Multitek Indonesia: Jurnal Ilmiah
Volume: 20 No. 1 July 2026, Hal. 44 - 57
ISSN: 1907-6223 (print)
ISSN: 2579-3497 (Online)
http://journal.umpo.ac.id/index.php/multitek
doi.org/10.24269/mtkind.v20i1.13681

SEMANTIC AUTOMATED ASSESSMENT OF STUDENT FLOWCHARTS VIA GRAPH NEURAL NETWORKS AND SYMBOLIC EXECUTION

Usman Nurhasan[1], Dian Hanifudin Subhi [1], Anugrah Nur Rahmanto[1], Endah Septa Sintiya[1], Deddy Kusbianto Purwoko Aji[1]
[1]Jurusan Teknologi Informasi, Politeknik Negeri Malang
Jl. Soekarno Hatta No.9, Jatimulyo, Kecamatan Lowokwaru, Kota Malang, 65141, Indonesia
Corresponding Author's Email: **usmannurhasan@polinema.ac.id**

Received : 11 April 2026. Revised : 5 May 2026. Accepted : 24 May 2026.

Abstract

Automated evaluation of flowchart representations is essential for the facilitation of the acquisition of basic programming concepts. Nevertheless, traditional evaluation systems that rely exclusively on structural matching demonstrate some of their most fundamental limitations. The false negative misclassification rates of such systems are frequently high when students create visually distinct structures for algorithmic logic that are semantically equivalent. A hybrid assessment framework is introduced in this study to improve the reliability and efficacy of code evaluation in order to address this challenge. The model that has been proposed combines the probabilistic feature extraction capabilities of Graph Convolutional Networks (GCNs) with mathematical logic verification through symbolic execution of an SMT Solver. While the SMT Solver deterministically establishes functional equivalence, the GCN module adaptively manages graph topological variations. Use of a real-world dataset consisting of 3.600 flowcharts generated by novice students was implemented to assess the hybrid system's functionality. According to quantitative experimental results, the proposed framework obtained a peak F1 Score of 0.88, which is a substantial improvement over conventional Abstract Syntax Tree (AST) methods (F1 Score 0.75). Additionally, the 77.4% reduction in false negative rates was achieved by incorporating the SMT Solver in comparison to a pure GCN configuration. Finally, the semantic equivalence and structural divergence issues that arise during algorithm assessment are effectively resolved by this dual architectural integration. By implementing the proposed system, higher education institutions are equipped with a more dependable mechanism for reducing human error, thereby improving the impartiality, accuracy, and efficiency of the evaluation process.

Keyword: Automated Evaluation, Flowchart, Graph Convolutional Network, Symbolic Execution, Semantic Equivalence

INTRODUCTION

Computational thinking (CT) constitutes a vital skill set in modern educational systems, augmenting critical thinking and problem-solving abilities. Proficiency in CT establishes a vital algorithmic framework and acts as a stimulus for novice learners to accurately understand various programming languages [1], [2]. In educational settings, visual aids like flowcharts serve as essential cognitive tools in the early phases of programming instruction[3]. Flowcharts

facilitate a superior understanding of algorithmic reasoning compared to mere textual formats [4]. Recent research in software engineering indicates that structure-oriented assessments, such graphical representations, surpass text-based methods in capturing the essential elements of logic [5]. Consequently, higher education institutions continue to incorporate flowcharts as a core component of introductory programming courses.

In the age of digital educational platforms, institutions increasingly necessitate automated evaluation systems (auto graders) to facilitate quantifiable skill improvement in pupils[6]. The rapid increase in the student population makes manual assessment by teachers susceptible to inconsistency, overwhelming workload, and delayed feedback[7] [8] Furthermore, paradigm shifts in software development, shown by the concept of vibe coding, intensify these issues by shifting from deterministic instructions to probabilistic methods, resulting in significant variability in student solutions [9] [10]. Traditional visual auto graders have demonstrated insufficiency, as they overlook both structural and semantic information while depending on manually produced metrics [11].

The dependence on singular methodological approaches has resulted in a significant research gap between syntactic and semantic evaluation. Structural divergence occurs when students create functionally accurate solutions with visual topologies that differ from the reference answers. Pure Graph Neural Network (GNN) models may discern topological patterns via representation learning. Nevertheless, they do not substantiate mathematical functional equivalence (semantic equivalent). Recent data corroborates this restriction, indicating that single representation frameworks markedly elevate false positive and false negative rates in code assessment [12]. Precise analysis requires tree or graph-based extraction to capture the semantic depth of programs, as demonstrated in contemporary code clone detection techniques[13].

This paper presents a hybrid paradigm to address these constraints in automated flowchart assessment. The methodology corresponds with cutting-edge research trends that incorporate structure-guided and semantics-enhanced designs [14]. The proposed model integrates the probabilistic features of Graph Neural Networks (GNNs) with deterministic formal verification through Satisfiability Modulo Theories (SMT) solvers to ascertain semantic equivalence. This study yields three principal outcomes: (1) the creation of a novel hybrid assessment framework; (2) the availability of an anonymized empirical flowchart dataset for subsequent research; and (3) empirical evidence substantiating the efficacy of formal validation in AI-based auto-graders.

