# Automated Cognitive Assessment in Programming Education Using Graph Neural Networks and CodeBERT: A Multi-Dimensional Approach

**Nithianantham Arunprakash[1], Sangeetha Arunprakash[2]**

**[1]Department of Mechanical Engineering, Faculty of Engineering, University of Jaffna, Kilinochchi, Sri Lanka**

**[2]Department of Information Technology, Faculty of Computing, SLIIT Northern Uni, Jaffna, Sri Lanka**

**DOI: https://doi.org/10.47772/IJRISS.2026.1026EDU0468**

**Received: 15 July 2026; Accepted: 20 July 2026; Published: 30 July 2026**

## ABSTRACT

Automated programming assessment has traditionally emphasized functional correctness, test-case outcomes, and efficiency, while giving less attention to the cognitive depth and structural quality demonstrated in a learner's code. This study presents AI-SOLO, a multi-dimensional framework that maps programming submissions to the five levels of the Structure of Observed Learning Outcomes (SOLO) Taxonomy by integrating semantic, structural, and behavioural evidence. Semantic representations are generated with CodeBERT, structural evidence is derived from Abstract Syntax Trees and Program Dependency Graphs processed by graph-based models, and behavioural evidence is obtained from platform interaction metadata. The study involved 120 undergraduate students and 1,824 Python submissions collected across introductory and intermediate programming activities. The archived aggregate evaluation reports an overall classification accuracy of 91.4%, with the strongest performance at lower SOLO levels and a reported 83.4% classification rate for Extended Abstract responses. A separately retained 400-prediction confusion-matrix output yields 90.25% overall accuracy and 71.7% recall for Extended Abstract; this discrepancy is reported transparently because the two stored summaries cannot be assumed to represent the same evaluation slice. The results nevertheless show a clear pattern of adjacent-level confusion, particularly between Relational and Extended Abstract responses. AI-SOLO's main contribution is therefore not merely higher predictive performance, but a pedagogically grounded architecture that combines what the code means, how it is structured, and how the learner arrived at it. The evidence is promising, but external validation, formal inter-rater reliability reporting, paired statistical testing, modality ablation, and cross-language evaluation remain necessary before strong claims of generalisability can be made.

**Keywords:** Automated Assessment, Cognitive Assessment, Solo Taxonomy, Graph Neural Network, Programming Education, Formative Feedback, CodeBERT.

## INTRODUCTION

Programming courses routinely require instructors to evaluate large numbers of code submissions while still providing feedback quickly enough to influence learning. Automated grading systems have therefore become central to scalable programming education, particularly for syntax checking, compilation, test-based correctness, and immediate feedback (Cheang et al., 2003; Ihantola et al., 2010; Wang et al., 2011). More recent artificial intelligence (AI) applications extend this landscape to code-quality assessment, feedback generation, performance monitoring, and tutoring, but the research remains fragmented across different pedagogical functions and data sources (Jukiewicz, 2024; Manorat et al., 2025).

The central limitation is conceptual rather than purely technical. A program that passes a test suite does not necessarily reveal the quality of the learner's conceptual organisation, abstraction, decomposition, or generalisation. Two students can obtain the same output while demonstrating markedly different levels of understanding through their use of control structures, data abstractions, modularity, reuse, and explanation. Conventional automated assessment can therefore be reliable for checking whether a program works while

remaining comparatively weak at explaining what level of understanding is visible in the submitted artefact (Cheang et al., 2003; Wang et al., 2011).

Cognitive taxonomies provide a principled way to move beyond correctness. Bloom's original taxonomy and its later revision organise educational objectives by increasingly complex cognitive processes and have influenced assessment design across disciplines (Anderson & Krathwohl, 2001; Bloom, 1956). However, programming artefacts often exhibit several cognitive operations simultaneously, making verb-based categorisation difficult when the object being classified is the produced response rather than the intended learning outcome. The Structure of Observed Learning Outcomes (SOLO) Taxonomy instead classifies the structural complexity of an observable response through five levels: Prestructural, Unistructural, Multistructural, Relational, and Extended Abstract (Biggs & Collis, 1982). SOLO has been used to analyse progression in university curricula and to characterise novice programming understanding, supporting its relevance for response-based cognitive assessment (Brabrand & Dahl, 2009; Lister et al., 2006).

At the same time, advances in code representation learning make richer automated assessment technically plausible. CodeBERT learns contextual representations jointly from programming and natural languages, enabling semantic modelling beyond surface tokens (Feng et al., 2020). Graph-based program representations can explicitly model syntactic and semantic relationships that token sequences obscure (Allamanis et al., 2018), while later structure-aware models such as GraphCodeBERT incorporate data-flow information into learned code representations (Guo et al., 2021). Language-agnostic approaches that combine source context with Abstract Syntax Tree (AST) structure further suggest a route toward transfer across programming languages (Zügner et al., 2021).

This study brings these strands together in AI-SOLO, a hybrid framework that integrates semantic code embeddings, graph-based structural representations, and behavioural process metadata to classify student submissions according to SOLO response complexity. The research addresses a specific gap: prior automated graders primarily evaluate technical correctness or code properties, cognitive taxonomies are commonly applied manually, and modern code models are rarely connected to an explicit response-structure taxonomy in a single educational pipeline (Ihantola et al., 2010; Manorat et al., 2025). The study therefore asks whether a multi-dimensional representation of code and learner behaviour can support automated SOLO classification, how classification errors are distributed across adjacent cognitive levels, and how such predictions can be translated into formative feedback for students and interpretable information for educators.

