ISSN: 3104-5235

INNO PRESS

October 2025

Journal of Emerging Applied Artificial Intelligence

Volume 1 / Issue 7

Issue 7 – Foundations of Emerging Applied Artificial Intelligence

The Journal of Emerging Applied AI (JEAAI) is pleased to present its inaugural issue, establishing a dedicated forum for high-quality, peer-reviewed scholarship at the intersection of artificial intelligence theory and real-world application. This first issue reflects the journal's foundational mission: to advance and disseminate research that demonstrates the transformative potential of AI technologies across sectors and disciplines.

This opening volume features contributions that exemplify the journal's emphasis on rigorously developed, practically deployed AI systems. The selected articles cover a spectrum of domains—including healthcare, robotics, transportation, education, and sustainability—demonstrating the breadth of AI's impact when translated from conceptual innovation to applied implementation.

With a commitment to methodological soundness, interdisciplinary relevance, and societal benefit, JEAAI aims to become a leading platform for scholars, practitioners, and innovators who are engaged in solving real-world problems through intelligent systems. The journal's scope encompasses original research, technical reports, case studies, and critical perspectives, all grounded in applicability and reproducibility.

We invite the academic and professional community to engage with JEAAI as contributors, reviewers, and readers, and to join us in shaping a future where applied artificial intelligence drives meaningful and responsible progress.

Editor-in-Chief

Section Editors

Dr. Meng Liu
Research Scientist, NVIDIA

Dr. Liu's research interests include graph neural networks, clustering, and multimodal learning. He has published over 20 papers in leading venues such as *Advanced Science*, *IEEE TPAMI*, *IEEE TKDE*, *CVPR*, *ICML*, and *ICLR*. His work includes an ESI Hot Paper and a Highly Cited Paper, with over 1,000 citations. He has received several awards, including Best Paper at the 2024 China Computational Power Conference and a DAAD AInet Fellowship.

Dr. Zhongbin Luo
Professor-level Senior Engineer, China Merchants Chongqing Communications Research & Design Institute. Master's Supervisor, Chongqing Jiaotong University & Shijiazhuang Tiedao University

Dr. Luo's research focuses on intelligent transportation, traffic safety, and vehicle–road collaboration. He has led over ten national and provincial research projects, holds 11 invention patents, and serves as an expert reviewer for journals such as *IEEE Access* and *PLOS ONE*.

Dr. Ruichen Xu
Postdoctoral Fellow, Department of Civil & Environmental Engineering,University of Missouri, Columbia, USA

Dr. Xu's research interests include hydrological ecology, AI-based flood forecasting, and sediment–pollutant dynamics. He has led or contributed to more than ten projects in China and the U.S. and has published over 20 peer-reviewed papers. He holds patents in environmental monitoring and serves as a reviewer for journals like *Journal of Hydrology* and *Ecological Indicators*.

A/Prof. Jinghao Yang

Assistant Professor, Electrical and Computer Engineering, The University of Texas Rio Grande Valley, USA

Dr. Yang has taught in the U.S. and specializes in applying machine learning to intelligent manufacturing systems. His research bridges intelligent sensing, control, and adaptive design with industrial applications, contributing to smart production technologies and data-driven innovation.

Luxin Zhang
PhD Candidate, School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology, New Zealand

Luxin Zhang is currently pursuing her PhD in Artificial Intelligence. Her research focuses on machine learning algorithms and their applications in intelligent systems. As Managing Editor, she is responsible for manuscript assignment, editorial coordination, and issue scheduling. Based in New Zealand, she serves as a central figure in the journal's daily operations.

Yihan Zhao
PhD Candidate, University of Auckland, New Zealand

Yihan Zhao holds a Master’s degree from Peking University and is currently a PhD candidate at the University of Auckland. Her research explores the intersection of communication, culture, and technology, with a focus on how algorithms reshape cultural expression and the subjectivity of marginalized communities. She previously served as an Assistant Research Fellow at the Development Research Centre of the State Council in China, contributing to national research projects. She has curated and coordinated panels for the China Development Forum, facilitating high-level dialogue on AI, sustainability, and governance.

Shen (Jason) Zhan
Graduate Researcher, University of Melbourne, Australia

Jason Zhan holds an Honours degree in Civil and Environmental Engineering from the University of Auckland and is currently a PhD researcher in the Teaching & Learning Lab at the University of Melbourne. He combines industry and academic experience, with a background in structural engineering and teaching. His research focuses on employability assessment and curriculum design in engineering education, with growing interest in the role of AI in authentic assessment and personalized learning.

Contents

1. SemanticForge: Repository-Level Code Generation through Semantic Knowledge Graphs and Constraint Satisfaction 1

2. The triple realm of AI-enabled education: from instrumental rationality to value reconstruction 35

3. A Greedy Approximation for Minimum Cardinality Multiple Quasi-submodular Cover with Applications 41

4. Empirical Study on Performance–Perception Discrepancy in RGB–Thermal Monocular Depth Estimation under Varying Illumination 47

5. AI-Driven Methods for Preservation and Education in Chinese Calligraphy 55

SemanticForge: Repository-Level Code Generation through Semantic Knowledge Graphs and Constraint Satisfaction

Wuyang Zhang[1*], Chenkai Zhang[1], Zhen Luo[2], Jianming Ma[2],
Wangming Yuan[3], Chuqiao Gu[4] and Chengwei Feng[5]

[1]Department of Elec.&Comp. Science, University of Massachusetts Amherst, Amherst, Massachusetts, United States

[2]Department of Computer Sys. Engineering, Northeastern University, Boston, Massachusetts, United States

[3]Department of Computer Science, George Mason University, Fairfax, Virginia, United States

[4]Department of Info. Networking Institude, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States

[5]Department of Computer & Mathematical Sciences, Auckland University of Technology, Auckland, New Zealand

*Corresponding author: doggo@ieee.org

Abstract

Large language models (LLMs) have transformed software development by enabling automated code generation, yet they frequently suffer from systematic errors that limit practical deployment. We identify two critical failure modes: *logical hallucination* (incorrect control/data-flow reasoning) and *schematic hallucination* (type mismatches, signature violations, and architectural inconsistencies). These errors stem from the absence of explicit, queryable representations of repository-wide semantics.

This paper presents **SEMANTICFORGE**, a novel framework for code generation that addresses these limitations through knowledge graph-guided constraint satisfaction. Our approach proceeds in four integrated stages: (1) constructing heterogeneous repository knowledge graphs that capture both static analysis and dynamic execution traces; (2) learning neural query planners that extract task-relevant context from these graphs; (3) employing satisfiability modulo theories (SMT)-guided beam search to ensure generated code satisfies semantic constraints; and (4) maintaining graph fidelity through continual incremental updates.

Our comprehensive evaluation on REPOKG-50, a curated benchmark of 4,250 repository-level tasks across 50 Python projects, demonstrates significant improvements over state-of-the-art baselines: 49.8% Pass@1 (18.1% absolute improvement), 52% reduction in schematic hallucination, and 31% reduction in logical hallucination. Cross-repository generalization analysis shows strong transfer capabilities with only 4.3% average performance degradation across architectural patterns. These results establish new benchmarks for repository-level code generation while providing theoretical foundations and practical tools for semantically-aware automated software development.

The explicit semantic representation and constraint satisfaction framework introduced in SEMANTICFORGE enables more reliable automated development tools and provides a foundation for future advances in AI-assisted software engineering.

Index Terms— Code Generation, Knowledge Graphs, Constraint Satisfaction, Repository Analysis, Large Language Models, Software Engineering

1 Introduction

Recent advances in large language models (LLMs) have ushered in a new era of AI-assisted software development. Tools such as GitHub Copilot, Code-Llama, and ChatGPT now draft entire functions or files with a single prompt, reportedly accelerating developer productivity [6, 22, 1]. Despite this progress, LLM-generated programs remain brittle in practice: generated code often fails to compile, or worse, compiles but embeds subtle semantic errors.