This research empirically establishes three leading questions to assess the suggested framework. Initially, it examines the degree to which hybrid models surpass traditional structural matching techniques in evaluative precision. The examination focuses on the specific contribution of integrating symbolic execution to the reduction of false negative rates resulting from structural divergence. The strength of the statistical correlation between the automated scores produced by the proposed system and the expert human evaluation metrics, which serve as the ground truth, is validated.

RESEARCH METHOD

This study employed a quantitative experimental research design to evaluate the effectiveness of a hybrid intelligent assessment framework for automated evaluation of student-generated flowchart algorithms. As illustrated in Figure 1, the proposed methodology consists of four major stages: (1) Data Collection and Preprocessing, (2) Hybrid System Architecture, (3) Experimental Framework, and (4) Assessment Metrics. The framework integrates graph-based structural learning and formal semantic verification to achieve robust, reliable, and semantically consistent automated assessment results.

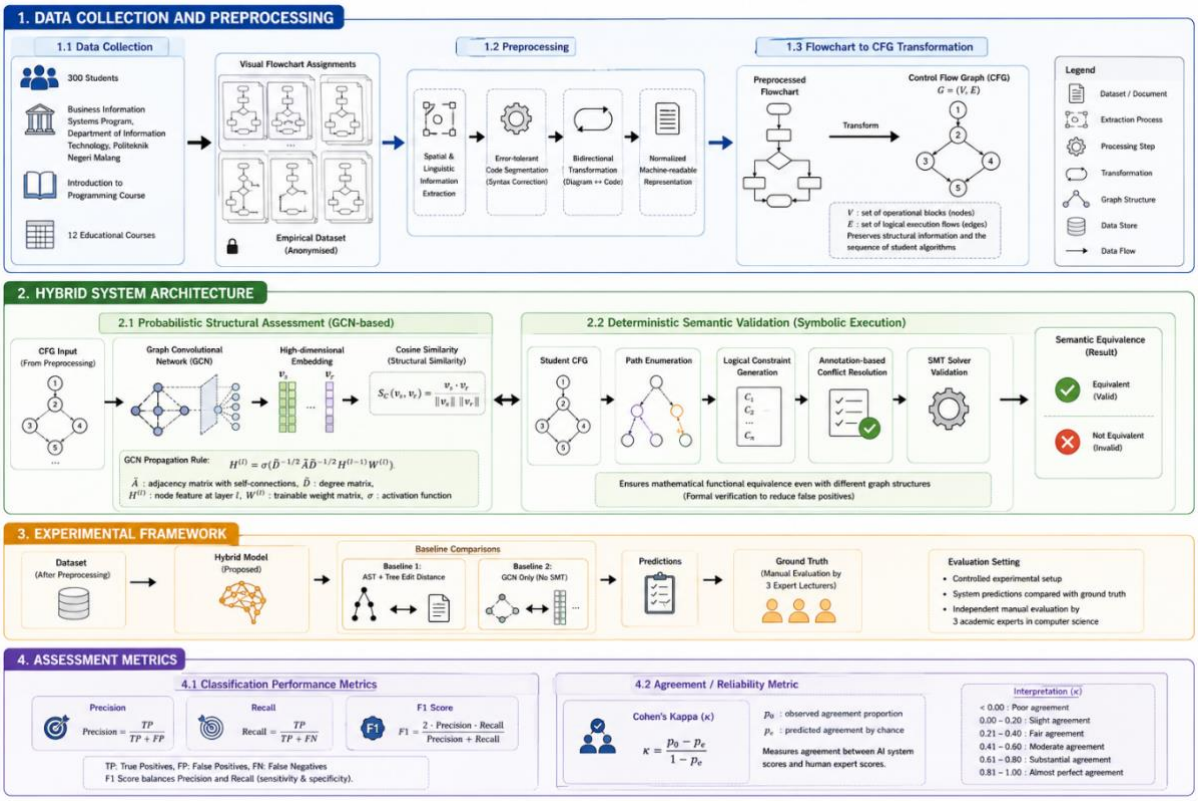

Figure 1. Proposed hybrid intelligent assessment framework for automated flowchart-based algorithm evaluation.

Data Collection and Pre-processing

This research utilized an empirical dataset of visual flowchart representations. The information was gathered from assignments submitted by 300 students participating in the Introduction to Programming course in the Business Information Systems program at the Department of Information Technology, Politeknik Negeri Malang. The dataset included algorithmic answers obtained from 12 educational courses. All data were meticulously anonymised to adhere to academic research ethical norms. In the preprocessing stage, spatial and linguistic information was extracted from raw flowcharts to create machine-processable mathematical data structures. This method corresponds with the bidirectional transformation paradigm between diagrammatic and code representations [15]. Due to the significant heterogeneity in student replies, error-tolerant code segmentation techniques were employed to rectify syntactic errors in visually flawed flowcharts [16]. Thereafter, each flowchart was transformed into a Control Flow Graph (CFG), officially defined as a directed graph $G = (V, E)$.

The set V corresponds to operational blocks in the flowchart, while the set E denotes logical execution flows as edges. This extraction approach successfully maintained both structural information and the sequence of student algorithms [17].

