JURNAL TAMPIASIH
INSTITUT TEKNOLOGI DAN KESEHATAN ASPIRASI
E-ISSN : 2986 - 8742

ANALISIS PENERAPAN ARTIFICIAL INTELLIGENCE DALAM DETEKSI BUG DAN PREDIKSI KESALAHAN PERANGKAT LUNAK: A SYSTEMATIC LITERATURE REVIEW

Nurul Umami[1], Nurhasanah[2], Nurhasiyah[3] dan Susi Susilawati[4]
[1]Teknologi Informasi, Institut Teknologi dan Kesehatan Aspirasi, Indonesia
[2,3,4]Teknologi Informasi, Universitas Mataram, Indonesia

Email : nurulumamy27@gmail.com[1], nurha789110@gmail.com[2], nurhasiyah685@gmail.com[3], sasqiaerawati@gmail.com[4]

ABSTRAK

Pengembangan perangkat lunak saat ini mengalami pergeseran paradigma seiring dengan masifnya integrasi teknik Kecerdasan Buatan (AI) dan *Machine Learning* (ML). Perkembangan sistem yang sangat cepat menuntut mekanisme deteksi kesalahan secara berkelanjutan, sementara metode pengujian tradisional masih bersifat reaktif, memakan waktu, dan membutuhkan sumber daya besar. Tinjauan pustaka sistematis ini menyintesis penelitian terkini mengenai deteksi bug cerdas dan prediksi kesalahan berbasis AI. Kami menganalisis arsitektur lanjutan termasuk LSTM, *Graph Neural Network* (JITGNN), *Large Language Model* (LLM), serta algoritma optimasi metaheuristik (ILFA). Kajian ini menyoroti peran *Explainable AI* (XAI) dalam meningkatkan transparansi keputusan serta teknik transformasi fitur seperti Algoritma Genetika dan FeatBoost. Meskipun menunjukkan peningkatan kinerja yang signifikan, tantangan seperti ketidakseimbangan data, biaya pelabelan manual, dan generalisasi lintas proyek masih menjadi hambatan utama penerapan di industri.

Kata Kunci: *Deteksi Bug, Prediksi Kesalahan, Kecerdasan Buatan, Machine Learning, Explainable AI.*

ABSTRACT

Software engineering is undergoing a paradigm shift driven by the integration of Artificial Intelligence (AI) and Machine Learning (ML) techniques. Rapid software updates demand continuous and proactive fault detection, whereas traditional testing methods are often reactive, time-consuming, and resource-intensive. This systematic literature review synthesizes recent state-of-the-art research on AI-based intelligent bug detection and fault prediction. We analyze advanced architectures including LSTM, Graph Neural Networks (JITGNN), Large Language Models (LLMs), and metaheuristic optimization algorithms (ILFA). The review highlights the role of Explainable AI (XAI) in improving decision transparency and feature transformation techniques like Genetic Algorithms and FeatBoost. Despite significant performance gains, challenges such as data imbalance, expensive manual labeling, and cross-project generalizability remain critical barriers to industrial adoption.

Keywords: *Bug Detection, Fault Prediction, Artificial Intelligence, Machine Learning, Explainable AI.*

***Corresponding author:** nurulumamy27@gmail.com*

PENDAHULUAN

Seiring meningkatnya kompleksitas arsitektur perangkat lunak modern, risiko terjadinya *bug* dan kegagalan sistem (*system failure*) menjadi tantangan kritis yang berpotensi menimbulkan kerugian finansial, penurunan produktivitas, hingga ancaman keamanan [1]. Pada era otomasi saat ini, keberhasilan rilis produk sangat bergantung pada seberapa cepat dan akurat pengembang dalam menemukan cacat kode sebelum sistem digunakan oleh pengguna akhir [2]. Namun, pendekatan pengujian tradisional yang mengandalkan inspeksi manual sering kali tidak mampu mengejar laju iterasi pengembangan yang sangat cepat [5].