Earlier work by Arunprakash et al. (2025) explored an AI-driven framework that aligned programming assessment with Bloom's Taxonomy by combining semantic, structural, and behavioural indicators of student programming activity. That study demonstrated the potential of artificial intelligence to move programming assessment beyond conventional correctness-based grading toward cognitively informed evaluation. However, Bloom's Taxonomy primarily characterizes categories of cognitive processes and can be difficult to operationalize directly from observable programming artefacts, particularly when neighbouring cognitive processes overlap within a single programming task. Building on this earlier foundation, the present study adopts the Structure of the Observed Learning Outcome (SOLO) Taxonomy as an artefact-oriented developmental hierarchy and introduces the AI-SOLO framework, which integrates CodeBERT-derived semantic representations, graph neural network-based structural features, and behavioural programming metadata for automated cognitive assessment (Arunprakash et al., 2025).

## LITERATURE REVIEW AND RESEARCH GAP

### Automated Programming Assessment

Early automated assessment systems were motivated by the need to reduce grading workload and provide rapid feedback. Online judging environments demonstrated the scalability of test-based evaluation, but their primary decision criterion was whether submitted programs satisfied expected outputs (Cheang et al., 2003). Subsequent systems combined static and dynamic analysis to provide richer diagnostics, including syntax, structural defects, and specification compliance (Wang et al., 2011). Reviews of automated programming assessment consistently show that such systems are effective for operationalising correctness, testing, and

immediate feedback, but also reveal substantial diversity in what is assessed and limited attention to theoretically grounded cognitive development (Ihantola et al., 2010).

The newest generation of AI-assisted assessment adds natural-language feedback and holistic code judgement. For example, large language models can produce grading decisions and feedback that correlate with teacher judgements in bounded programming tasks, but reliability, repeatability, hallucination risk, and the need for human oversight remain important concerns (Jukiewicz, 2024). A recent systematic review of 119 studies found that AI in programming education increasingly spans course design, classroom implementation, assessment and feedback, and performance monitoring, yet the field still lacks unified pedagogical frameworks that connect advanced code representations directly to explicit models of learning progression (Manorat et al., 2025).

**Cognitive Taxonomies and Programming**

A preliminary step toward taxonomy-informed automated programming assessment was presented by Arunprakash et al. (2025), who proposed an AI-driven framework that aligned programming assessment with Bloom's Taxonomy by integrating semantic, structural, and behavioural characteristics of student programming activity. That framework demonstrated the potential of artificial intelligence to extend assessment beyond conventional correctness-based grading toward cognitive-level interpretation, adaptive feedback, and learner progression. However, Bloom's categories describe cognitive processes such as Apply, Analyze, Evaluate, and Create, which may overlap within a single programming activity and are not always directly observable from the submitted code artefact. The present study builds on this earlier work by shifting the assessment target from inferred cognitive processes to the structural quality demonstrated in the learner's observable response. Accordingly, SOLO Taxonomy provides a more directly operationalisable hierarchy for automated programming assessment because it distinguishes progression according to the integration, organisation, and abstraction evident within the produced solution (Arunprakash et al., 2025; Biggs & Collis, 1982).

Programming education research has shown that novice understanding can be examined meaningfully through SOLO because code and explanations expose identifiable structural relationships among concepts (Lister et al., 2006). At curriculum scale, SOLO has also been used to examine competence progression across hundreds of university courses, demonstrating that its levels can support systematic analysis of increasing complexity (Brabrand & Dahl, 2009). These properties make SOLO an attractive target for automated classification when the evidence includes both the semantics of a solution and its structural organisation.

**Semantic and Structural Representation of Source Code**

Modern program-representation research shows that source code should not be treated as an ordinary token sequence. Graph-based representations can encode long-range relationships among variables, control structures, and program entities, allowing graph neural networks to learn patterns that are difficult to capture from local syntax alone (Allamanis et al., 2018). CodeBERT complements this structural view by learning contextual semantic representations from paired natural-language and programming-language data (Feng et al., 2020). GraphCodeBERT extends pretrained code modelling by incorporating data flow, explicitly linking variable relations to contextual representations (Guo et al., 2021). These developments provide a technical basis for modelling both what a program expresses and how its internal relationships are organised.

Cross-language representation learning is also relevant to educational transferability. Models that combine source context and AST structure using language-agnostic features have demonstrated multilingual benefits across several programming languages (Zügner et al., 2021). This does not automatically make an educational assessment system language-independent, because task design, platform telemetry, coding conventions, and taxonomy rubrics also affect validity. It does, however, indicate that the structural and semantic components of an AI-SOLO-like architecture can potentially be adapted beyond Python if language-specific parsing and validation are handled carefully.

## Comparative Literature and Identified Gap

Table 1 positions the proposed framework against representative developments in automated grading, cognitive-taxonomy use, and learned code representation. The comparison is deliberately selective rather than exhaustive: it focuses on studies that establish the technical or pedagogical components necessary to understand the research gap. As Table 1 shows, prior work typically addresses only part of the problem—technical grading, cognitive classification, semantic representation, structural representation, or AI-generated feedback, whereas AI-SOLO attempts to combine these elements around an explicit SOLO-aligned assessment target.