Hybrid System Architecture

Recent literature has shown that dependence on singular representations be they structural or semantic results in elevated false positive rates in code assessment [12]. This paper provides a hybrid framework based on this theoretical foundation. The system functions via two main subsystems: a probabilistic structural assessment module and a deterministic semantic validation module. The initial subsystem employs a Graph Convolutional Network (GCN) to derive latent representations from Control Flow Graphs (CFGs). The GCN architecture executes convolutional operations on graph structures, consolidating features from each node and its adjacent nodes in accordance with the specified propagation rule [13][18].

$$H^{(l-1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \quad (1)$$

where $\widetilde{A}$ denotes the adjacency matrix with self-connections, $\widetilde{D}$ represents the degree matrix, $H^{(l)}$is the feature representation at layer l, $W^{(l)}$ is the trainable weight matrix, and σ is a nonlinear activation function. This model produces high-dimensional embedding vectors v. The system then computes cosine similarity between the student vector $v_s$and the reference answer vector v_r to quantitatively measure structural similarity:

$$S_C(v_s, v_r) = \frac{v_s . v_r}{|v_s||v_r|} \quad (2)$$

The second subsystem tackles structural divergence via symbolic execution. Graph-based methodologies have demonstrated their significance in encapsulating the semantic depth of programs, particularly in code clone detection, where architecturally disparate programs display semantic equivalence [13]. Every potential execution path in student control flow graphs is converted into collections of logical constraint equations. Semantic conflict resolution is implemented by annotation-based rules to adeptly manage logical circumstances [19]. An SMT solver subsequently validates these logical restrictions to provide mathematical functional equivalence between student algorithms and reference algorithms [20]. The incorporation of formal verification guarantees that solutions with varying graph structures yet identical mathematical outputs are accurately assessed as valid responses.

Experimental Framework

A controlled experimental setup was established to assess the efficacy of the suggested hybrid model. An empirical evaluation was performed by contrasting system predictions with ground truth values. Ground truth was determined via independent manual evaluations conducted by three academic specialists in computer science. To illustrate the importance of the model's contribution, two baseline techniques were established for comparison. The first baseline utilized a traditional method employing the tree edit distance algorithm on

Abstract Syntax Trees (ASTs) to quantify code edit distance [21][22]. Previous literature has condemned the utilization of pure Abstract Syntax Trees (ASTs) for their informational redundancy and representational inefficiency [23]. This baseline was utilized to confirm the comparative advantage of the proposed CFG-based representation. The second baseline employed a purely artificial intelligence methodology, depending exclusively on a GCN classification model without activating the symbolic execution module [18].

Assessment Metrics

Two classifications of evaluation indicators were utilized to examine the efficacy of the automated assessment system. The initial category included typical classification measures for quantifying model accuracy, primarily focusing on the F1 Score, which equilibrates Precision and Recall:

$$Precision = \frac{TP}{TP+FP},\ Recall = \frac{TP}{TP+FN},\ F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (3)$$

TP represents true positives, FP signifies false positives, while FN indicates false negatives. The $F1$Score was chosen as the primary metric because it offers a balanced assessment of both sensitivity and specificity in classification tasks. The second category concentrated on evaluating system reliability with Cohen's Kappa (κ). This statistical indicator quantifies the level of concordance between scores produced by the artificial intelligence system and those provided by human experts. The formulation is articulated as:

$$\kappa = \frac{p_0 - p_1}{1 - p_e} \quad (4)$$

where p_0 denotes the observed agreement proportion and p_e represents the predicted agreement by chance. Cohen's Kappa was chosen to guarantee that the evaluation framework attained quantitative correctness while also aligning with expert judgment, thus confirming the reliability of the automated assessment method.

RESULTS AND DISCUSSION

Comparative Analysis of Model Performance (Addressing RQ1)

The efficacy of the suggested hybrid framework was empirically assessed in comparison to two baseline techniques. A comparative analysis was performed to assess the correctness and reliability of each system architecture. Table 1 provides a detailed description of evaluation metrics derived from testing on the student flowchart dataset.

Table 1. Comparative Performance of Automated Assessment Models

Architecture Model	Accuracy	Precision	Recall	F1-Score
AST + *Tree Edit Distance* (Baseline 1)	0.76	0.77	0.73	0.75
GCN Murni (Baseline 2)	0.83	0.84	0.80	0.82
Hybrid (GCN + SMT Solver)	0.89	0.90	0.87	0.88

As shown in Table 1, the hybrid architecture obtained higher values across all evaluation

metrics compared to both baselines. These results suggest a consistent trend of improved performance, although statistical significance was not formally tested. Figure 2 illustrates the comparative F1 Scores of the three system topologies, underscoring the importance of collective performance enhancement.