Hadirnya inovasi di bidang Kecerdasan Buatan (*Artificial Intelligence* / AI) dan *Machine Learning* (ML) membawa pendekatan baru yang lebih proaktif dan presisi [3]. AI mampu mempelajari pola kompleks dari data historis, seperti skrip *source code*, log eksekusi, metrik performa, dan laporan masalah (*issue tracking system*) [3], [4]. Melalui pemrosesan data masif ini, AI tidak hanya mendeteksi keberadaan *bug* secara otomatis, tetapi juga mampu mengklasifikasikan tingkat keparahannya (*severity*) serta memprediksi area kode yang paling rentan terhadap kesalahan (*fault-prone*) [3], [4].

Meskipun potensi penerapannya sangat besar, pengimplementasian AI dalam identifikasi cacat perangkat lunak tidak bebas dari hambatan praktikal [5]. Diperlukan analisis komprehensif untuk memahami efektivitas setiap model, kontribusi variabel atau metrik tertentu, serta sejauh mana keandalan AI bila dibandingkan dengan metode pengujian konvensional ataupun analis manusia. Oleh karena itu, penelitian ini disusun untuk menjawab empat pertanyaan penelitian utama (*Research Questions* / RQ):

1. RQ1: Metode apa saja yang digunakan untuk memprediksi bug dan mendeteksi anomali dalam pengembangan perangkat lunak?
2. RQ2: Fitur, metrik, atau atribut data apa yang paling berpengaruh dan umum digunakan dalam membangun model prediksi bug?
3. RQ3: Apakah model AI menunjukkan peningkatan signifikan pada metrik evaluasi dibanding metode konvensional atau analis manusia?
4. RQ4: Apa saja tantangan dan hambatan utama dalam menyiapkan data, melakukan pelabelan, dan melatih model AI?

METODE PENELITIAN

Penelitian ini menggunakan pendekatan kualitatif dengan desain *Systematic Literature Review* (SLR) yang dilaksanakan secara terstruktur untuk mengidentifikasi, menganalisis, dan menyintesis literatur ilmiah terkini. Data yang digunakan dalam penelitian ini berupa 16 artikel ilmiah sekunder dari jurnal dan prosiding konferensi internasional bereputasi terbitan tahun 2021–2025 yang diperoleh melalui basis data ilmiah seperti IEEE Xplore, ScienceDirect, dan Google Scholar.

Pengumpulan data dilakukan dengan menetapkan kriteria inklusi (artikel ilmiah terkaji sejawat yang berfokus pada penerapan AI/ML untuk deteksi bug dan prediksi kesalahan) dan kriteria eksklusi (artikel populer, blog, esai non-ilmiah, serta publikasi tanpa data eksperimental). Metode analisis data menggunakan teknik sintesis kualitatif dan komparatif deskriptif yang dikelompokkan berdasarkan empat Pertanyaan Penelitian (*Research Questions* / RQ1–RQ4) untuk memetakan arsitektur model AI, ekstraksi fitur, efektivitas kinerja, serta hambatan teknis saat diimplementasikan.

HASIL

Berdasarkan hasil pengumpulan data dari 16 artikel ilmiah utama yang dianalisis dalam *Systematic Literature Review* (SLR) ini, rincian mengenai penulis, metode atau arsitektur AI yang

digunakan, fokus penelitian, serta temuan dari masing–masing penelitian dapat dilihat pada Tabel 1.