**Table 1. Comparative analysis of representative literature and the research gap addressed by AI-SOLO**

| Study | Approach / Technique | Main Contribution | Limitation / Research Gap |
|---|---|---|---|
| Cheang et al. (2003) | Automated online judge using test cases | Technical correctness; no cognitive taxonomy | Efficient large-scale grading, but output-centred and not designed to infer depth of understanding. |
| Lister et al. (2006) | Manual/qualitative SOLO analysis of novice programming understanding | SOLO | Establishes relevance of SOLO to programming, but not an automated multi-modal classifier. |
| Ihantola et al. (2010) | Review of automated programming-assessment systems | Varied; mostly technical assessment | Shows maturity of automated grading but limited systematic cognitive-taxonomy integration. |
| Wang et al. (2011) | Static + dynamic automated analysis with immediate feedback | Ability-oriented; no explicit cognitive taxonomy | Adds structural checking but does not classify observed learning complexity with SOLO. |
| Allamanis et al. (2018) | Graph-based neural representation of programs | None | Models source-code structure effectively but is not an educational cognitive-assessment framework. |
| Feng et al. (2020) | CodeBERT semantic code representation | None | Provides contextual semantic embeddings, but no SOLO mapping, learner behaviour, or educational feedback layer. |
| Guo et al. (2021) | GraphCodeBERT with data-flow-aware pretraining | None | Integrates code structure into pretrained representations, but not a cognitive taxonomy or classroom assessment pipeline. |
| Zügner et al. (2021) | Language-agnostic learning from source context + AST | None | Supports multilingual structural transfer but does not evaluate cognitive learning levels. |
| Jukiewicz (2024) | LLM-based grading and feedback for Python assignments | No explicit cognitive taxonomy | Demonstrates AI-assisted grading potential but requires reliability safeguards and lacks response-structure taxonomy alignment. |
| Manorat et al. (2025) | Systematic review of AI in programming education | Broad pedagogical scope | Shows fragmented AI applications across assessment, feedback, tutoring, and monitoring; no single SOLO-aligned multi-dimensional architecture. |
| Arunprakash et al. (2025) | AI-supported Bloom's Taxonomy framework integrating semantic, | Proposed cognitively informed programming | Bloom-based cognitive categories can overlap when inferred from observable code; further empirical |

| | | | |
|---|---|---|---|
| | structural, and behavioural programming features | assessment and adaptive feedback beyond correctness-based grading | operationalization using an artefact-oriented developmental taxonomy was required. |
| AI-SOLO (current study) | CodeBERT + AST/PDG graph modelling + behavioural metadata + feature fusion | SOLO | Integrates semantic, structural, and process evidence; current limitations include Python-only evaluation, missing formal reliability statistic, no verified ablation, and no cross-language validation. |

As shown in Table 1, the authors' earlier Bloom-aligned framework established the conceptual feasibility of combining AI-based code analysis with cognitive taxonomy mapping (Arunprakash et al., 2025). The present study represents a substantive extension of that work rather than a direct replication: it replaces Bloom-level classification with the SOLO developmental hierarchy, formalizes semantic–structural multimodal modelling through CodeBERT and graph neural networks, and evaluates the resulting AI-SOLO architecture using student programming submissions.

The gap identified in Table 1 is therefore not the absence of automated grading, code-aware AI, or cognitive taxonomies individually. The gap is the lack of a unified assessment architecture in which semantic meaning, program structure, learner process evidence, and an observable-response taxonomy are jointly used to produce a cognitive classification. AI-SOLO is designed to address that integration gap. At the same time, the current study should be interpreted as an initial validation rather than definitive proof of generalisability, because several reviewer-requested validation elements especially formal inter-rater reliability coefficients, inferential comparisons, ablation experiments, and cross-language tests cannot be reconstructed from the retained study outputs and are not fabricated in this revision.

# METHODOLOGY

### Research Design, Participants, and Data Collection

A design-and-evaluation approach was used to develop and test AI-SOLO as an automated cognitive-assessment framework for programming education. The study was conducted over one academic year with 120 undergraduate students enrolled in introductory and intermediate Python programming courses. Programming activities were completed through Replit and HackerRank for Education. Across the study period, 1,824 unique submissions were retained, together with available platform-generated behavioural metadata. The problem bank contained tasks intended to elicit responses across the five SOLO levels, from fragmented or minimally relevant solutions to integrated and generalisable solutions (Biggs & Collis, 1982).

The use of authentic course submissions is important because automated assessment can otherwise overfit synthetic tasks or narrowly defined test suites. However, the present dataset is limited to Python and to the specific interaction logs available from the two platforms. This design therefore supports an in-context validation of the framework, not a claim that the same feature distributions will hold for Java, C/C++, JavaScript, or project-scale repositories. Recent work on language-agnostic structural code representation suggests that technical transfer is feasible, but educational validity still requires language- and context-specific testing (Zügner et al., 2021).

### SOLO Annotation and Ground-Truth Construction

Each submission was assigned to one of the five SOLO levels: Prestructural, Unistructural, Multistructural, Relational, or Extended Abstract (Biggs & Collis, 1982). The programming-specific rubric considered conceptual correctness, the number and coordination of relevant programming constructs, logical organisation, integration among components, and evidence of abstraction or generalisation. This response-structure orientation follows the use of SOLO to distinguish qualitatively different levels of programming understanding rather than simply counting correct elements (Lister et al., 2006).

Multiple computer-science educators independently reviewed submissions, and disagreements were resolved through structured discussion and consensus. The original study notes that annotation consistency was considered, but the retained materials do not contain separate rater-level labels or a reported reliability coefficient. Consequently, Cohen's kappa or Krippendorff's alpha cannot be reconstructed responsibly from the available evidence and is not invented here. This is a material limitation because reproducible cognitive labels require evidence of agreement beyond consensus adjudication; formal agreement statistics should accompany future replications when independent rating records are available (Cohen, 1960; Krippendorff, 2018).

## AI-SOLO Multi-Dimensional Architecture

AI-SOLO combines three evidence streams. The semantic stream uses CodeBERT embeddings to represent contextual meaning in source code (Feng et al., 2020). The structural stream transforms each program into graph-based representations derived from its Abstract Syntax Tree and Program Dependency Graph, allowing graph models to encode syntactic hierarchy and logical or dependency relationships that flat token sequences may miss (Allamanis et al., 2018). The behavioural stream uses available learning process metadata, including time-on-task and revision activity, to represent aspects of how the solution was produced rather than only the final artefact. The three streams are fused before five-class SOLO prediction.

Figure 1 presents the complete AI-SOLO workflow before deployment. A student submission is analysed in parallel by the semantic, structural, and behavioural components; the resulting representations are fused into a common feature space; the classifier produces a SOLO level and confidence score; and the output is translated into student-facing formative feedback and educator-facing dashboard information. This architecture operationalises the study's central premise that cognitive assessment should use complementary evidence rather than equating functional correctness with understanding.