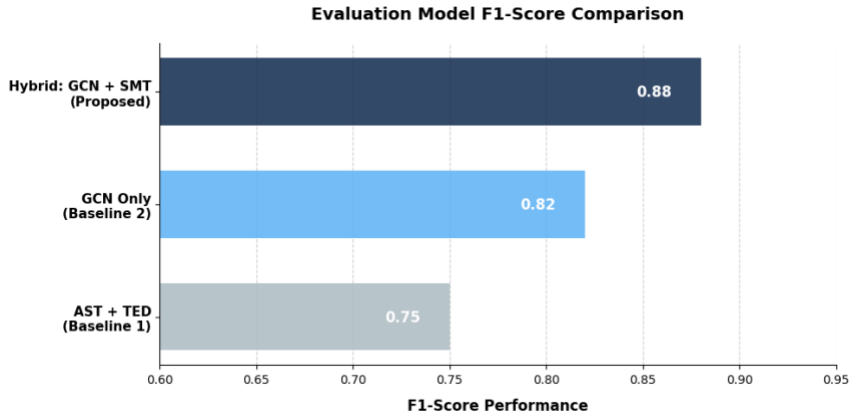

Figure 2. F1-score comparison of the proposed hybrid system against baseline models

A comprehensive examination of Table 1 and Figure 1 verifies that the hybrid system's superiority arises from its capacity to address structural divergence and semantic equivalence challenges. The traditional AST method in Baseline 1 had significant deficiencies in evaluative flexibility, achieving the lowest *F1* Score of 0.75. The Tree Edit Distance technique exhibited significant sensitivity to differences in graph topology [21]. Consequently, AST based systems erroneously penalized student solutions when algorithmic structures varied visually yet remained semantically equal. This structural stiffness supports previous studies on redundancy and inefficiency in pure tree representations [23], [24] which directly diminishes evaluation scores and elevates false negative rates.

Conversely, the pure GCN model in Baseline 2 demonstrated a significant enhancement in performance, with an *F1* Score of 0.82. This artificial intelligence model demonstrated probabilistic adaptability in representing structural differences in graphs of student solutions. Nonetheless, the pure GCN design exhibited significant shortcomings in semantic anomaly detection. The system consistently accepted logically flawed solutions as it identified solely visual representation patterns without executing mathematical logic checking. This empirical result corroborates theoretical assertions that dependence solely on latent structural representations induces increased false positive rates in algorithmic assessment [12]. The hybrid framework successfully mitigated the deficiencies of both baselines via thorough integration, attaining optimal performance (*F1* Score 0.88). The suggested approach offers practical benefits by utilizing GCN capabilities to probabilistically address changes in graph structure, while concurrently employing the SMT Solver to ensure semantic validity an key concept for recognizing program equivalence [25], [26]. This investigation further confirms that graph-based visual evaluation methods provide adaptive benefits for novice programming tasks in comparison to text-based representations.

Prospective Research Avenues: This research presents multiple strategic opportunities

for enhancing automated evaluation systems. For instance, ensemble learning frameworks can be employed to improve prediction stability. In addition, symbolic reasoning tools such as SMT solvers may be explored to handle semantic conflicts more [27][28][29]. The suggested method has the potential for incorporation into collaborative code editors powered by large language models (LLMs), facilitating interactive programming instruction [30]. Third, subsequent research ought to implement Explainable AI (XAI) frameworks to convert mathematical graph topologies into detailed textual feedback for students [31]. The utilization of generative artificial intelligence approaches to facilitate bidirectional transformations between diagrams and code will enhance system resilience during preprocessing [15]. These developments will collectively convert static evaluation tools into comprehensive intelligent teaching systems.

Effects of Semantic Verification (Responding to RQ2)

An ablation study was performed to assess the distinct impact of the symbolic execution module on overall system performance. The evaluation utilized a comprehensive dataset of 3.600 flowcharts gathered from 300 students over 12 training modules. Ground truth values were independently determined by three teaching assistants via manual evaluation. Table 2 provides a detailed comparison of performance characteristics between single component configurations and the hybrid design.

Table 2. Ablation Study Results for System Configurations

System Configuration	Accuracy	Precision	Recall	F1-Score	False-Negative Cases
GCN Only	0.83	0.84	0.80	0.82	420
SMT Solver Only	0.78	0.81	0.76	0.78	510
Hybrid (GCN + SMT Solver)	0.89	0.90	0.87	0.88	95

Table 2 demonstrates that the hybrid arrangement consistently surpassed single component structures in all evaluation metrics. The paramount discovery pertains to the false negative (*FN*) metric. The unadulterated GCN setup yielded 420 false negative cases, indicating the system's inability to identify roughly 11.7% of accurate student solutions. The activation of the semantic verification module in the hybrid architecture decreased the mistake count to 95 cases, reflecting a 77.4% reduction in misclassification. Figure 3 illustrates the association between the significant reduction in false negatives and enhanced model performance.

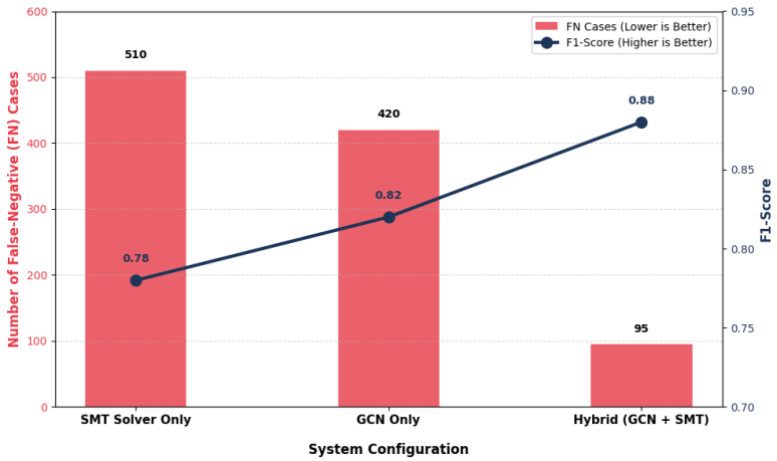

Figure 3. Ablation study of F1-score and False-Negative (FN) cases across different system configurations