Tabel 1. Rincian Artikel

No	Penulis & Tahun	Metode / Arsitektur AI	Fokus Penelitian	Hasil & Temuan Utama
1	Hossain (2024) [1]	OracleGuru & LLM	*Test Oracle Quality & Bug Detection*	Mengurangi *coverage gap* hingga 34% dibanding metode tradisional.
2	Mostafa et al. (2025) [2]	GA + t-SNE + Random Forest / DL	*Feature Transformation & Commit Classification*	MeninBerhasil menaikkan kemampuan menemukan bug (*recall*) sampai 21%.
3	Yuan et al. (2025) [3]	*Time-Series Signal Analysis* + AI	*Hydrodynamics & Anomaly Detection*	Berhasil memetakan pola sinyal kompleks untuk deteksi anomali.
4	Vidal et al. (2025) [4]	*Static Analysis* (Securify, Slither) vs AI	*Blockchain Reliability & Elusive Faults*	Analisis statis konvensional hanya mencapai akurasi 6,4%.
5	Awad et al. (2022) [5]	*Hybrid Threshold* + Machine Learning	*Fault Prediction in WBAN Systems*	Berhasil memprediksi kerusakan sensor dan gangguan sistem dengan baik.
6	Keshavarz & Rodríguez (2024) [6]	JITGNN (*Graph Neural Network*)	*Just-In-Time Bug Prediction*	Efektif mengekstrak struktur AST untuk prediksi bug *real-time*.
7	Vargas et al. (2023) [7]	*Convolutional Kernels* + Event-Log	*ATM Fault Prediction*	Membuktikan efektivitas *kernel* konvolusi pada data log.
8	Long et al. (2025) [8]	MnasNet + LSTM + ILFA (Metaheuristik)	*Software Defect Prediction*	Mencapai akurasi hingga 93,0% pada dataset PROMISE.
9	Ardimento et al. (2025) [9]	LLM (LLaMA3-8B)	*Bug Fixing Time Prediction*	Menunjukkan efektivitas *zero-shot learning* pada laporan bug.
10	Wei (2025) [10]	*Category Balance* + Ensemble Voting	*Software Defect Preprocessing*	Meningkatkan stabilitas dan akurasi prediksi cacat.
11	Agrawalla & Reddy (2024) [11]	Heterogeneous Ensemble Selection	*Optimal Classifier Selection*	Meningkatkan nilai AUC secara signifikan di berbagai dataset.
12	Pandey & Kumar (2022) [12]	SMOTE + Machine Learning	*Imbalanced Data in Software Fault*	Mengatasi isu *class imbalance* pada prediksi kesalahan.
1 3	Medicharla et al. (2024) [13]	FeatBoost (*Iterative Boosting*) + RF/SVM	*Feature Selection Algorithm*	Mencapai akurasi tertinggi 90,8% menggunakan Random Forest.
14	Moradi (2025) [14]	MLANN + *Bayesian Optimization*	*Fault Prediction in Complex Systems*	Mencapai akurasi sangat tinggi sebesar 97,99%.
15	Hirsch & Hofer (2022) [15]	NLP (TF-IDF) + Ensemble Classifier	*Textual Bug Report Classification*	Mencapai skor F1 0,69, mengungguli analis manusia non-pakar (0,62).

16	Chmielowski et al. (2023) [16]	Explainable AI (XAI) + Decision Tree	*Bug Classification Transparency*	Memberikan akurasi tinggi setara model *black-box* dengan interpretabilitas.

Selain itu, ringkasan perbandingan kinerja antara metode lama, pemeriksaan manusia, dan teknologi AI disajikan pada Tabel 2.

Tabel 2**.** Perbandingan Performa Cara Lama vs Teknologi AI

Pendekatan / Model	Metrik Kinera Utama	Karakteristik Utama
Analisis Statis Konvensional (*Securify*)	Akurasi -6.4%	Berbasis aturan kaku, *false positive* tinggi dan sering saah tebak [4].
Analis Manusia (Non-Pakar)	F1-Score 0.62	Subjektif dan memakan waktu lama [15].
Ensemble ML + NLP (Hirsch & Hofer)	Skor F1 0.69	Mengungguli non-pakar pada analisis teks bug [15].
FeatBoost + Random Forest	Akurasi 90.8%	Seleksi fitur berbasis *boosting* iteratif [13].
MnasNet + LSTM + ILFA	Akurasi 93.0%	Ekstraksi fitur AST dan urutan sekuensial [8].
DL + Bayesian Optimization	Akurasi 97.99%	Optimasi hiperparameter pada data kompleks [14].