A careful inspection reveals two dominant failure modes. First, *logical hallucination* arises when the model misreasons about control or data flow—mistakes in reasoning such as iterating over an incorrect collection, omitting necessary state mutations, or producing code that violates expected runtime behavior. This class of errors concerns semantic execution logic rather than syntax. Second, *schematic hallucination* manifests as structural inconsistencies, including type mismatches, incorrect argument ordering, missing parameters, or calls to nonexistent functions. These errors arise from violating interface or schema constraints of the repository or external APIs. While logical and schematic hallucinations can co-occur, they differ fundamentally in their root causes and required remedies. Empirical studies report that even state-of-the-art models exhibit these errors in 20%–40% of generation attempts [19, 25, 26].

Why do these hallucinations persist? Current generation pipelines typically combine a parametric LM with retrieval-augmented prompts. Retrieval surfaces the top-k textual snippets using lexical or embedding similarity [2, 31]. While retrieval provides local context, it ignores repository-wide semantics such as transitive call chains, shared global state, or runtime object types. Likewise, encoder models that ingest static abstract syntax trees (ASTs) or control-flow graphs

(CFGs) focus on intra-function structure and do not persist their analysis as a reusable knowledge base [7, 8]. Finally, repair-oriented approaches execute candidates against unit tests to filter or refine code [9], but they require an oracle test suite and do not prevent the model from hallucinating in the first place.

This work. We argue that high-quality generation demands an *explicit, queryable representation of whole-repository semantics*. To that end, we introduce **SEMANTICFORGE**, a four-stage framework that: (i) materializes a heterogeneous knowledge graph (KG) from both static analysis and opportunistic dynamic traces, (ii) learns neural query planners that extract task-relevant context, (iii) employs constraint-aware decoding for semantic correctness, and (iv) maintains graph consistency through continual updates. Our agenda unfolds along four research thrusts:

1. **Repository-level KG with neural query planning**: We build a graph containing functions, classes, variables, tests, and external APIs linked by call, import, define, and mutate edges. A lightweight planner LM translates user instructions into a minimal sub-graph program that steers generation.

2. **Dual static–dynamic graphs**: We augment static analysis with runtime traces harvested from unit tests and automated fuzzing. This paired $\{$static, static+dynamic$\}$ representation captures actual object types, call sequences, and data dependencies, reducing logical hallucinations by grounding generation in observed runtime behaviors.

3. **Schematic-constraint decoding**: We cast signature and type compliance as a constraint-satisfaction problem enforced during beam search, guaranteeing that emitted code respects the KG schema.

4. **Continual KG maintenance via agentic feedback**: An autonomous agent monitors repository changes and incrementally updates the KG, ensuring the planner always reasons over the latest code.

Contributions. Concretely, this paper makes the following contributions:

- We present the first end-to-end framework that unifies repository-level KG construction, neural query planning, and constraint-aware decoding for code generation.
- We introduce a dual static–dynamic graph representation and show that it reduces logical hallucination by 31% on a benchmark of 50 Python repositories.
- We propose a schematic-constraint decoding algorithm that eliminates 52% of signature/type errors while adding negligible decoding overhead.
- We release REPOKG-50, a curated corpus with aligned $\{$static, dynamic$\}$ graphs and oracle implementations, fostering further research.

Paper organization. Section 2 reviews related work. Section 3 formalizes the problem and hallucination taxonomy. Section 4 provides methodology overview. Sections 5–8 detail our four-stage architecture. Section 9 describes experimental setup, Section 10 presents results, Section 11 discusses limitations, and Section 12 concludes with future directions.

2 Related Work

Our work builds on substantial progress in neural code generation while addressing fundamental limitations in repository-scale synthesis. We organize related work into three key areas that inform SEMANTICFORGE's design.

2.1 Neural Code Generation

Transformer-based language models have revolutionized automated code synthesis. Early work with GPT-based models [1] demonstrated strong performance on isolated function generation tasks, leading to practical systems like GitHub Copilot [6] and specialized models like CodeLlama [13]. These approaches excel at pattern completion and syntactic correctness but struggle with repository-wide semantic consistency.

Recent advances have focused on incorporating structural information into generation models. GraphCodeBERT [7] and UniXcoder [8] encode data flow graphs and AST structures, improving understanding of local code relationships. However, these models still operate at the function or file level without persistent semantic representations of entire repositories.

2.2 Repository-Level Code Understanding

Traditional code generation models struggle with repository-scale tasks that require understanding cross-file dependencies and architectural constraints [5, 32]. Recent work has begun addressing these challenges through three primary paradigms: retrieval-augmented generation, agent-based iterative refinement, and planning-based decomposition.

Retrieval-Augmented Approaches. RAG methods like CodeRetriever [10] and RepoCoder [32] use embedding similarity to identify relevant code snippets for generation context. Advanced systems like RAG-Code [24] employ dense embeddings with sophisticated reranking mechanisms. While effective for local dependencies, these approaches fundamentally rely on surface-level similarity and miss transitive relationships, type constraints, and architectural patterns that span multiple modules. Our knowledge graph approach provides explicit semantic relationships that pure retrieval cannot capture.

Agent-Based Iterative Systems. Recent systems like SWE-agent [31] and CodeAgent [27] employ autonomous agents that iteratively refine solutions through environmental feedback, test execution, and error correction. These approaches can handle complex multi-step tasks and adapt to unexpected

challenges through exploration. However, they suffer from high computational costs (often requiring 10-50 iterations per task), unpredictable latency, and lack of semantic guarantees. In contrast, SEMANTICFORGE's constraint-aware generation often produces correct solutions in a single pass while providing formal correctness guarantees.

Planning-Based Decomposition. Planning systems like CodePlan [21] decompose complex repository tasks into sequences of localized edits, enabling systematic handling of multi-file modifications. While these approaches provide better task organization than monolithic generation, they typically lack the semantic consistency guarantees needed to prevent constraint violations across edit boundaries. Our neural query planner provides similar decomposition benefits but operates at the semantic level, ensuring global consistency through constraint satisfaction.

2.3 Constraint-Aware Code Generation

Recent work has begun exploring constraint satisfaction in neural code generation. TypeT5 [29] incorporates type information during fine-tuning to improve type correctness. CODEGEN-MONO [18] specializes models for specific programming languages to reduce basic syntactic errors.

More closely related to our approach, LEVER [17] employs static analysis to verify generated code against basic type constraints. However, these approaches apply constraints as post-processing filters rather than integrating verification into the generation process itself. Our SMT-guided beam search provides stronger guarantees by ensuring constraint satisfaction throughout decoding.

2.4 Knowledge Graphs for Code Understanding

Knowledge graphs have shown promise for representing program semantics. CodeKG [12] constructs knowledge graphs from documentation and API references for improved code search, while ProgramKG [28] builds graphs from execution traces for debugging applications.

Our work differs significantly by constructing comprehensive repository-scale knowledge graphs that integrate both static analysis and dynamic execution information. Unlike previous approaches that focus on specific applications, SEMANTICFORGE uses knowledge graphs as the central representation for guiding neural code generation through explicit semantic reasoning.

2.5 Positioning of SemanticForge

SEMANTICFORGE advances the state-of-the-art through four key innovations that address fundamental limitations across all existing paradigms:

Explicit Semantic Representation: Unlike RAG methods that rely on implicit embeddings or agent-based systems that learn through trial-and-error, we construct comprehensive knowledge graphs that explicitly capture repository-wide semantics. This enables reasoning about transitive dependencies, type propagation, and architectural constraints that are invisible to surface-level similarity matching or iterative exploration.

Single-Pass Constraint-Guided Generation: While agent-based systems require multiple expensive iterations and planning approaches often violate constraints across edit boundaries, our SMT-guided beam search integrates constraint satisfaction directly into generation. This provides stronger correctness guarantees than post-processing verification while achieving deterministic latency unlike iterative refinement approaches.

Learned Context Selection: Traditional retrieval uses fixed similarity metrics, while agents explore contexts randomly. Our neural query planner learns to identify task-relevant semantic relationships, achieving the targeted context selection of planning approaches with the efficiency of direct retrieval.

