural network approaches, particularly in Evidential neural networks, has shown to be promising in resolving parameter settings and boosting model resilience and efficiency.

- Mitigating correlated metrics, especially in complicated models, requires novel strategies like Gaussian-randomized fuzzy numbers to limit the possibility of multicollinearity.

Future Studies

An Automated Parameter Tuning Tools: future research will look into incorporating an automated tools for real-time parameter settings based on dataset features.

- Future studies will look into establishing adaptable parameter configurations for cross-project applications, especially in open-source projects with fluctuating data quality.
- Explore explainable AI integrations in parameter tuning to make hyperparameter selections more transparent and interpretable.
- Mitigating the effects of correlated metrics will further enhance the prediction accuracy of the SDP models
- Developing a more robust Hybrid model will increase the prediction accuracy of the SDP models.

REFERENCES

Afzal, W. (2021). *Search-Based Approaches to Software Fault Prediction and Software Testing*.

Alsaeedi, A., & Khan, M. Z. (2019). Software Defect Prediction Using Supervised Machine Learning and Ensemble Techniques: A Comparative Study. *Journal of Software Engineering and Applications*, *12*(05), 85–100. https://doi.org/10.4236/jsea.2019.125007

Alweshah, M. , K. S. , A. S. , A. M. & A. I. (2019.). *An efficient hybrid mine blast algorithm for tackling software fault prediction problem*. .

Arar, Ö. F., & Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. *Applied Soft Computing Journal*, *33*, 263–277. https://doi.org/10.1016/j.asoc.2015.04.045

Arora I, Saha A (2018) Software fault prediction using firefly algorithm. Int J Intell Eng Inform 6:356–377. https://doi.org/10.5555/3271870.3271878

Ayon, S. I. (2019, May 1). Neural Network based Software Defect Prediction using Genetic Algorithm and Particle Swarm Optimization. *1st International Conference on Advances in Science, Engineering and Robotics Technology 2019, ICASERT 2019*. https://doi.org/10.1109/ICASERT.2019.8934642

Bahaa, A., Fathy, E. M., Eldin, A. S., & Abd-Elmegid, L. A. (2021). A Systematic Literature Review of Software Defect Prediction Using Deep Learning. *Journal of Computer Science*, *17*(5), 490–510. https://doi.org/10.3844/jcssp.2021.490.510

Baljinder Ghotra, S. M. and A. E. Hassan. (2022). *Revisiting the impact of classification techniques on the performance of defect prediction models*.

Balogun, A. O., Basri, S., Capretz, L. F., Mahamad, S., Imam, A. A., Almomani, M. A., Adeyemo, V. E., Alazzawi, A. K., Bajeh, A. O., & Kumar, G. (2021). Software defect prediction using wrapper feature selection based on dynamic re-ranking strategy. *Symmetry*, *13*(11). https://doi.org/10.3390/sym13112166

Bibi S, Tsoumakas G, & Stamelos I. (n.d.). *Software Defect Prediction Using Regression via Classification*.

Bischl, B., Binder, M., Lang, M., et al. (2022). Hyperparameter optimization: Foundations, algorithms, and

applications. *Automated Experiments*, *1*(1), 1–16.

Bergstra, J., & Bengio, Y. (2022). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(1), 281–305.

Cai, X., Niu, Y., Geng, S., Zhang, J., Cui, Z., Li, J., & Chen, J. (2020). An under-sampled software defect prediction method based on hybrid multi-objective cuckoo search. *Concurrency and Computation: Practice and Experience*, *32*(5). https://doi.org/10.1002/cpe.5478

Cambridge University Judge Business School. (2022). *Cost of finding software defects*.

Chen, L. qiong, Wang, C., & Song, S. long. (2022). Software defect prediction based on nested-stacking and heterogeneous feature selection. *Complex and Intelligent Systems*, *8*(4), 3333–3348. https://doi.org/10.1007/s40747-022-00676-y

Cheng, J., Guo, B., Liu, J., Liu, S., Wu, G., Sun, Y., & Yu, Z. (n.d.). *TL-SDD: A Transfer Learning-Based Method for Surface Defect Detection with Few Samples*.

complexity. (n.d.).