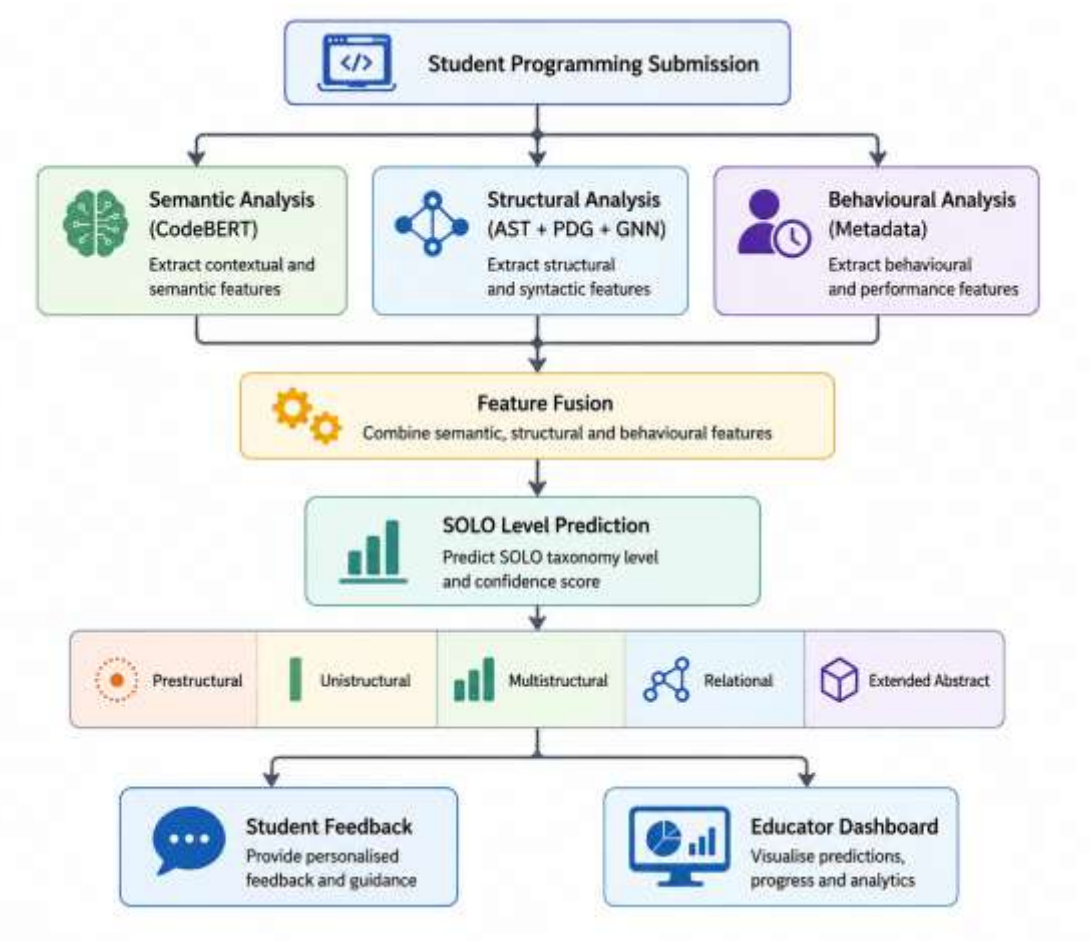


**Figure 1. Architectural overview of the multi-dimensional AI-SOLO pipeline**

As illustrated in Figure 1, the architecture separates evidence extraction from pedagogical interpretation. CodeBERT contributes contextual semantic information, while graph processing captures explicit structural

relationships in code; this distinction is consistent with research showing that semantic pretraining and structural representations provide complementary views of programs (Allamanis et al., 2018; Feng et al., 2020; Guo et al., 2021). Behavioural metadata adds a process dimension that neither the final code tokens nor the graph alone can represent. The fusion layer then maps these combined signals to the ordered SOLO categories.

### Training Protocol and Evaluation

The dataset was partitioned into training (70%), validation (15%), and test (15%) subsets. The study protocol states that a student's work was not reused across data partitions, reducing the risk that the model would recognise individual coding patterns rather than generalise to unseen students. Training was conducted for 20 epochs with the Adam optimiser, class-weighted cross-entropy loss to reduce the influence of class imbalance and early stopping to limit overfitting. Experiments were executed on an NVIDIA RTX 3090 GPU using PyTorch 2.1.0. The evaluation plan included overall and class-level accuracy, precision, recall, F1 score, and confusion-matrix analysis.

The retained results provide aggregate accuracy values and a confusion matrix, but they do not contain the complete per-instance prediction files required for robust paired significance testing against the reported baselines. Therefore, differences between AI-SOLO and comparator accuracies are treated as descriptive rather than statistically significant. A valid inferential comparison would require identical test instances and paired predictions, for example through McNemar-type testing or bootstrap confidence intervals. Reporting a p-value without those data would create false precision and is intentionally avoided in this revision.

The reviewer also requested an ablation study isolating semantic, structural, and behavioural contributions. No verified single-modality or leave-one-modality-out outputs are present in the retained experimental materials. Accordingly, this paper does not fabricate ablation numbers. The current evidence evaluates the fused system as a whole; the marginal contribution of each modality remains an open empirical question that should be addressed in a fully reproducible follow-up using semantic-only, structural-only, behavioural-only, pairwise-fusion, and full-fusion configurations under the same student-disjoint split.