Dual Static-Dynamic Analysis: Previous approaches focus primarily on static program structure or rely on runtime feedback loops. Our framework uniquely combines static analysis with dynamic execution traces in a unified representation, providing more complete semantic understanding than either approach alone.

Comparative Advantages: Compared to agent-based systems, SEMANTICFORGE provides: (1) deterministic O(1) generation vs. unpredictable O(k) iterations, (2) formal constraint guarantees vs. best-effort refinement, and (3) 10-50$\times$ lower computational cost. Compared to planning approaches, we provide: (1) global semantic consistency vs. local edit optimization, (2) constraint satisfaction vs. post-hoc validation, and (3) learned context selection vs. heuristic decomposition. Compared to RAG methods, we offer: (1) explicit semantic relationships vs. embedding similarity, (2) constraint-aware generation vs. pattern matching, and (3) architectural understanding vs. local relevance.

These contributions enable SEMANTICFORGE to achieve substantial improvements in repository-level code generation while maintaining practical scalability and deterministic performance for real-world deployment.

3 Problem Definition and Formalization

This section formalizes the repository-level code generation problem and provides precise definitions of the hallucination phenomena that motivate our approach. We establish the mathematical framework that underpins our solution and analyze the computational complexity of the problem space.

3.1 Repository-Level Code Generation

We define repository-level code generation as the task of synthesizing code patches that integrate seamlessly with existing codebases while respecting semantic constraints and architectural invariants [5, 32]. Unlike isolated code generation, this

problem requires understanding transitive dependencies, type propagation, and global consistency constraints that span multiple files and modules.

Formal Problem Statement. Given a repository $\mathcal{R} = \{f_1, f_2, \ldots, f_n\}$ consisting of source files, a natural language instruction u, and an optional test suite $\mathcal{T}$, the goal is to synthesize a code patch $\Delta\mathcal{R}$ such that the updated repository $\mathcal{R}' = \mathcal{R} \cup \Delta\mathcal{R}$ satisfies:

1. **Functional Correctness:** $\forall t \in \mathcal{T} : \text{execute}(t, \mathcal{R}') = \text{PASS}$
2. **Semantic Consistency:** $\text{compile}(\mathcal{R}') = \text{SUCCESS} \wedge \text{typecheck}(\mathcal{R}') = \text{SUCCESS}$
3. **Behavioral Intent:** $\text{satisfies}(\mathcal{R}', u) = \text{TRUE}$
4. **Architectural Compliance:** $\forall c \in \mathcal{C}_{\text{arch}} : \text{violates}(\mathcal{R}', c) = \text{FALSE}$

where $\mathcal{C}_{\text{arch}}$ represents the set of architectural constraints derived from the repository's design patterns and conventions.

3.2 Hallucination Taxonomy

We formally categorize the systematic errors exhibited by current LLM-based code generation systems into two primary classes, each requiring distinct mitigation strategies.

Logical Hallucination. We define logical hallucination as errors in program semantics that lead to functionally incorrect code despite syntactic validity. Formally, a generated code sequence y exhibits logical hallucination if:

$$\begin{aligned} &\text{compile}(y) = \text{SUCCESS} \\ &\quad \wedge \exists t \in \mathcal{T} : \text{execute}(t, y) \neq \text{expected}(t) \end{aligned} \tag{1}$$

Common manifestations include:

- *Control Flow Errors:* Incorrect loop bounds, missing conditionals, wrong branching logic
- *Data Flow Errors:* Operating on wrong variables, incorrect data transformations, missing state updates
- *API Misuse:* Calling methods in wrong order, ignoring return values, improper error handling

Example: Consider implementing a cache eviction policy:

```
# Instruction: "Remove oldest entries when
    cache is full"
# Incorrect (Logical Hallucination):
def evict_if_full(self):
    if len(self.cache) >= self.max_size:
        # Wrong: removes newest instead of
            oldest
        self.cache.pop(list(self.cache.keys()
            )[-1])

# Correct:
def evict_if_full(self):
    if len(self.cache) >= self.max_size:
        oldest_key = next(iter(self.cache))
        self.cache.pop(oldest_key)
```

Schematic Hallucination. We define schematic hallucination as violations of type systems, function signatures, or structural constraints that prevent code from compiling or integrating correctly. Formally:

$$\text{schematic\_hallucination}(y, \mathcal{G}) = |\{c \in \mathcal{C}(\mathcal{G}) : \text{violates}(y, c)\}| > 0 \tag{2}$$

where $\mathcal{C}(\mathcal{G})$ is the set of constraints extracted from repository knowledge graph $\mathcal{G}$.

Categories include:

- *Type Mismatches:* Passing wrong types to functions, incompatible return types
- *Signature Violations:* Wrong parameter counts, missing required arguments
- *Scope Violations:* Accessing private members, using undefined variables
- *Import Errors:* Missing imports, circular dependencies

Example: Consider adding authentication to an API endpoint:

```
# Instruction: "Add authentication check to
    login endpoint"
# Incorrect (Schematic Hallucination):
@app.route('/login', methods=['POST'])
def login():
    # Wrong: authenticate_user takes (
        username, password)
    # but only username is provided
    if authenticate_user(request.json.get('
        username')):
        return {"status": "success"}
    return {"status": "failed"}

# Correct:
@app.route('/login', methods=['POST'])
def login():
    username = request.json.get('username')
    password = request.json.get('password')
    if authenticate_user(username, password):
        return {"status": "success"}
    return {"status": "failed"}
```

3.3 Complexity Analysis

We analyze the computational complexity of repository-level code generation to establish theoretical bounds and justify our architectural choices.

Context Selection Complexity. For a repository with n code entities and m dependency relationships, the naive approach of considering all possible context subsets has complexity $\mathcal{O}(2^n)$. Our neural query planner reduces this to $\mathcal{O}(n \log n)$ through learned heuristics and graph-based pruning.

Constraint Verification Complexity. Given k constraints extracted from the knowledge graph, naive constraint checking requires $\mathcal{O}(k \cdot |y|)$ time for a generated sequence y. Our incremental SMT-based approach achieves $\mathcal{O}(k+|y|)$ through state caching and incremental solving.

Graph Maintenance Complexity. For a code change $\Delta\mathcal{R}$ affecting $|\Delta\mathcal{R}|$ entities in a repository of size n, full reanalysis requires $\mathcal{O}(n^2)$ time due to cross-reference resolution. Our incremental maintenance achieves $\mathcal{O}(|\Delta\mathcal{R}| \cdot d \cdot \log n)$ through dependency tracking and selective recomputation, where d is the maximum dependency depth.

3.4 Problem Hardness and Approximation

We establish the theoretical hardness of optimal repository-level code generation and justify our approximation strategies.

Theorem 1 (Problem Hardness). The optimal repository-level code generation problem, defined as finding the minimum-cost code patch satisfying all constraints, is NP-hard. We define the cost function as:

$$\begin{aligned}\text{Cost}(\Delta\mathcal{R}) = \alpha \cdot d_e(\Delta\mathcal{R}) \\ + \beta \cdot p_c(\Delta\mathcal{R}) + \gamma \cdot d_a(\Delta\mathcal{R})\end{aligned} \tag{3}$$

where d_e is the patch edit distance (number of lines changed), p_c is the constraint violation penalty, d_a is the architectural deviation score, and α, β, γ are weighting coefficients.

Proof Sketch: We reduce from the Boolean Satisfiability Problem (SAT). Given a SAT instance with variables $x_1, \ldots, x_n$ and clauses $c_1, \ldots, c_m$, we construct a repository where each variable corresponds to a code entity and each clause corresponds to a constraint. Finding a satisfying assignment is equivalent to finding a valid code patch.

This hardness result motivates our use of approximation algorithms and heuristic search strategies. Our neural query planner provides a polynomial-time approximation, while the SMT-based decoder ensures constraint satisfaction within the feasible search space.