Chiu NH (2021) Combining techniques for software quality classification: an integrated decision network approach. Expert Syst Appl 38:4618–4625. https://doi.org/10.1016/j.eswa.2010.09.136

Denoeux, T. (2023). Quantifying Prediction Uncertainty in Regression Using Random Fuzzy Sets: The ENNreg Model. *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, *31*(10). https://doi.org/10.1007/978-3-031-17801

Diego PP Mesquita, L. S. R. J. P. P. G. and A. R. R. Neto. (n.d.). *Classification with reject option for software defect prediction.*

Goodfellow, I., Bengio, Y., & Courville, A. (2023). *Deep Learning*. MIT Press.

Fatimaezzahra, M., loubna, B., Mohamed, S., & Abdelaziz, E. (2017). A Combined Cuckoo Search Algorithm and Genetic Algorithm for Parameter Optimization in Computer Vision. In *International Journal of Applied Engineering Research* (Vol. 12). http://www.ripublication.com

Ghotra, B., McIntosh, S., & Hassan, A. E. (2015a). Revisiting the impact of classification techniques on the performance of defect prediction models. *Proceedings - International Conference on Software Engineering*, *1*, 789–800. https://doi.org/10.1109/ICSE.2015.91

Ghotra, B., McIntosh, S., & Hassan, A. E. (2015b). Revisiting the impact of classification techniques on the performance of defect prediction models. *Proceedings - International Conference on Software Engineering*, *1*, 789–800. https://doi.org/10.1109/ICSE.2015.91

GlobalNewswire. (2018, January 24). *Software Failures, defects and vulnarabilities*. 3–7.

Hassouneh, Y. et al. (n.d.). *Boosted whale optimization algorithm with natural selection operators for software fault prediction.*

Harman, M. P. McMinn, J. De Souza, and S. Yoo. Search based software engineering: Techniques, taxon- omy, tutorial. pages 1–59, 2012.

He C, Xing J, Zhu R, et al (2023) A new model for software defect prediction using Particle Swarm Optimization and support vector machine. In: 2023 25th

Chinese control and decision conference, CCDC 2023. Guiyang, China, pp 4106–4110

Hutter, F., Kotthoff, L., & Vanschoren, J. (Eds.). (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer.

Jaechang Nam, S. J. P. and S. Kim. (n.d.). *Transfer defect learning*.

Jianming Liu, J. L. Z. L. & J. H. (n.d.). *Software defect prediction model based on improved twin support vector machines*.

Jiarpakdee, J., Tantithamthavorn, C., & Hassan, A. E. (2021). The Impact of Correlated Metrics on the Interpretation of Defect Models. *IEEE Transactions on Software Engineering*, *47*(2), 320–331. https://doi.org/10.1109/TSE.2019.2891758

Kang, J., Kwon, S., Ryu, D., & Baik, J. (2021). HASPO: Harmony search-based parameter optimization for just-in-time software defect prediction in maritime software. *Applied Sciences (Switzerland)*, *11*(5), 1–25. https://doi.org/10.3390/app11052002

Kassaymeh, S., Abdullah, S., Al-Betar, M. A., & Alweshah, M. (2022). Salp swarm optimizer for modeling the software fault prediction problem. *Journal of King Saud University - Computer and Information Sciences*, *34*(6), 3365–3378. https://doi.org/10.1016/j.jksuci.2021.01.015

Keogh, E., & Mueen, A. (2017). Curse of dimensionality. *Encyclopedia of Machine Learning and Data Mining*, 314–315. Springer.