### Deployment and Pedagogical Evaluation

The trained classifier was embedded in a web-based prototype that returned to a predicted SOLO level and confidence score. The student-facing component was designed to provide formative guidance aligned with the structural quality of the response, while the educator dashboard summarised class-level SOLO distributions and supporting evidence from semantic, structural, and behavioural streams. This emphasis on timely feedback is consistent with the broader evolution of automated programming assessment from simple judging toward richer diagnostic and instructional support (Jukiewicz, 2024; Manorat et al., 2025; Wang et al., 2011).

Educator perceptions were explored through structured interviews and Likert-scale questionnaires. The retained manuscript reports generally positive perceptions of the dashboard and multi-dimensional evidence, but does not provide the number of educator respondents, item-level questionnaire statistics, or a complete qualitative coding protocol. The qualitative findings are therefore interpreted as exploratory usability evidence rather than as a statistically generalisable evaluation of pedagogical impact.

## RESULTS

### Reported Aggregate Classification Performance

The archived aggregate results report an overall AI-SOLO classification accuracy of 91.4%. Performance decreases as the SOLO categories require greater integration and abstraction: the reported class-level classification rates are 97.2% for Prestructural, 94.5% for Unistructural, 92.8% for Multistructural, 89.1% for Relational, and 83.4% for Extended Abstract. Figure 2 presents these reported aggregate values and makes the gradient across the five SOLO levels visible.

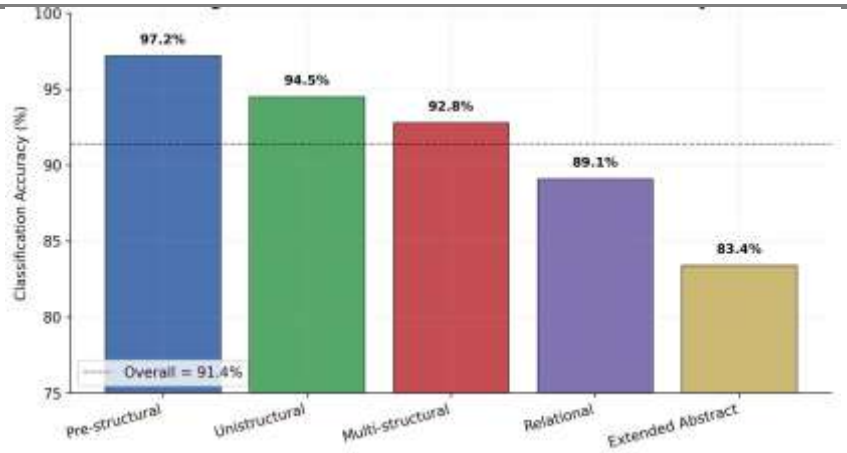


**Figure 2. Reported aggregate classification rates across the five SOLO Taxonomy levels**

Figure 2 shows that performance is strong for structurally simpler responses and declines toward the Extended Abstract level. This pattern is plausible because higher SOLO levels depend not only on the presence of relevant elements but on their integration, abstraction, and potential generalisation (Biggs & Collis, 1982). The 83.4% reported rate for Extended Abstract is therefore encouraging, but it should not be interpreted in isolation from the separate confusion-matrix output discussed below, which contains different evaluation counts and produces a lower Extended Abstract recall.

## Descriptive Baseline Comparison

The original experimental summary reports two comparator accuracies: 88.2% for a correctness-oriented baseline and 88.5% for a Bloom-aligned AI comparator, compared with 91.4% for AI-SOLO. These differences suggest a potential advantage for the multi-dimensional SOLO-aligned design, but the retained materials do not include matched prediction files, confidence intervals, or the full specification of the comparator pipelines. The comparison must therefore remain descriptive. It is more defensible to interpret the result as evidence that the proposed integration is promising than to claim statistically demonstrated superiority. This caution is especially important because cognitive taxonomy choice and model architecture change simultaneously, preventing the observed difference from being attributed to SOLO alone.

## Error Analysis and Confusion-Matrix Dynamics

To examine where classification errors occur, Figure 3 presents the archived 400-prediction confusion matrix in both count and row-normalised form. Unlike a single overall accuracy value, this matrix shows the direction of errors and is particularly useful for testing whether mistakes occur mainly between adjacent SOLO categories, which would be expected from a hierarchical response-complexity model (Biggs & Collis, 1982).

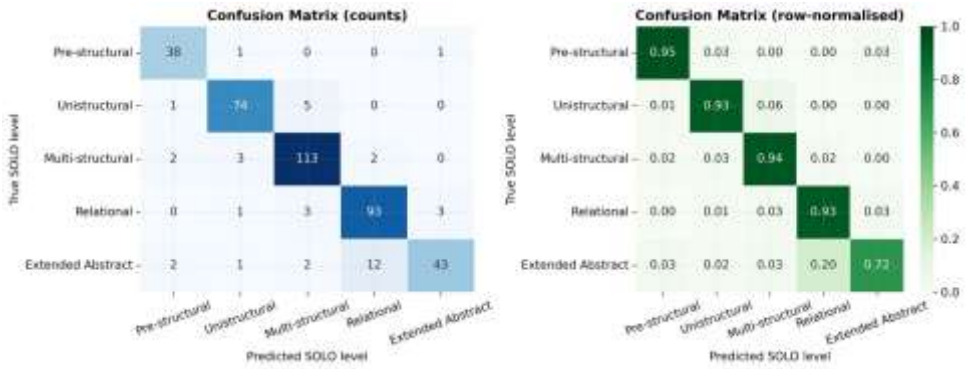


**Figure 3. Archived 400-prediction confusion matrix for diagnostic analysis of inter-level classification errors**

The counts displayed in Figure 3 contain 400 predictions, of which 361 are correct, giving an accuracy of 90.25% for this specific matrix. Prestructural recall is 95.0% (38/40), Unistructural recall is 92.5% (74/80),

Multistructural recall is 94.2% (113/120), Relational recall is 93.0% (93/100), and Extended Abstract recall is 71.7% (43/60). The most prominent error is Extended Abstract being predicted as Relational in 12 of 60 cases (20.0%), which supports the interpretation that the model has difficulty distinguishing integrated relational solutions from genuinely generalised or abstracted solutions. A small number of distant errors also occur, including Extended Abstract to Prestructural and Prestructural to Extended Abstract, so the error structure is not perfectly monotonic.