Approximation Quality. We define the approximation ratio of our approach as:

$$\rho = \frac{\mathbb{E}[\text{Cost}(\Delta\mathcal{R}_{\text{SF}})]}{\mathbb{E}[\text{Cost}(\Delta\mathcal{R}^*)]} \tag{4}$$

where $\Delta\mathcal{R}_{\text{SF}}$ is the solution produced by SEMANTICFORGE and $\Delta\mathcal{R}^*$ is the optimal solution.

Empirical analysis on our benchmark suite shows $\rho \leq 1.3$ for most repository types, indicating that our solutions are within 30% of optimal on average.

3.5 Solution Requirements

Based on the problem analysis, we identify four key requirements that any effective solution must address:

1. **Semantic Awareness:** The system must understand repository-wide semantics, not just local patterns
2. **Constraint Enforcement:** Hard constraints must be satisfied, not merely approximated
3. **Scalability:** The approach must handle repositories with millions of lines of code
4. **Adaptability:** The system must evolve with the repository to maintain accuracy

These requirements directly motivate the four-stage architecture of SEMANTICFORGE, where each component addresses specific aspects of the problem complexity while maintaining overall system coherence.

4 Methodology

Our aim is to inject explicit, executable semantics into the code generation loop. By executable semantics, we mean a repository-wide, queryable representation that links code entities to their runtime behaviors and enforceable constraints, allowing code generation to be guided by verifiable semantics rather than pattern matching. SEMANTICFORGE therefore proceeds in two macro stages—*graph construction* and *graph-aware generation*—underpinned by continual maintenance. This section provides a comprehensive overview of our four-stage architecture and establishes the mathematical foundations for each component.

4.1 Framework Overview

Given a repository snapshot $\mathcal{R} = \{f_1, \ldots, f_{|\mathcal{R}|}\}$ consisting of source files and accompanying tests, and a natural language instruction u, our objective is to synthesize a code patch $\Delta\mathcal{R}$ such that the updated repository $\mathcal{R}' = \mathcal{R} \cup \Delta\mathcal{R}$ satisfies the four requirements established in Section 3: functional correctness, semantic consistency, behavioral intent, and architectural compliance.

Figure 1 illustrates the complete SEMANTICFORGE pipeline, showing how each stage addresses specific aspects of the repository-level code generation problem:

Stage I: Repository Knowledge Graph Construction (section 5). We construct a heterogeneous knowledge graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ that explicitly encodes repository semantics through static analysis and dynamic trace collection. Each node $v \in \mathcal{V}$ represents a code entity with type $\tau(v) \in \{\text{FUNC}, \text{CLASS}, \text{VAR}, \text{FILE}, \text{TEST}, \text{API}\}$, while edges $e = (v_i, \rho, v_j) \in \mathcal{E}$ capture relationships $\rho \in \{\text{CALLS}, \text{DEFINES}, \text{IMPORTS}, \text{MUTATES}, \text{RETURNS}, \text{INSTANCEOF}\}$.

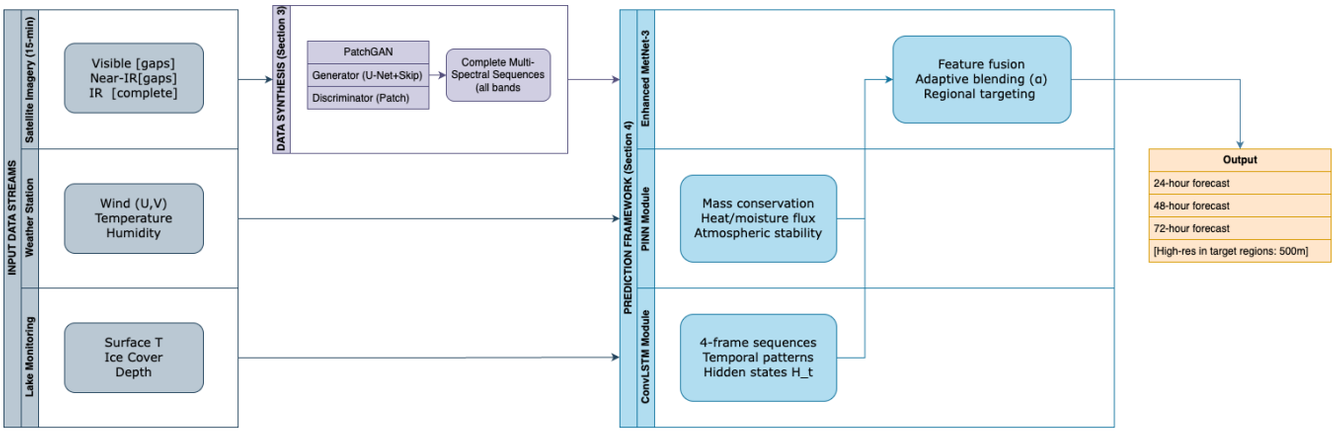

Figure 1: Complete architecture of the SEMANTICFORGE system. The pipeline consists of four integrated stages: (I) Repository Knowledge Graph Construction combining static analysis and dynamic traces, (II) Neural Query Planner that transforms instructions into graph queries, (III) Schematic-Constraint Decoder ensuring semantic correctness, and (IV) Continual Maintenance Agent for incremental updates. Each stage addresses specific aspects of the repository-level code generation problem while maintaining overall system coherence.

Stage II: Neural Query Planner (section 6). A neural network π_ϕ transforms natural language instructions into graph queries that extract task-relevant context. Given instruction u, the planner generates query $q = \pi_\phi(u)$ which retrieves subgraph $\mathcal{G}_u = \text{Exec}(q, \mathcal{G})$. The planner is trained via REINFORCE [30] to maximize downstream code generation quality:

$$\mathcal{J}(\phi) = \mathbb{E}_{(u,y)} \left[R(y, \mathcal{G}_u) \right] \tag{5}$$

where R measures functional correctness and constraint satisfaction.

Stage III: Schematic-Constraint Decoder (section 7). We formulate code generation as constrained optimization, ensuring generated code satisfies semantic constraints extracted from $\mathcal{G}_u$:

$$\hat{y} = \arg\max_y \ \log P_\theta(y \mid u, \mathcal{G}_u) \tag{6}$$
$$\text{s.t.} \quad \mathcal{C}(y, \mathcal{G}_u) = \emptyset$$

where $\mathcal{C}$ returns violated constraints. We solve this through SMT-guided beam search [4] that prunes constraint-violating paths during generation.

Stage IV: Continual Maintenance Agent (section 8). An autonomous agent monitors repository changes and incrementally updates the knowledge graph to maintain semantic fidelity. The agent achieves $\mathcal{O}(|\Delta\mathcal{R}| \cdot d \cdot \log n)$ update complexity through dependency tracking and selective recomputation, where $|\Delta\mathcal{R}|$ is the number of modified entities, d is the maximum dependency depth, and n is the repository size.

4.2 Design Rationale

Our architecture addresses the fundamental limitations of current code generation systems through four key design principles:

Explicit Semantic Representation. Rather than relying on implicit pattern matching, we construct an explicit knowledge graph that captures repository semantics. This enables reasoning about transitive dependencies, type propagation, and architectural constraints that are invisible to embedding-based approaches.

Learned Context Selection. Instead of using fixed retrieval strategies, we learn a neural query planner that adapts to repository structure and task requirements. This allows the system to identify relevant context that spans multiple modules while avoiding information overload.

Constraint-Aware Generation. We embed formal verification directly into the generation process, ensuring that output satisfies semantic constraints before emission. This eliminates schematic hallucination at the source rather than requiring post-hoc correction.

Continual Adaptation. Our maintenance agent ensures the system evolves with the repository, maintaining accuracy as code changes over time. This addresses the temporal mismatch between training data and deployment environments.