PEMBAHASAN

Berdasarkan empat pertanyaan penelitian (RQ1–RQ4), analisis dan interpretasi temuan dibahas sebagai berikut:

1. Metode Prediksi Bug dan Deteksi Anomali (RQ1)

Temuan penelitian menunjukkan bahwa alat penguji tradisional berbasis aturan kaku (seperti *Securify*) mulai ditinggalkan karena tingkat kesalahan tebak (*false positive*) yang tinggi. Penggunaan teknologi AI modern [4], seperti *Deep Learning* (LSTM), *Graph Neural Network* (JITGNN), dan *Large Language Models* (LLM), terbukti mampu membaca struktur logika kode serta riwayat *error* secara proaktif dan aur kode secara dinamis [6], [9]. Model gabungan seperti MnasNet dan LSTM yang dioptimasi algoritma metaheuristik berhasil mengenali alur program yang rumit dengan efisiensi tinggi [8].

2. Fitur dan Metrik Paling Berpengaruh (RQ2)

Keberhasilan AI dalam menemukan bug sangat dipengaruhi oleh kualitas data masukan. Pengubahan struktur kode program menjadi bentuk angka (*vector*) melalui *Abstract Syntax Tree* (AST) menjadi teknik yang paling efektif [8]. Pada data laporan *error* berupa teks, pemrosesan bahasa alami (NLP) seperti TF-IDF berhasil mengekstraksi kata kunci kritis (seperti *"memory leak"* atau *"crash"*) [15], [16]. Selain itu, metrik kualitas pengujian seperti *Host Checked Coverage* (HCC) terbukti lebih informatif ketimbang sekadar menghitung jumlah baris kode (LOC) [1].

3. Kinerja AI Dibanding Cara Konvensional dan Manusia (RQ3)

Hasil komparasi pada Tabel 2 memperlihatkan bahwa alat kaku lama hanya mencapai akurasi 6,4% [4]. AI terbukti melampaui kemampuan manusia non-pakar dalam mengelompokkan laporan masalah (skor F1 naik dari 0,62 menjadi 0,69) [15]. Model AI tingkat lanjut bahkan mampu menembus tingkat akurasi 90,8% hingga 97,99% [8], [13], [14]. Penggunaan *Explainable AI* (XAI) juga memberikan keunggulan baru, yaitu keputusan AI tidak lagi berupa "kotak hitam" melainkan memiliki alasan rasional yang bisa dipahami oleh tim pengembang [16].

4. Masalah dan Kendala Nyata di Lapangan (RQ4)

Meskipun mencatatkan performa tinggi di pengujian, penerapan AI pada lingkungan kerja nyata masih menghadapi hambatan:

a. Kualitas Data:
   Laporan *error* dari sistem pelacak masalah sering kali tidak rapi, tidak lengkap, atau formatnya acak-acakan [15].
b. Ketidakseimbangan Data (*Class Imbalance*):
   Bug berbahaya (seperti celah keamanan) jumlahnya sangat sedikit dalam data (2%–16%), sehingga AI kurang bahan untuk belajar [15], [16].
c. Kebutuhan Komputer Canggih dan Biaya:
   Melatih model *Deep Learning* membutuhkan daya komputasi tinggi dan hasilnya tidak selalu bisa langsung dipakai di proyek perangkat lunak lain (*cross-project generalizability*) [2], [8].

KESIMPULAN

Penerapan *Artificial Intelligence* (AI) dan *Machine Learning* (ML) terbukti mampu meningkatkan akurasi, efisiensi, dan keandalan proses deteksi bug serta prediksi kesalahan perangkat lunak secara signifikan dibandingkan metode tradisional maupun pemeriksaan manual manusia. Penggunaan arsitektur hibrida (MnasNet-LSTM), teknik pemilihan fitur (FeatBoost), dan pendekatan *Explainable AI* (XAI) menjadi kunci utama peningkatan kinerja ini..

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada rekan-rekan sejawat yang telah memberikan masukan dan saran konstruktif demi menyempurnakan karya ilmiah ini.

DAFTAR PUSTAKA

[1] S. B. Hossain, “Ensuring Critical Properties of Test Oracles for Effective Bug Detection,” Proceedings - International Conference on Software Engineering, pp. 176–180, 2024, doi: 10.1145/3639478.3639791.

[2] S. Mostafa, S. T. Cynthia, B. Roy, and D. Mondal, “Feature transformation for improved software bug detection and commit classification,” Journal of Systems and Software, vol. 219, no. July 2024, p. 112205, 2025, doi: 10.1016/j.jss.2024.112205.