A critical reproducibility issue is visible when Figure 3 is compared with Figure 2. The 400-prediction matrix yields 90.25% overall accuracy and 71.7% Extended Abstract recall, whereas the aggregate summary reports 91.4% overall accuracy and 83.4% for Extended Abstract. In addition, a 15% test split of 1,824 submissions would not contain 400 cases. The retained files do not provide prediction identifiers or split manifests that would allow these outputs to be reconciled. For scientific accuracy, the two summaries are therefore reported as distinct archived evaluation outputs rather than treated as interchangeable measurements from the same test set. This discrepancy should be resolved from the original experiment logs before any future replication or secondary analysis.

**Training and Validation Dynamics**

Figure 4 shows the recorded training and validation accuracy trajectories across 20 epochs. Both curves rise rapidly during the early epochs and then converge more gradually, with validation accuracy reaching the reported 91.4% by the end of training. The training curve remains only modestly higher than the validation curve near the final epoch, which is more consistent with controlled convergence than with severe divergence between training and validation performance.

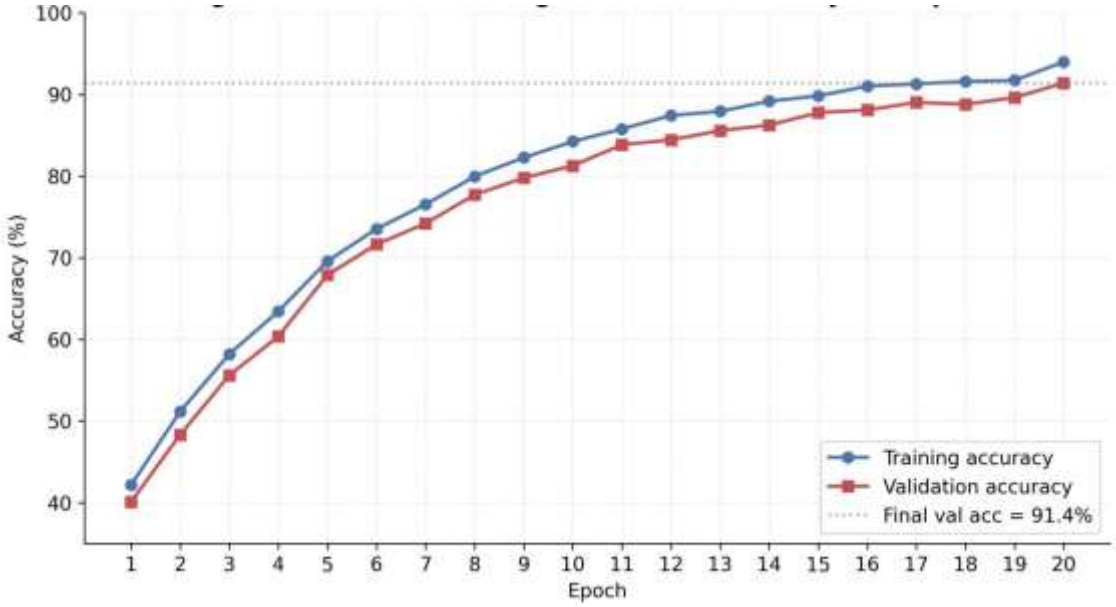


**Figure 4. Training and validation accuracy across 20 epochs**

Figure 4 should be interpreted only as an overall optimisation trace. The figure does not contain class-specific learning curves and therefore cannot independently support claims that particular SOLO classes saturated at specific epochs. Those stronger class-saturation claims have been removed from the revised interpretation. The figure nevertheless documents that the model's overall training and validation trajectories stabilised by the end of the 20-epoch schedule.

**Exploratory Pedagogical Findings**

Educator feedback indicated that combining semantic, structural, and behavioural evidence was perceived as more informative than a single correctness score, particularly for identifying students whose programs worked but showed weak organisation or limited abstraction. The dashboard was also perceived as useful for viewing class-wide cognitive distributions and identifying students who might benefit from targeted support. Because detailed respondent counts, questionnaire distributions, and qualitative coding evidence are not available in the

retained results, these observations should be treated as exploratory implementation feedback rather than formal proof of learning impact.

# DISCUSSION

## Interpretation of the Multi-Dimensional Approach

The main contribution of AI-SOLO is conceptual integration. Traditional automated graders are highly valuable for checking execution and specification compliance, but the resulting evidence is often dominated by correctness and technical defects (Cheang et al., 2003; Ihantola et al., 2010; Wang et al., 2011). AI-SOLO extends the assessment target by combining semantic meaning, graph-based organisation, and behavioural traces before mapping the evidence to a response-complexity taxonomy. This design is consistent with the view that programs contain both contextual and structural information and that richer representations can capture relationships not visible in flat token sequences (Allamanis et al., 2018; Feng et al., 2020; Guo et al., 2021).

The reported performance gradient across SOLO levels is also pedagogically meaningful. Prestructural and Unistructural responses are often characterised by missing, irrelevant, or isolated elements, which can be relatively distinctive computationally. Relational and Extended Abstract responses require the classifier to recognise how multiple elements are integrated and whether the solution demonstrates a transferable abstraction beyond the immediate task (Biggs & Collis, 1982). The confusion matrix's concentration of errors between Extended Abstract and Relational is therefore more informative than a single accuracy score: it identifies the boundary at which automated classification is most fragile.