The SMT Solver module's impact is plainly demonstrated by the visualization in Figure 2. As the system transitions from pure GCN to the hybrid framework, the trend line indicates a precipitous decline in false negative cases, which is consistent with an increase in F1 Score. This reduction in FN mathematically induced an increase in Recall from 0.80 to 0.87 (see Table 2). The SMT Solver is not merely a supplementary component; it is a critical "*savior component*" that prevents the unjust penalization of correct pupil solutions, as these consistent quantitative findings confirm.

The hybrid framework's ability to suppress false negative metrics is illustrated by a representative case of structural divergence that is emphasized through qualitative analysis. In a particular test scenario, the reference solution employed a FOR structure to implement a loop, with counter initialization embedded within the iteration block. In contrast, a student constructed the flowchart by introducing a fictitious variable and positioning counter initialization outside the main block, utilizing a WHILE structure. During the initial stages of programming education, it is common for neophyte students to decompose intricate algorithmic instructions into more explicit procedural steps, resulting in this type of divergence. The computational semantics of both flowcharts were identical for iterations from 1 to N, despite the visual topological distinctions. The student solution was not accurately evaluated by the unadulterated GCN subsystem. The cosine similarity score was 0.62 as a result of the spatial similarity measurement. The Adjacency Matrix Extraction during Graph Convolution was substantially distorted by the addition of external initialization nodes and dummy variables, which significantly altered the Control Flow Graph (CFG) architecture. Consequently, the GCN subsystem misclassified the student solution as inappropriate by reducing similarity below the heuristic acceptance threshold (0.70). This prediction failure directly validates the theoretical limitations in prior literature, which underscored the susceptibility of singular topological representations to misclassification in software engineering evaluation [12], [18].

To resolve this evaluation conflict, the SMT Solver subsystem acted as a deterministic verifier. Symbolic execution traced all possible CFG paths to extract path constraints. The

student graph was translated into the logical formula $\phi_s = \{i = 0; \text{while}(i < N)\{i++\}\}$, while the reference graph was expressed as $\phi_r = \{\text{for}(i = 0; i < N; i++)\}$. The Z3 Theorem Prover comprehensively analyzed both logical constraints. During inference, Z3 identified that the presence of the dummy variable in the student algorithm had no computational impact on the final state of output variables. This computational flexibility demonstrated the system's capability to resolve semantic conflicts effectively [32]. Symbolic computation ultimately proved absolute equivalence, $\phi_s \equiv \phi_r$.

According to the program equivalence proof [13], the hybrid decision subsystem immediately superseded the initial GCN classification, reclassifying the student answer as accurate. This empirical argument offers definitive proof that the hybrid design demonstrates remarkable resilience to various programming styles (vibe coding). The GCN subsystem probabilistically manages visual variability, whilst the SMT Solver subsystem ensures logical correctness. This architectural integration efficiently bridges the divide between syntactic diversity and semantic accuracy, allowing higher education institutions to provide automated assessment systems that are more equitable, precise, and dependable.

Human–AI Concordance (Addressing RQ3)

The viability of automated evaluation systems is evaluated not just by technical accuracy measures but also by their capacity to continuously emulate human judgment. This study assessed the concordance between the ground truth ratings assigned by teaching assistants (human evaluators) and the predictions produced by the Hybrid system (GCN + SMT Solver). The statistical study utilized two main metrics: Cohen's Kappa (κ) for assessing categorical agreement (Correct/Incorrect) and Pearson's correlation coefficient (r) for evaluating score distribution. Table 3 encapsulates the outcomes of the statistical tests.

Table 3. Statistical Agreement Tests: Human Raters vs. Hybrid AI System

Evaluation Metric	Statistical Value	Significance Level	Agreement Interpretation
Cohen's Kappa (κ)	0.86	$p < 0.001$	Almost Perfect Agreement
Pearson's Correlation (r)	0.89	$p < 0.001$	Very Strong Positive Correlation
Absolute Agreement Accuracy	89.2%	–	Very High Alignment

Table 3 demonstrates that the suggested Hybrid system attained a Cohen's Kappa score of $\kappa=0.86$. As per the statistical interpretation criteria established by Landis and Koch (1977), this number indicates "almost perfect agreement." Additionally, Pearson's correlation analysis demonstrated a robust positive connection ($r=0.89$, $p<0.001$) between human and machine ratings. Figure 4 illustrates the alignment level shown by the confusion matrix, affirming that the AI system did not compromise the grading criteria set by the teaching team.