[3] Y. Yuan et al., “Time-series signal analysis of sustainable process intensification: Characterization method development of gas-solid fluidized bed hydrodynamics towards AI enhanced algorithms,” Green Energy and Resources, vol. 3, no. 2, p. 100128, 2025, doi: 10.1016/j.gerr.2025.100128.

[4] F. R. Vidal, N. Ivaki, and N. Laranjeiro, “Analyzing the Impact of Elusive Faults on Blockchain Reliability,” Blockchain: Research and Applications, p. 100295, 2025, doi: 10.1016/j.bcra.2025.100295.

[5] M. Awad, F. Sallabi, K. Shuaib, and F. Naeem, “Artificial intelligence-based fault prediction framework for WBAN,” Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 9, pp. 7126–7137, 2022, doi: 10.1016/j.jksuci.2021.09.017.

[6] H. Keshavarz and G. Rodríguez-Pérez, “JITGNN: A deep graph neural network framework for Just-In-Time bug prediction,” Journal of Systems and Software, vol. 210, no. January 2023, p. 111984, 2024, doi: 10.1016/j.jss.2024.111984.

[7] V. M. Vargas, R. Rosati, C. Hervás-Martínez, A. Mancini, L. Romeo, and P. A. Gutiérrez, “A hybrid feature learning approach based on convolutional kernels for ATM fault prediction using

event-log data," Eng Appl Artif Intell, vol. 123, no. April, p. 106463, 2023, doi: 10.1016/j.engappai.2023.106463.

[8] W. Long, Z. Qixin, M. A. Zakharov, and S. Lee, "Optimizing fault prediction in software based on MnasNet/LSTM optimized by an improved lotus flower algorithm," Egyptian Informatics Journal, vol. 29, no. January, p. 100623, 2025, doi: 10.1016/j.eij.2025.100623.

[9] P. Ardimento, M. Capuzzimati, G. Casalino, D. Schicchi, and D. Taibi, "A novel LLM-based classifier for predicting bug-fixing time in Bug Tracking Systems," Journal of Systems and Software, vol. 230, no. July, p. 112569, 2025, doi: 10.1016/j.jss.2025.112569.

[10] X. Wei, "Research on Preprocessing Techniques for Software Defect Prediction Dataset Based on Hybrid Category Balance and Synthetic Sampling Algorithm," Procedia Comput Sci, vol. 262, pp. 840–848, 2025, doi: 10.1016/j.procs.2025.05.117.

[11] B. Agrawalla and B. R. Reddy, "Software Fault Prediction Using Optimal Classifier Selection: An Ensemble Approach," Procedia Comput Sci, vol. 235, pp. 2965–2974, 2024, doi: 10.1016/j.procs.2024.04.280.

[12] S. Pandey and K. Kumar, "Software Fault Prediction for Imbalanced Data: A Survey on Recent Developments," Procedia Comput Sci, vol. 218, pp. 1815–1824, 2022, doi: 10.1016/j.procs.2023.01.159.

[13] S. Medicharla, S. Kumar, P. Devarakonda, B. Agrawalla, and B. R. Reddy, "Software Fault Prediction Using FeatBoost Feature Selection Algorithm," Procedia Comput Sci, vol. 235, pp. 316–325, 2024, doi: 10.1016/j.procs.2024.04.032.

[14] E. Moradi, "Leveraging Bayesian optimization and multilayer artificial neural network (MLANN) for fault prediction in oil-immersed transformers," e-Prime - Advances in Electrical Engineering, Electronics and Energy, vol. 12, no. May, 2025, doi: 10.1016/j.prime.2025.101013.

[15] T. Hirsch and B. Hofer, "Using textual bug reports to predict the fault category of software bugs," Array, vol. 15, no. May, p. 100189, 2022, doi: 10.1016/j.array.2022.100189.

[16] L. Chmielowski, M. Kucharzak, and R. Burduk, "Application of Explainable Artificial Intelligence in Software Bug Classification," Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Srodowiska, vol. 13, no. 1, pp. 14–17, 2023, doi: 10.35784/iapgos.3396.