## Why SOLO Is a Useful Assessment Target

Bloom's taxonomy remains highly valuable for curriculum planning and specifying intended cognitive processes (Anderson & Krathwohl, 2001; Bloom, 1956). AI-SOLO does not invalidate that role. Rather, SOLO offers a complementary advantage when the object of analysis is an observable programming response. The taxonomy asks how well the response's elements are organised and related, which maps naturally code artefacts in which modularity, dependencies, abstraction, and generalisation are visible properties (Biggs & Collis, 1982; Lister et al., 2006). This theoretical alignment is a stronger justification for SOLO than simply claiming that one taxonomy achieves a higher percentage than another on a single dataset.

The present findings also extend the authors' earlier Bloom's Taxonomy-aligned framework for AI-driven cognitive assessment in programming education (Arunprakash et al., 2025). That earlier work established the conceptual basis for moving beyond correctness-oriented programming assessment by integrating semantic, structural, and behavioural evidence with a cognitive taxonomy to support automated classification and adaptive feedback. The present study advances this foundation in two important respects. First, it shifts the assessment target from Bloom's cognitive-process categories to SOLO's observable hierarchy of response complexity, enabling cognitive classification to be grounded more directly in the structural and conceptual characteristics demonstrated within student code. Second, the AI-SOLO framework operationalises this theoretical shift through an integrated architecture combining CodeBERT semantic representations, graph-based structural modelling using ASTs, PDGs, and GNNs, and behavioural evidence from students' programming activities. The findings therefore represent a progression from the earlier Bloom-aligned conceptual framework toward an empirically evaluated, artefact-oriented SOLO-based assessment approach capable of distinguishing increasingly integrated levels of programming understanding (Arunprakash et al., 2025).

The distinction also clarifies the framework's novelty. The value of AI-SOLO is not that CodeBERT, graph neural networks, or automated feedback are new individually. Each has an established research base (Allamanis et al., 2018; Feng et al., 2020; Jukiewicz, 2024). The novelty lies in using complementary semantic, structural, and process representations to infer an explicitly pedagogical response-complexity construct and then exposing that inference through formative feedback and educator analytics.

**Pedagogical Utility and Explainability**

A cognitive classifier becomes educationally useful only when its output can inform action. A predicted SOLO level can support feedback such as identifying a relevant concept that is present but isolated, highlighting multiple correct elements that are not yet integrated, or encouraging a relational solution to generalise its abstraction. This is more pedagogically specific than returning only pass/fail or a numeric score. Recent AI-in-programming-education literature increasingly emphasises timely feedback and instructional support, but also warns that automated feedback should remain interpretable and subject to educator oversight (Jukiewicz, 2024; Manorat et al., 2025).

The current dashboard design partially addresses this need by presenting the predicted level together with supporting evidence streams. However, explainability should not be overstated. A dashboard that displays semantic, graph, and behavioural indicators is more transparent than an unexplained label, but it is not equivalent to a validated causal explanation of why a student learned in a particular way. Future implementations should expose feature contributions, uncertainty, and exemplar comparisons while avoiding deterministic claims about a learner's cognition from code traces alone.

**Transferability Across Languages and Platforms**

The present evaluation is restricted to Python, Replit, and HackerRank for Education. The semantic and structural architecture has a plausible path to broader transfer because pretrained code models and AST-based representations have been applied across multiple languages, and language-agnostic structure-plus-context learning has demonstrated multilingual benefits (Guo et al., 2021; Zügner et al., 2021). Nevertheless, the behavioural stream is more platform dependent. Time-on-task, revision counts, run frequency, hint usage, and event timestamps may be recorded differently—or not recorded at all—across learning environments.

A robust cross-platform implementation should therefore define a small canonical behavioural schema and map platform-specific events into that schema. Missing features should be explicitly handled rather than silently imputed as equivalent behaviour. Cross-language validation should likewise rebuild or validate the SOLO rubric for language-specific constructs, preserve student-disjoint evaluation, and include project-style tasks in which abstraction, modularity, testing, and design evolve over longer periods. The multilingual representation literature supports technical feasibility, but educational equivalence must be demonstrated empirically rather than assumed (Zügner et al., 2021).

**Methodological Limitations and Required Validation**

Several limitations materially constrain the strength of the conclusions. First, the evaluation is confined to introductory and intermediate Python tasks and platform-specific behavioural logs. Second, although multiple educators annotated the data, no reconstructable Cohen's kappa or Krippendorff's alpha is available, so the reliability of the SOLO labels cannot be quantified from the retained records (Cohen, 1960; Krippendorff, 2018). Third, the baseline comparisons are descriptive because paired predictions and full comparator specifications are unavailable. Fourth, no verified ablation experiment isolates the contribution of semantic, structural, and behavioural modalities. Fifth, the archived aggregate metrics and confusion matrix are numerically inconsistent and appear to represent different evaluation slices, preventing a single reconciled test-set account. Sixth, the qualitative educator evaluation lacks sufficient respondent and item-level detail for generalisable claims. Finally, formal prospective institutional ethical approval was not obtained, which must remain transparent when interpreting the educational data collection.

These limitations do not make the framework unusable, but they change the appropriate claim. The study provides promising proof-of-concept evidence for a SOLO-aligned, multi-dimensional assessment architecture; it does not yet establish definitive superiority, causal pedagogical benefit, or cross-context generalisability. A stronger next-stage validation should preregister the evaluation split, preserve rater-level annotations, report agreement statistics with confidence intervals, perform paired significance testing on identical test cases, conduct full modality ablations, evaluate at least one additional programming language and project-style task set, and publish enough implementation detail to reproduce graph construction, feature fusion, hyperparameters, and platform-event mapping.