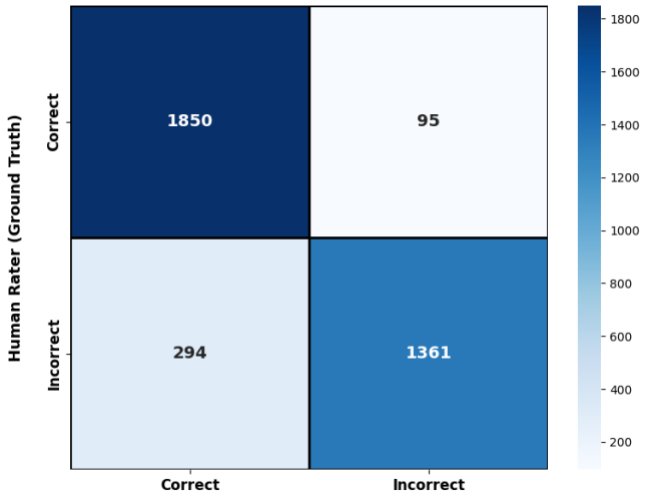

Figure 4. Confusion Matrix of Agreement Between Human Raters And The Hybrid Model

The significant concordance between human and computer assessments has profound implications for operational efficiency in higher education. A comparison of time efficiency was made to emphasize this influence. Teaching assistants required an average of 4 minutes for a comprehensive assessment of each flowchart. The manual grading of the dataset of 3.600 flowcharts required an estimated 240 man-hours. The Hybrid architecture executed graph preprocessing, GCN feature extraction, and Z3 SMT Solver verification in an average of 1.2 seconds per flowchart. The AI system completed the full dataset in just 1.5 hours. This comparison indicates a substantial workload decrease over 99%. The significant time savings illustrate the practical viability of applying the suggested framework in actual educational settings. By diminishing the cognitive burden linked to repetitious administrative duties, institutions can redirect faculty and assistant resources to more essential educational initiatives, such as individualized support for students facing challenges with algorithmic reasoning [12], [33].

In addition to speed, qualitative assessments during testing indicated significant reliability issues. In several anomalous instances, the AI model effectively identified possible concealed infinite loops that were missed by teaching assistants. This discovery underscores the vulnerability of human assessors to cognitive exhaustion when evaluating several assignments featuring repeated visual structures [34], [35]. In contrast, the SMT Solver module reliably performed mathematical logic verification without endurance constraints. The empirical findings validate that the suggested Hybrid system is academically reliable, serving both as an effective assessment tool and as an auxiliary verification layer to reduce human error in code evaluation.

Validity Threats

Despite the hybrid framework exhibiting robust empirical performance, certain risks to validity were found that must be acknowledged for an impartial evaluation of the results. This essential discourse has three principal dimensions: construct validity, internal validity, and external validity. Regarding construct validity, which assesses the degree to which assessment

metrics reflect the original notion, the primary hazard is in the definition of algorithmic correctness. This study utilized an SMT Solver to establish semantic equivalence between student solutions and reference answers, concentrating solely on functional correctness while neglecting non-functional efficiency. The suggested system continued to classify student loop logic as similar despite the approach demonstrating suboptimal temporal complexity. This constraint may skew assessment results in advanced instructional modules where code optimization is emphasized [23], [36].

Concerning internal validity, focus was placed on possible biases or confounding variables affecting experimental outcomes. The initial threat in this realm stemmed from the ground truth creation process. Despite the calibration of AI evaluation with human judgment, teaching assistants continued to be susceptible to cognitive fatigue and subjective bias [37]. To alleviate this risk, cross-validation processes were established, incorporating three independent teaching assistants utilizing majority voting mechanisms. In addition to human concerns, internal validity was further compromised by the technological constraints of mathematical proof engines. The Z3 Theorem Prover is prone to state space inflation when evaluating graphs with extensively nested branches [38]. While computational failures were absent in the current dataset, the potential for processing timeouts remains for more intricate structures. Ultimately, external validity underscores the constraints of extrapolating results outside the present experimental parameters. The collection comprised 3.600 flowcharts sourced from 12 beginning programming programs. These lessons concentrated extensively on imperative and procedural paradigms for beginner learners. Thus, the efficacy of the hybrid system cannot currently be extrapolated to graph representations of more dynamic paradigms, such object-oriented programming or concurrent systems [39]. Structural transformations inherent in these advanced paradigms would require significantly more complex path constraint formulations for SMT Solver verification.

4. Conclusion

This study introduced and assessed a hybrid automated evaluation framework that combines the probabilistic adaptability of Graph Convolutional Networks (GCNs) with the mathematical precision of symbolic execution using an SMT Solver. Empirical testing indicated that this architectural integration helped address anomalies in algorithmic assessment. The suggested approach achieved an F1 Score of 0.88, showing higher values than traditional baseline techniques (RQ1). Furthermore, ablation studies suggested that the inclusion of the semantic verification module reduced false negative rates by 77.4% compared to the pure GCN model. These findings indicate that the system can accommodate structural differences while maintaining functional equivalence across various student solutions (RQ2). In addition to technological benefits, operational practicality was supported by a high concordance with human evaluators, reflected in a Cohen's Kappa value of 0.86. The AI approach also reduced assessment time from 240 hours of human grading to under 1.5 hours of computational processing. This quantitative evidence highlights the potential feasibility of the framework in alleviating teacher workload, reducing human error due to cognitive fatigue, and supporting more objective assessment procedures in higher education (RQ3). Future

research will aim to enhance system capabilities by integrating Explainable AI (XAI) frameworks. A key direction involves converting mathematical proofs produced by the SMT Solver into detailed textual feedback. This development is expected to transform the current static assessment tool into an intelligent tutoring system, enabling interactive programming instruction for beginner learners.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge UPA P3M Politeknik Negeri Malang for funding this research.