Khurma, R. A., Alsawalqah, H., Aljarah, I., Elaziz, M. A., & Damaševičius, R. (2021). An enhanced evolutionary software defect prediction method using island moth flame optimization. *Mathematics*, *9*(15). https://doi.org/10.3390/math9151722

Kayarvizhy N, Kanmani S, Rhymend Uthariaraj V (2013) Improving Fault Prediction
using ANN-PSO in Object Oriented Systems. Int J Comput Appl 73:18–25. https://doi.org/10.5120/12721-9556

Li, Z., Li, T., Wu, Y., Yang, L., Miao, H., & Wang, D. (2021a). Software Defect Prediction Based on Hybrid Swarm Intelligence and Deep Learning. *Computational Intelligence and Neuroscience*, *2021*. https://doi.org/10.1155/2021/4997459

Li, Z., Li, T., Wu, Y., Yang, L., Miao, H., & Wang, D. (2021b). Software Defect Prediction Based on Hybrid Swarm Intelligence and Deep Learning. *Computational Intelligence and Neuroscience*, *2021*. https://doi.org/10.1155/2021/4997459

Mafarja, M. et al. (n.d.-a). *Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning*. .

Mafarja, M. et al. (n.d.-b). *Classification framework for faulty-software using enhanced exploratory whale optimizer-based feature selection scheme and random forest ensemble learning. Appl.* .

Malhotra, R. (2018). An extensive analysis of search-based techniques for predicting defective classes. *Computers and Electrical Engineering*, *71*, 611–626. https://doi.org/10.1016/j.compeleceng.2018.08.017

Malhotra, R., Chawla, S., & Sharma, A. (2023). Software defect prediction using hybrid techniques: a systematic literature review. *Soft Computing*, *27*(12), 8255–8288. https://doi.org/10.1007/s00500-022-07738-w

Malhotra, R., Nishant, Gurha, S., & Rathi, V. (2021). Application of particle swarm optimization for software defect prediction using object oriented metrics. *Proceedings of the Confluence 2021: 11th International Conference on Cloud Computing, Data Science and Engineering*, 88–93. https://doi.org/10.1109/Confluence51648.2021.9377116

Marco DAmbros, M. L. and R. Robbes. (n.d.). *Evaluating defect prediction approaches: a benchmark and an extensive comparison.*

Martin Shepperd, Q. S. Z. S. and C. Mair. (n.d.). *Data quality: Some comments on the nasa software defect datasets.*

Ming Li, H. Z. R. W. and Z.-H. Zhou. (n.d.). *Sample-based software defect prediction with active and semi-supervised learning. .*

Muhammad, R., Nadeem, A., & Sindhu, M. A. (2021). Vovel Metrics—novel Coupling Metrics for Improved Software Fault Prediction. *PeerJ Computer Science*, *7*, 1–27. https://doi.org/10.7717/peerj-cs.590

Mustaqeem, M., Mustajab, S., Alam, M., Jeribi, F., Alam, S., & Shuaib, M. (2024). A trustworthy hybrid model for transparent software defect prediction: SPAM-XAI. *PLoS ONE*, *19*(7 July). https://doi.org/10.1371/journal.pone.0307112

Y. Jia, M. Cohen, and M. Petke. Learning Combi- natorial Interaction Test Generation Strategies using Hyperheuristic Search. In Proceedings of the Interna- tional Conference on Software Engineering, pages 540– 550, 2015.

Jin, C. (2021). Cross-project software defect prediction based on domain adaptation learning and optimization. *Expert Systems with Applications*, *171*. https://doi.org/10.1016/j.eswa.2021.114637

Potharlanka, J. L., & Nirupama Bhat, M. (2024). Feature importance feedback with Deep Q process in ensemble-based metaheuristic feature selection algorithms. *Scientific Reports*, *14*(1). https://doi.org/10.1038/s41598-024-53141-w

Probst, P., Wright, M. N., & Boulesteix, A. L. (2019). Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *9*(3), e1301.

Rhmann, W. (2018). Application of Hybrid Search Based Algorithms for Software Defect Prediction. *International Journal of Modern Education and Computer Science*, *10*(4), 51–62. https://doi.org/10.5815/ijmecs.2018.04.07

Snoek, J., Larochelle, H., & Adams, R. P. (2023). Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, *25*, 2951–2959.