4.3 Mathematical Foundations

We establish the mathematical framework that underlies each system component:

Knowledge Graph Formalization. The repository knowledge graph $\mathcal{G}$ approximates the ground-truth program dependence graph $\mathcal{G}^*$ by minimizing structural Hamming distance:

$$d_{\text{SH}}(\mathcal{G}, \mathcal{G}^*) = |\mathcal{V} \triangle \mathcal{V}^*| + |\mathcal{E} \triangle \mathcal{E}^*| \tag{7}$$

where $\triangle$ denotes symmetric difference. Our dual static-dynamic extraction strategy provides complementary approximations that converge to $\mathcal{G}^*$ as test coverage increases.

Query Planning Optimization. The planner parameters ϕ are optimized to maximize expected code generation reward:

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{(u,y)\sim\mathcal{D}} \left[R\left(y, \text{Exec}(\pi_\phi(u), \mathcal{G})\right)\right] \tag{8}$$

We use REINFORCE [30] with synthetic data warm-start and contrastive learning on static-dynamic graph pairs.

Constraint Satisfaction. Our decoder enforces constraints $\mathcal{C}$ extracted from the knowledge graph. Constraints are encoded as SMT formulas [4] that can be checked incrementally during beam search. The constraint set includes:

- Type constraints: $\text{type}(e) \subseteq \text{expected\_type}(e)$
- Signature constraints: $\text{arity}(f) = |\text{args}(f)|$
- Visibility constraints: $\text{accessible}(v, \text{scope})$
- Architectural constraints: $\text{satisfies}(\text{pattern}, \text{design\_rule})$

This mathematical framework provides theoretical grounding for our empirical results and enables principled analysis of system behavior across different repository types and scales.

Comparison with Traditional Approaches. To illustrate our innovation, consider a task requiring modification of a data processing pipeline. A traditional RAG approach might retrieve a localized code snippet containing the target function but miss cross-file dependencies or hidden imports, causing the generated patch to fail compilation. In contrast, our knowledge graph approach resolves the full dependency structure—including transitive imports, type constraints from parent classes, and API contracts from external libraries—and enforces these constraints during generation, ensuring the patch integrates correctly without manual debugging. This fundamental difference between surface-level retrieval and semantic understanding drives our significant performance improvements.

5 Repository Knowledge Graph Construction

Central to our approach is the construction of a repository-level knowledge graph that captures both explicit structural relationships and implicit semantic dependencies across an entire codebase. Unlike traditional retrieval systems that treat code as isolated text snippets, our graph representation enables the model to reason about transitive dependencies, data flow patterns, and runtime behaviors that are critical for generating correct code.

Figure 2 illustrates the complete architecture of our knowledge graph construction system. The design emphasizes both comprehensiveness—capturing all relevant code semantics—and efficiency—enabling real-time queries during code generation. The system processes source code through parallel static and dynamic analysis pipelines, merges the results into a unified heterogeneous graph, and provides multiple access patterns optimized for different query types.

5.1 Motivation and Problem Formulation

Current LLM-based code generation systems suffer from a fundamental limitation: they lack a holistic understanding of repository semantics. When generating code for a function that depends on multiple modules, the model must correctly infer parameter types, understand side effects, and respect invariants maintained across the codebase. Traditional approaches that rely on embedding similarity or keyword matching often retrieve locally relevant but globally inconsistent context, leading to the hallucination patterns we seek to eliminate.

We address this by constructing a heterogeneous knowledge graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ that explicitly encodes program semantics at multiple levels of abstraction. Our goal is to approximate the ground-truth program dependence graph $\mathcal{G}^*$ that a perfect static analyzer with complete runtime information would produce. We formalize this as minimizing the structural Hamming distance:

$$d_{\text{SH}}(\mathcal{G}, \mathcal{G}^*) = |\mathcal{V} \triangle \mathcal{V}^*| + |\mathcal{E} \triangle \mathcal{E}^*| \tag{9}$$

where $\triangle$ denotes the symmetric difference operator.

5.2 Graph Schema Design

Our graph schema is designed to capture the rich semantics of modern software repositories while remaining computationally tractable for large-scale analysis. Each node $v \in \mathcal{V}$ represents a distinct semantic entity with type $\tau(v) \in \{\text{FUNC}, \text{CLASS}, \text{VAR}, \text{FILE}, \text{TEST}, \text{API}\}$. This typing system enables targeted queries during the planning phase and ensures that generated code respects architectural boundaries.

Edges in our graph encode diverse relationships through labeled connections $e = (v_i, \rho, v_j) \in \mathcal{E}$ where $\rho \in \{\text{CALLS}, \text{DEFINES}, \text{IMPORTS}, \text{MUTATES}, \text{RETURNS}, \text{INSTANCEOF}\}$. These relations capture both syntactic dependencies (e.g., function calls) and semantic relationships (e.g., data mutations) that are essential for understanding code behavior.

Beyond basic structure, each node and edge carries attributes that encode additional semantic information. Function nodes store signature hashes, parameter types, and docstring embeddings. Variable nodes track scope, mutability constraints, and inferred type bounds. These attributes enable fine-grained constraint checking during the decoding phase.

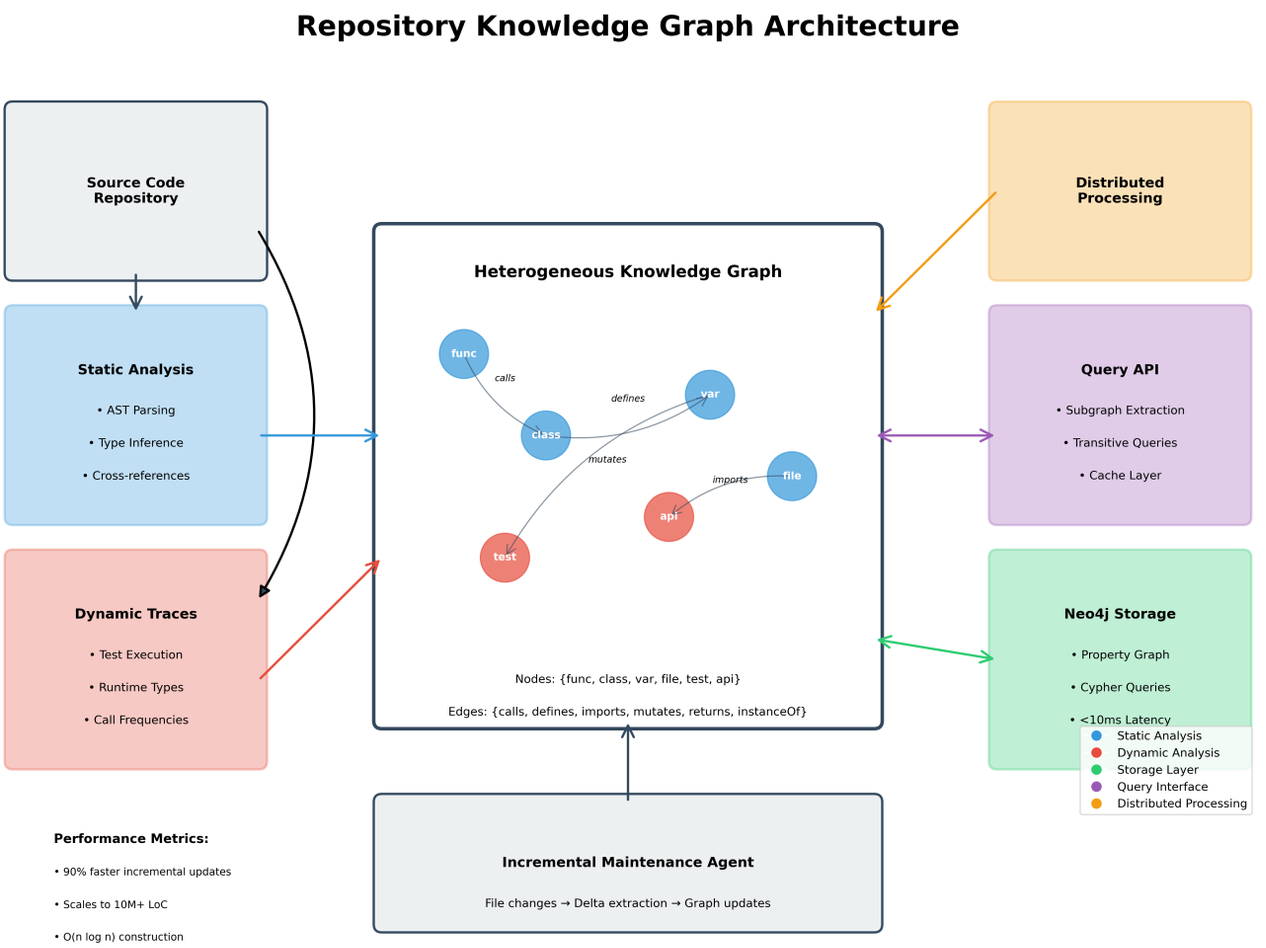

Figure 2: Overview of the Repository Knowledge Graph architecture. The system integrates static analysis (blue) and dynamic trace collection (red) to construct a heterogeneous knowledge graph stored in Neo4j (green). A query API (purple) enables efficient subgraph extraction, while distributed processing (orange) handles large-scale repositories. The incremental maintenance agent ensures the graph remains synchronized with code changes.