REFERENCES

[1] D. Tagare, "WIP : Do Your Students Learn Multiple Programming Languages ? How Do Computational Thinking Skills Help Them ?," *2025 IEEE Frontiers in Education Conference (FIE)*, pp. 1–5, doi: 10.1109/FIE63693.2025.11328630.

[2] R. Fabián, Z. Muñoz, J. Ariel, H. Alegría, G. Robles, and S. Member, "Assessment of Computational Thinking Skills : A Systematic Review of the Literature," vol. 18, no. 4, pp. 319–330, 2023, doi: 10.1109/RITA.2023.3323762.

[3] J.-H. Zhang, B. Meng, L.-C. Zou, Y. Zhu, and G.-J. Hwang, "Progressive flowchart development scaffolding to improve university students' computational thinking and programming self-efficacy," *Interactive Learning Environments*, vol. 31, no. 6, pp. 3792–3809, Aug. 2023, doi: 10.1080/10494820.2021.1943687.

[4] M. Zhang *et al.*, "PFDial : A Structured Dialogue Instruction Fine-tuning Method Based on UML Flowcharts," pp. 2626–2649, 2025.

[5] B. E. N. Wang, Y. Tong, S. Ji, and B. Data, "A Review of Learning-based Smart Contract Vulnerability Detection : A Perspective on Code Representation," *ACM Transactions on Software Engineering and Methodology*, vol. 35, no. 6, 2026, doi: 10.1145/3750042.

[6] S. Xiong, Y. Li, W. Luo, and H. Kong, "Design and implementation of online programming skill improvement system based on collaborative filtering algorithm," *2025 IEEE 8th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, vol. 8, pp. 143–148, 2025, doi: 10.1109/IAEAC65194.2025.11166572.

[7] D. Deepshikha, "A comprehensive review of AI-powered grading and tailored feedback in universities," 2025.

[8] I. M. Technologies and M. A. Models, "Interactive Mobile Technologies," vol. 19, no. 14, pp. 57–69, 2025.

[9] C. Zhao and H. Education, "AI-assisted Assessment in Higher Education : A Systematic Review," pp. 39–58.

[10] M. Landers, "Adapting to the Unsanctioned Use of AI-Supported Technologies in Student Assessments," *Higher Education for the Future*, 2024, doi: 10.1177/23476311241300608.

[11] J. Lu, B. K. Balasubramanian, M. Joy, and Q. Xu, "Survey and Analysis for the Challenges in Computer Science," vol. 58, no. 1, 2025.

[12] A. Abdu, Z. Zhai, H. A. Abdo, and R. Algabri, "Software Defect Prediction Based on Deep Representation Learning of Source Code From Contextual Syntax and Semantic Graph," *IEEE Transactions on Reliability*, vol. 73, no. 2, pp. 820–834, 2024, doi: 10.1109/TR.2024.3354965.

[13] Y. Wu, S. Feng, W. Suo, D. Zou, and H. Jin, "Goner : Building Tree-Based N-Gram-Like Model for Semantic Code Clone Detection," *IEEE Transactions on Reliability*, vol. 73, no. 2, pp. 1310–1324, 2024, doi: 10.1109/TR.2023.3312294.

[14] D. Yang, "Structure-Guided and Semantics-Enhanced Collaborative Code Generation," *2025 8th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pp. 556–562, 2025, doi: 10.1109/AEMCSE65292.2025.11042693.

[15] M. Trifan, B. Ionescu, and D. Ionescu, "Generative AI for Diagrams as Code and Code as Diagrams," *2025 IEEE 19th International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pp. 1–6, 2025, doi: 10.1109/SACI66288.2025.11030105.

[16] J. Jiang *et al.*, "Error-Tolerant Code Segmentation for Supporting Semantic Conflict Prevention in Real-Time Collaborative Programming," *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 4588–4593, 2024, doi: 10.1109/SMC54092.2024.10832108.

[17] S. Qiu, H. Huang, W. Jiang, F. Zhang, and W. Zhou, "Defect Prediction via Tree-Based Encoding with Hybrid Granularity for Software Sustainability," vol. 9, no. 3, pp. 249–260, 2024.

[18] Y. Wu, "Research on prediction algorithm of college students' academic performance based on Bert-GCN multi-modal data fusion," *Systems and Soft Computing*, vol. 7, p. 200327, 2025, doi: https://doi.org/10.1016/j.sasc.2025.200327.

[19] M. Wang, S. Member, B. Fang, and S. Member, "Annotation-based Semantic Conflict Prevention in Real-Time Collaborative Programming : Approach , Techniques , Prototype , and User Study," *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1755–1760, 2024, doi: 10.1109/SMC54092.2024.10831572.

[20] C. Dechsupa, T. Panboonyuen, W. Vatanawood, P. Padungweang, and C. So-in, "Toward AI-Augmented Formal Verification : A Preliminary Investigation of ENGRU and Its Challenges," *IEEE Access*, vol. 13, no. April, pp. 84357–84379, 2025, doi: 10.1109/ACCESS.2025.3568194.