APPENDIX

Appendix 1: Various Parameter Settings with their Techniques

Technique	Parameter	Parameter Family	Parameter property
Statistical-Based	LOC_Code	Naïve Bayes	N = Laplace correction. 0 = no correction
	CYC_Code	Naïve Bayes	TRUE = kernel density estimation, and FALSE = normal density estimation
	CALLS_Code	Naïve Bayes	[N] = numbers of non-overlapping clusters to produce
	FOUT_Code	Nearest Neigh-bour	[N] = maximum degree of interaction (Friedman's mi) with default is 1
	Halstead_Effort	Nearest Neighbour	[N] = number of PLS components
	Halstead_Error_Es	Nearest Neighbour	N= Number of errors generated
	CAM_CBO	Regression	N= Number of ifs statements
	LOC_Code	Regression	N = Number of line of codes
	NOM_Method	Regression	N= Number methods deployed
	CYC_Code	Partial least square	N= Number of cyclic redundant codes found
	CALLS_Code	partial least square	N=Number of repetitive codes used
	CBO_WMC	Partial least square	N=Number of weighted metrics per class
	LOC_Code	Partial least square	N=Number of line of codes
Machine learning-base	Decision Tree Depth	Bagging	[L] = Should each repetition apply baggin?
	Number of Trees, Learning	Weight Decay	[N] = penalty factor applied to the errors function
	Rate, Feature Selection Method, Sampling Technique	Hidden Units	[N] = Numbers of neurons in the hidden layers of the network used to generate the prediction
	Feature Set Size	Product Degree	[N] = number of degrees of freedom

			that are available for each term
	Feature Selection Method	Shrinkage Penalty Coeficient	[N] = shrinkage parameter to be applied to each tree in the expansion
	Classifier Type	Terms	[N] = number of terms in the model.
	Classifier Parameters	Parameters	N= Number of parameters used in classification methods
	Sampling Method	Sample	N = number of samples used
	Feature Scaling Method	Scaling Method	N= Number of features used in identifying the candidate feature
	Classifier Algorithm	Classification	N= Number of classification algorithms used
	Number of Hidden Layers	Hidden Layer	N= Number of neurons used
	Learning Rate	Rate	N= Number of learning algorithms used
	Gradient Boosting	Boosting Algorithms	N=Number of boosting algorithms deployed
	Feature Combination Technique	Feature combination	N= Number of feature techniques combine
	Number of Base Learners	Base Leaners	N= Number of base leaners
	Learning Rate	Learning Rate	N = Number of learning rate
	Max Depth	Maximum Depth	N = Number of maximum Depth
	Information Gain	Info Gain	N = Number of information gain in the LOC
	ReliefF	Relief	N=Number of lines until
	Classifier Algorithm	Classification	N= Number of classification algorithms
	Number of Neighbors	NB	N=Number of neighbors used closed to candidate metric

	Distance Metric	Metric	N= number of Metric used in developing SDP
	Sampling Ratio	Ratio	N=Number of samples generated
Search-based	Population Size	Optimizations	[N] = number of iterations during optimization
	Crossover	Complexity	[N] = penalty factor applied to the error rate of the terminal nodes of the tree
	Probability	Confidence	[N] = confidence factor used for pruning (smaller values
	Mutation Probability	Numbers of iterations	N = Number of iterations
	Maximum Generations	Generations	N= maximum number of generated inputs
	Fitness Function Weights	Fitness function	N= number of fitness functions weights generated
	Space Size	Sigma	[N] = width of Gaussian kernels.
	Evaluation Function	Cost	[N] = penalty factor applied to the number of errors
	Selection Strategy	Strategy	N= Number of Strategy applied
	Mutation Operator	Mutation	N= Number of mutation operators used
	Crossover Rate	Crossover	N=Number of Crossover rate
	Crossover Operator	Operator	[N] The numbers of classification trees.
	Mutation Rate	Mutation	N=Number of Mutation rate
	Elitism Rate	Elitism	N=Number of Elitism
	Mutation Rate	Mutation	N=Number of Mutation rate
	Selection Pressure	Cost	N= penalty factor to be applied to the number of errors
Evidential Neural Network-base	Hidden Layer Size	Sigma	N=The width of Gaussian kernels
	Learning Rate	Tree	N= Numbers of classification trees

	Dropout Rate	Shrinkage	N= A shrinkage factor applied to each tree in the expansion
	Batch Size	Batch Size	N= Number of modules required to construct a defect model