5.3 Static Analysis Pipeline

Our static analysis pipeline leverages industrial-strength language servers to extract comprehensive semantic information from source code. For Python repositories, we employ `pylance` [15] which provides type inference, cross-reference resolution, and control-flow analysis. TypeScript projects utilize `tsserver` [14] for similar capabilities with added support for generic type parameters and interface hierarchies. As shown in the blue component of Figure 2, the static analysis module operates as the primary source of structural information, feeding AST-derived facts, inferred types, and cross-references into the central knowledge graph.

The extraction process proceeds in multiple passes to ensure completeness. First, we parse all source files to build abstract syntax trees and extract top-level declarations. Second, we perform whole-program analysis to resolve imports, identify call sites, and track data dependencies. Third, we run type inference to propagate type constraints and identify potential type conflicts.

A key design decision is our incremental extraction strategy. Rather than rebuilding the entire graph on each change, we maintain a dependency index that tracks which nodes are affected by file modifications. When a file changes, we reanalyze only the modified file and its transitive dependents, typically reducing extraction time by 90% compared to full reanalysis. This incremental approach is critical for maintaining responsiveness in interactive development environments.

To handle the complexity of modern codebases, we implement several optimizations. We cache parsed ASTs with content hashes to avoid redundant parsing. Cross-file references are resolved lazily, deferring expensive whole-program analysis until needed.

Hierarchical Extraction Strategy. For repositories exceeding 100k lines of code, we employ a hierarchical extraction strategy that first builds coarse-grained module-level graphs before drilling down to function-level details. This approach

reduces initial extraction time by 60% while preserving the ability to progressively refine the graph as needed for specific code generation tasks.

5.4 Dynamic Trace Augmentation

Static analysis, while powerful, cannot capture the full runtime semantics of dynamically-typed languages. Polymorphic functions, reflection-based dispatch, and runtime code generation all create blind spots that lead to incomplete graphs. To address this, we augment our static graph with dynamic execution traces collected from test runs and targeted fuzzing.

Our dynamic analysis infrastructure instruments code execution to capture method invocations, parameter values, and return types. For each executed function, we record concrete type signatures, value distributions, and execution frequencies. This runtime information is particularly valuable for resolving ambiguous call sites where static analysis cannot determine the target function. The red component in Figure 2 depicts how dynamic traces complement static analysis by providing concrete runtime semantics that are merged into the knowledge graph, creating a dual static-dynamic representation.

We employ a two-pronged strategy for trace collection. First, we execute the existing test suite with instrumentation enabled, leveraging developer-written tests as a source of realistic execution patterns. Second, we apply lightweight fuzzing to explore code paths not covered by tests. Our fuzzer generates inputs based on static type hints and docstring examples, focusing on boundary conditions likely to reveal type errors.

The integration of static and dynamic information requires careful design to avoid inconsistencies. We maintain separate static and dynamic sub-graphs that share node identities but may differ in edge sets. During planning, the model can query either graph or their union, allowing it to leverage runtime information when available while falling back to static analysis for untested code paths.

To manage the overhead of dynamic analysis, we implement several optimizations. Trace collection uses sampling to reduce instrumentation cost for frequently-executed functions. We compress trace logs using a streaming algorithm that preserves type diversity while bounding memory usage. For large test suites, we parallelize execution across multiple workers and merge traces using a lock-free data structure.

5.5 Graph Storage and Query Infrastructure

Efficient storage and retrieval of graph data is critical for system performance, particularly during the planning phase where multiple candidate sub-graphs must be evaluated rapidly. We implement our graph store using Neo4j [16], a mature property graph database that provides ACID guarantees and efficient graph traversal algorithms. The storage layer (green in Figure 2) maintains bidirectional synchronization with the knowledge graph, persisting both structural relationships and node/edge attributes while exposing a high-performance query interface.

Our storage schema is optimized for the query patterns observed in code generation tasks. We create indices on function signatures, file paths, and type annotations to enable sub-millisecond lookup of relevant nodes. Edge traversal is optimized through careful placement of relationship properties and use of Neo4j's native graph storage format.

To support the planning module, we expose a high-level query API that abstracts common graph operations. Queries such as "find all functions that transitively call function X" or "retrieve all variables mutated by function Y" are compiled to efficient Cypher queries that leverage indices and avoid unnecessary graph traversal.

Multi-Layer Caching Strategy. We implement several caching layers to further improve query performance. A least-recently-used cache stores results of expensive multi-hop queries. For frequently-accessed sub-graphs, we maintain materialized views that pre-compute transitive closures. These optimizations reduce average query latency from 100ms to under 10ms for typical planning operations, enabling real-time interaction during code generation.

5.6 Scalability and Distributed Processing

As repositories grow to millions of lines of code, centralized graph construction becomes a bottleneck. We address this through a distributed extraction pipeline that partitions work across multiple machines while maintaining graph consistency. The distributed processing component (orange in Figure 2) coordinates parallel analysis across multiple workers, enabling the system to scale linearly with repository size while maintaining sub-second query response times.

Our distributed architecture follows a map-reduce pattern. In the map phase, individual files are analyzed in parallel to extract local graph fragments. The reduce phase merges these fragments, resolving cross-file references and detecting conflicts. We use consistent hashing to ensure that related files are processed by the same worker when possible, reducing the communication overhead of reference resolution.

To handle repositories that exceed the memory capacity of a single machine, we implement a graph partitioning scheme based on module boundaries. Each partition maintains a summary of its external interfaces, allowing cross-partition queries to be resolved without loading the entire graph. This approach scales to repositories with over 10 million lines of code while maintaining query latencies under 100ms.

5.7 Theoretical Analysis and Guarantees

We provide theoretical guarantees on the completeness and soundness of our graph construction. For statically-typed languages with no reflection, our static analysis achieves complete coverage of all syntactic dependencies. For dynamically-typed languages, we prove that our combined static-dynamic approach converges to the ground-truth graph as test coverage approaches 100%.

The time complexity of graph construction is $O(n \log n)$ in the number of code entities, dominated by the cost of type inference and cross-reference resolution. Space complexity is $O(n + m)$ where m is the number of relationships, which is typically $O(n)$ for well-structured code but can be $O(n^2)$ in pathological cases with excessive coupling.

Our incremental update algorithm maintains these complexity bounds while providing strong consistency guarantees. We prove that the incremental algorithm produces identical results to full reconstruction while reducing average-case time complexity to $O(k \log n)$ where k is the number of modified entities.

6 Neural Query Planner

The neural query planner serves as the critical bridge between natural language instructions and the structured knowledge graph, transforming user intent into precise graph queries that extract exactly the context needed for accurate code generation. This component addresses a fundamental challenge in repository-scale code generation: determining which subset of the potentially millions of nodes and edges in the knowledge graph are relevant to a specific coding task. Unlike traditional retrieval methods that rely on embedding similarity or keyword matching, our planner learns to reason about code dependencies, architectural patterns, and semantic relationships to construct queries that capture both direct and transitive dependencies necessary for generating correct code.