[21] W. Hashimoto, "Basic investigation of code edit distance measurement by CodeBERT," *2023 15th International Congress on Advanced Applied Informatics Winter (IIAI-AAI-Winter)*, pp. 13–18, 2023, doi: 10.1109/IIAI-AAI-Winter61682.2023.00012.

[22] C. Vyshnavi, "Enhancing Code Insights through Semantic Change Impact Evaluation," *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, pp. 1–7, 2024, doi: 10.1109/ICCCNT61001.2024.10725350.

[23] T. Yu, L. Yuan, L. Lin, and H. He, "A Multiple Representation Transformer with Optimized Abstract Syntax Tree for Efficient Code Clone Detection," *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pp. 281–293, 2025, doi: 10.1109/ICSE55347.2025.00050.

[24] Y. Wang and X. Wang, "PyReach : A Multi-Agent Framework for Vulnerability Reachability Analysis in Python," *2025 32nd Asia-Pacific Software Engineering Conference (APSEC)*, pp. 173–183, 2025, doi: 10.1109/APSEC66846.2025.00027.

[25] K. Kishor, P. Vishwakarma, L. Sengar, V. Kumar, and V. Gupta, "Development of Grader Provider System Using Deep Learning," *Procedia Computer Science*, vol. 259, pp. 172–181, 2025, doi: https://doi.org/10.1016/j.procs.2025.03.318.

[26] L. Sun, H. Du, and T. Hou, "FR-DETR: End-to-End Flowchart Recognition with Precision and Robustness," *IEEE Access*, vol. PP, p. 1, 2022, doi: 10.1109/access.2022.3183068.

[27] P. Prediction, "applied sciences Multi-Output Based Hybrid Integrated Models for Student Performance Prediction," 2023.

[28] A. A. Alsulami, A. S. A. M. AL-Ghamdi, and M. Ragab, "Enhancement of E-Learning Student's Performance Based on Ensemble Techniques," *Electronics (Switzerland)*, vol. 12, no. 6, pp. 1–18, 2023, doi: 10.3390/electronics12061508.

[29] C. Xiang, Y. Wang, Q. Zhou, and Z. Yu, "Graph semantic similarity-based automatic assessment for programming exercises," *Scientific Reports*, vol. 14, 2024, doi: 10.1038/s41598-024-61219-8.

[30] A. Aher, R. S. Waghode, M. Jamsutkar, A. J. Patil, U. Padelkar, and D. S. Gat, "C3 – Code Commit Collab - A collaborative Code Editor using Repository Level LLM," *2025 3rd International Conference on Communication, Security, and Artificial Intelligence (ICCSAI)*, vol. 3, pp. 2071–2076, 2025, doi: 10.1109/ICCSAI64074.2025.11064134.

[31] S. Kamal, S. F. Nimmy, and G. S. Member, "Interpretable Code Summarization," *IEEE Transactions on Reliability*, vol. 74, no. 1, pp. 2280–2289, 2025, doi: 10.1109/TR.2024.3392876.

[32] C. Wang, B. Chen, G. Li, H. Wang, and S. Member, "Federated Learning Framework," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 10, pp. 9959–9972, 2023, doi: 10.1109/TKDE.2023.3250264.

[33] E. J. Gutiérrez Beltrán and J. C. Martínez Arias, "Mi Superpoder es la Programación: A tool for teaching programming to children and youth," *Science of Computer Programming*, vol. 240, p. 103198, 2025, doi: https://doi.org/10.1016/j.scico.2024.103198.

[34] H. Wan, H. Luo, M. Li, and X. Luo, "Automated Program Repair for Introductory Programming Assignments," *IEEE Transactions on Learning Technologies*, vol. 17, pp. 1705–1720, 2024, doi: 10.1109/TLT.2024.3403710.

[35] J. W. Browning, J. Bustard, and N. Anderson, "ASTRO : A Semi-Automated Grading and Feedback System for Programming Assignments," *2025 IEEE Frontiers in Education Conference (FIE)*, pp. 1–8, doi: 10.1109/FIE63693.2025.11328280.

[36] C. Qiu, J. Liu, X. Xiao, and Y. Xiao, "OpCodeBERT : A Method for Python Code Representation Learning by BERT With Opcode," *IEEE Transactions on Software Engineering*, vol. 51, no. 11, pp. 3103–3116, 2025, doi: 10.1109/TSE.2025.3610244.

[37] G. Adorni and A. Piatti, "Designing the virtual CAT: A digital tool for algorithmic thinking assessment in compulsory education," *International Journal of Child-Computer Interaction*, vol. 45, p. 100760, 2025, doi: https://doi.org/10.1016/j.ijcci.2025.100760.

[38] X. Li *et al.*, "GAI Versus Teacher Scoring : Which is Better for Assessing Student Performance ?," vol. 18, pp. 569–580, 2025.

[39] C. Hillis *et al.*, "AI ethics education: A scoping review of pedagogy, curriculum, and assessment," *Information Processing & Management*, vol. 63, no. 6, p. 104767, 2026, doi: https://doi.org/10.1016/j.ipm.2026.104767.