# CONCLUSION AND RECOMMENDATIONS

This study introduced AI-SOLO, a hybrid framework for automated cognitive assessment of programming submissions using the Structure of Observed Learning Outcomes Taxonomy. The framework combines CodeBERT-based semantic representations, AST/PDG-derived graph structure, and behavioural metadata to move beyond correctness-only grading toward an estimate of observable response complexity (Biggs & Collis, 1982; Feng et al., 2020). Its design is supported by prior evidence that source-code semantics and structure provide complementary representations and that automated programming assessment benefits from richer diagnostic information than test outputs alone (Allamanis et al., 2018; Ihantola et al., 2010; Wang et al., 2011).

The archived aggregate results report 91.4% overall classification accuracy and an 83.4% Extended Abstract classification rate, while a separate 400-prediction confusion matrix yields 90.25% accuracy and 71.7% Extended Abstract recall. Reporting this discrepancy openly is essential: the available evidence supports the promise of architecture, but not a perfectly reconciled performance claim. The confusion pattern nevertheless identifies a meaningful challenge at the boundary between Relational and Extended Abstract responses, which should be a priority for future model and rubric refinement.

AI-SOLO's strongest contribution is therefore its pedagogical framing: semantic, structural, and process evidence are fused around a taxonomy designed to describe the quality of an observed response, and the resulting classification is intended to support formative feedback rather than replace educator judgement. Before broad deployment, the framework requires stronger reliability evidence, inferential comparison, ablation analysis, ethical governance, cross-language and project-based validation, and fully reproducible evaluation records. With those safeguards, the approach offers a credible direction for more explainable and learning-centred automated programming assessment (Manorat et al., 2025).

Based on these findings, future research should validate the AI-SOLO framework using larger, multi-institutional datasets and programming activities covering additional languages, project-based tasks, and open-ended problem-solving scenarios. Further studies should conduct systematic ablation experiments to quantify the individual contributions of semantic, structural, and behavioural representations and report inter-rater reliability for expert SOLO annotations. Developing platform-independent behavioural indicators would also improve transferability beyond specific coding environments. In addition, explainability mechanisms and educator-in-the-loop validation should be strengthened to ensure that automated cognitive classifications remain transparent, pedagogically meaningful, and suitable for responsible educational decision-making.

### Ethical Approval

Formal ethical approval was not obtained for this study. Nevertheless, all participants were informed about the purpose and use of the research data, and voluntary informed consent was obtained. All student submissions, behavioural records, interview responses, and questionnaire data were anonymised before analysis. Participation or withdrawal had no effect on students' grades or academic standing.

### Conflict of Interest

The authors declare that there are no known conflicts of interest or competing financial interests that could have influenced the work reported in this paper.

### Data Availability

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

# ACKNOWLEDGEMENTS

The authors express their sincere gratitude to the 120 undergraduate students who participated in this study by contributing their programming submissions. Special thanks are also extended to the panel of computer science educators who conducted the extensive SOLO annotation process and provided the expert insights necessary for

the development of the framework.

## REFERENCES

1. Allamanis, M., Brockschmidt, M., & Khademi, M. (2018). Learning to represent programs with graphs. In International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=BJOFETxR-
2. Anderson, L. W., & Krathwohl, D. R. (Eds.). (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Longman.
3. Arunprakash, N., Sangeetha, A., & Vishaliney, P. (2025). AI-Driven Cognitive Assessment in Programming Education: A Bloom's Taxonomy-Aligned Framework for Skill Development. 16th International Conference on Sustainable Built Environment and Next-generation Innovation & Advancement (ICSBE) 2025, Volume I, In Proceedings of ICSBE 2025 (pp. 546–553).
4. Biggs, J. B., & Collis, K. F. (1982). Evaluating the quality of learning: The SOLO taxonomy. Academic Press.
5. Bloom, B. S. (Ed.). (1956). Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain. Longmans, Green.
6. Brabrand, C., & Dahl, B. (2009). Using the SOLO taxonomy to analyze competence progression of university science curricula. Higher Education, 58(4), 531–549. https://doi.org/10.1007/s10734-009-9210-4
7. Cheang, B., Kurnia, A., Lim, A., & Oon, W.-C. (2003). On automated grading of programming assignments in an academic institution. Computers & Education, 41(2), 121–131. https://doi.org/10.1016/S0360-1315(03)00030-7
8. Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), 37–46. https://doi.org/10.1177/001316446002000104
9. Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. In Findings of the Association for Computational Linguistics: EMNLP 2020 (pp. 1536–1547). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.findings-emnlp.139
10. Guo, D., Ren, S., Lu, S., Feng, Z., Tang, D., Liu, S., Zhou, L., Duan, N., Svyatkovskiy, A., Fu, S., Tufano, M., Deng, S. K., Clement, C., Drain, D., Sundaresan, N., Yin, J., Jiang, D., & Zhou, M. (2021). GraphCodeBERT: Pre-training code representations with data flow. In International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=jLoC4ez43PZ
11. Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. In Proceedings of the 10th Koli Calling International Conference on Computing Education Research (pp. 86–93). Association for Computing Machinery. https://doi.org/10.1145/1930464.1930480
12. Jukiewicz, M. (2024). The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process. Thinking Skills and Creativity, 52, 101522. https://doi.org/10.1016/j.tsc.2024.101522
13. Krippendorff, K. (2018). Content analysis: An introduction to its methodology (4th ed.). SAGE.
14. Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: Novice programmers and the SOLO taxonomy. In Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (pp. 118–122). Association for Computing Machinery. https://doi.org/10.1145/1140124.1140157
15. Manorat, P., Tuarob, S., & Pongpaichet, S. (2025). Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence, 8, 100403. https://doi.org/10.1016/j.caeai.2025.100403
16. Wang, T., Su, X., Ma, P., Wang, Y., & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. Computers & Education, 56(1), 220–226. https://doi.org/10.1016/j.compedu.2010.08.003
17. Zügner, D., Kirschstein, T., Catasta, M., Leskovec, J., & Günnemann, S. (2021). Language-agnostic representation learning of source code from structure and context. In International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=Xh5eMZVONGF