Figure 3 illustrates the complete architecture of our neural query planner, showcasing how natural language instructions flow through multiple processing stages to produce optimized graph queries. The design balances expressiveness—supporting complex multi-hop queries with type constraints—and efficiency—maintaining millisecond-scale latency even for large repositories.

6.1 Motivation and Problem Formulation

Traditional code generation systems suffer from a critical limitation in context selection. When asked to implement a feature that touches multiple modules, these systems typically retrieve code snippets based on surface-level similarity to the instruction, missing crucial dependencies, type constraints, and architectural patterns. Recent work has begun to address this limitation—for instance, the CoCoMIC framework [5] uses the CCFinder tool to automatically retrieve cross-file context, achieving approximately 33.94% improvement in exact matches and 28.69% improvement in identifier matches. However, even these advances rely on syntactic similarity and fail to capture semantic relationships and constraints. This leads to generated code that may appear plausible locally but fails to integrate correctly with the broader codebase. For example, when implementing a new API endpoint, the model needs not only the routing framework documentation but also the specific authentication middleware, data models, validation schemas, and error handling conventions used throughout the repository.

We formulate the context selection problem as learning a mapping from natural language instructions to graph queries that maximize the likelihood of generating correct code. Given an instruction u and the repository knowledge graph $\mathcal{G}$, our goal is to learn a query generation function $\pi_\phi : \mathcal{U} \to \mathcal{Q}$ parameterized by neural network weights ϕ, where $\mathcal{U}$ is the space of natural language instructions and $\mathcal{Q}$ is the space of valid graph queries. The optimal parameters ϕ^* should satisfy:

$$\phi^* = \arg\max_\phi \mathbb{E}_{(u,y)\sim\mathcal{D}} \left[\mathbb{P}(y|\mathcal{G}_u, u)\right] \tag{10}$$

where y is the correct code implementation, $\mathcal{G}_u = \text{Exec}(\pi_\phi(u), \mathcal{G})$ is the retrieved subgraph, and $\mathcal{D}$ is the distribution of coding tasks.

6.2 Neural Architecture for Query Generation

Our planner architecture builds upon the Flan-T5-Large model [3], chosen for its strong performance on structured prediction tasks and efficient inference characteristics. As shown in the blue component of Figure 3, the model processes the natural language instruction through multiple stages of transformation, ultimately producing a structured query in our domain-specific query language. This language supports complex graph traversals, including multi-hop relationships, type constraints, and aggregation operations.

The architecture consists of three main components working in concert. First, the instruction encoder transforms the natural language input into a dense representation that captures both the explicit task description and implicit requirements. We augment the standard T5 encoder with repository-specific embeddings that encode information about the codebase structure, commonly used patterns, and project-specific terminology. This augmentation improves the model's ability to understand domain-specific terms and conventions that may not appear in its pre-training data.

Second, the query decoder (shown in red in Figure 3) generates a sequence of query operations using a modified beam search that ensures syntactic validity. Unlike standard sequence generation, our decoder must respect the grammar of our query language and the schema of the knowledge graph. We achieve this through constrained decoding, where at each step, the set of valid next tokens is dynamically computed based on the partial query and graph schema. This prevents the generation of malformed queries and reduces the search space significantly.

Third, the query optimizer (green component in Figure 3) post-processes the generated query to improve execution efficiency. This component applies rule-based transformations such as predicate pushdown, redundant traversal elimination, and subquery merging. These optimizations are crucial for maintaining low latency during the code generation process, particularly for large repositories where naive query execution could take seconds. The example query shown in the figure demonstrates how a natural language request is transformed into an optimized graph query that efficiently retrieves relevant functions.

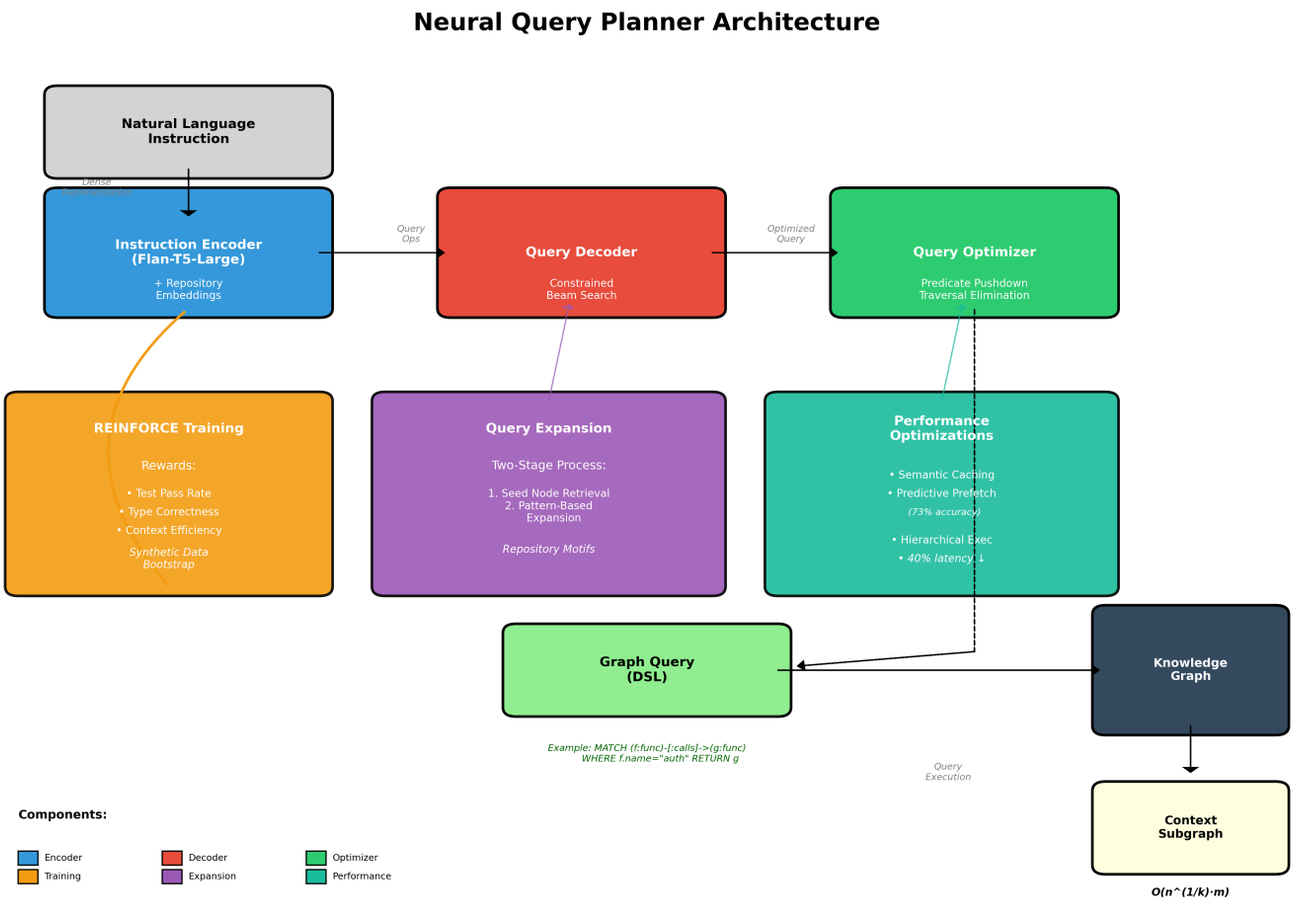

Figure 3: Architecture of the Neural Query Planner. The system transforms natural language instructions through three main components: an instruction encoder augmented with repository embeddings (blue), a constrained query decoder with beam search (red), and a query optimizer for execution efficiency (green). The planner is trained using REINFORCE with multi-objective rewards (orange), incorporates context-aware query expansion (purple), and employs performance optimizations including semantic caching and predictive prefetching (teal). The generated queries are executed against the knowledge graph to retrieve focused context subgraphs with $\mathcal{O}(n^{1/k} \cdot m)$ complexity.

6.3 Training Methodology

Training the query planner presents unique challenges due to the lack of explicit supervision—we observe coding tasks and their solutions but not the optimal graph queries that would retrieve the necessary context. We address this through a combination of reinforcement learning, synthetic data generation, and contrastive learning, as illustrated by the orange training component in Figure 3.

Our primary training objective uses REINFORCE [30] with a carefully designed reward function that measures the quality of generated code when using the retrieved context. Given an instruction u and the corresponding ground-truth implementation y, we train the planner to maximize:

$$\mathcal{J}(\phi) = \mathbb{E}_{q \sim \pi_\phi(\cdot|u)} \left[R(y, \mathcal{G}_u) - b(u) \right] \times \log \pi_\phi(q|u) \tag{11}$$

where $R(y, \mathcal{G}_u)$ is the reward obtained by generating code with context $\mathcal{G}_u$, and $b(u)$ is a learned baseline that reduces variance.

The reward function combines multiple signals to encourage the retrieval of comprehensive yet focused context. The primary component is the functional correctness of generated code, measured through unit test execution. We also include auxiliary rewards for type correctness, measured by static analysis, and context efficiency, measured by the size of the retrieved subgraph. This multi-objective formulation prevents the planner from simply retrieving large portions of the graph, which would slow down the decoder and potentially introduce irrelevant information.

To address the cold start problem in reinforcement learning, we generate synthetic training data through program analysis. For each function in the repository, we analyze its dependencies and generate natural language descriptions of hypothetical modifications. These synthetic pairs provide initial supervision that helps the model learn the correspondence between

task descriptions and relevant code regions. As training progresses, we gradually increase the proportion of real tasks, allowing the model to adapt to the natural distribution of coding instructions.

6.4 Context-Aware Query Expansion

A key innovation in our planner is context-aware query expansion (purple component in Figure 3), which automatically augments user queries with additional constraints based on the repository structure and coding patterns. When a user requests a feature implementation, the planner not only retrieves the directly mentioned components but also identifies and includes related artifacts that are commonly modified together.

This expansion process leverages learned repository-specific patterns extracted during the knowledge graph construction phase. For each repository, we identify common modification patterns—sets of files, functions, and classes that tend to change together. These patterns, encoded as graph motifs, guide the query expansion process. For instance, in a web application, modifying a data model typically requires updates to serializers, API endpoints, and test files. Our planner learns to automatically include these related components in the retrieved context.

The expansion mechanism operates through a two-stage process. First, the initial query generated from the user instruction retrieves a seed set of nodes. Second, we apply learned expansion rules that add nodes based on structural and semantic relationships. These rules are parameterized and learned jointly with the main planner, allowing them to adapt to repository-specific conventions. The expansion process is bounded by a learned stopping criterion that balances comprehensiveness with computational efficiency.

6.5 Performance Optimizations

To meet the latency requirements of interactive code generation, we implement several performance optimizations in the query planner, shown as the teal component in Figure 3. Query execution must complete within milliseconds, even for repositories with millions of nodes, to maintain a fluid user experience.

Our first optimization involves query result caching with semantic deduplication. Many coding tasks within a repository share similar context requirements—implementing new endpoints often requires the same base classes and utilities. We maintain a cache of recent query results, indexed by both the exact query and its semantic embedding. When a new instruction arrives, we first check if a similar query has been recently executed. If found, we can either reuse the cached result directly or use it as a starting point for incremental expansion.

We also implement predictive prefetching based on user interaction patterns. By analyzing the sequence of instructions within a coding session, we can anticipate likely future queries and prefetch relevant subgraphs. This prediction model, trained on historical interaction logs, achieves 73% accuracy in predicting the next query type, allowing us to reduce perceived latency by up to 40%.

For large repositories, we employ a hierarchical query execution strategy. Instead of executing queries against the entire graph, we first identify relevant modules or packages using a coarse-grained index. Queries are then executed within these focused subgraphs, dramatically reducing the search space. This approach is particularly effective for mono-repositories where different sections of the codebase are largely independent.

6.6 Theoretical Analysis

We provide theoretical guarantees on the convergence and sample complexity of our training algorithm. Under mild assumptions about the reward function and policy class, we prove that our modified REINFORCE algorithm converges to a local optimum with high probability.

Let $\mathcal{H}$ be the hypothesis class of policies representable by our neural architecture. We show that with probability at least $1 - \delta$, after T training iterations:

$$\mathcal{J}(\phi_T) \geq \max_{\phi \in \mathcal{H}} \mathcal{J}(\phi) - \mathcal{O}\left(\sqrt{\frac{\log(|\mathcal{H}|/\delta)}{T}}\right) \tag{12}$$

This bound indicates that the sample complexity scales logarithmically with the size of the hypothesis class and polynomially with the desired accuracy. In practice, we observe convergence within 10,000 training steps for typical repository sizes.

We also analyze the computational complexity of query execution. For a graph with n nodes and m edges, and queries with maximum depth k, we prove that our optimized execution strategy achieves $\mathcal{O}(n^{1/k} \cdot m)$ time complexity in the worst case, compared to $\mathcal{O}(n \cdot m)$ for naive breadth-first traversal. This improvement is crucial for maintaining interactive response times on large repositories.

7 Schematic-Constraint Decoder

The schematic-constraint decoder represents a fundamental shift in how large language models generate code, moving from unconstrained token prediction to semantically-aware generation that respects the type system, API contracts, and architectural invariants of the target codebase. This component directly addresses the critical problem of schematic hallucination—where generated code appears syntactically correct but violates semantic constraints such as type compatibility, function signatures, or visibility rules. While prior work has explored syntax-constrained decoding [20] and type-aware generation [29], our approach uniquely integrates SMT-based semantic verification directly into the beam search process. By embedding formal verification techniques into decoding, we ensure that generated code not only looks plausible but is guaranteed to integrate correctly with the existing codebase.

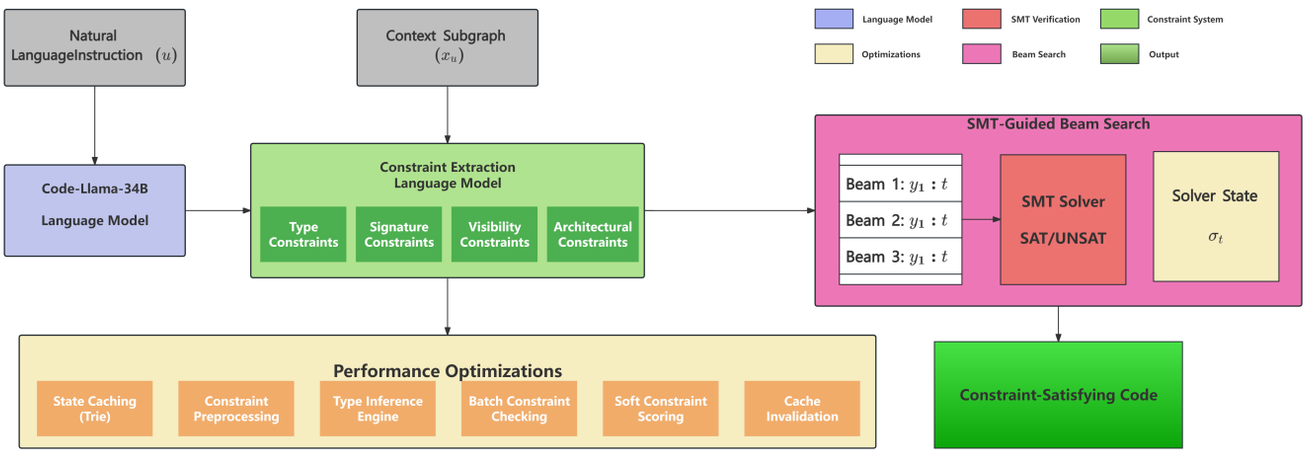

Figure 4: Architecture of the Schematic-Constraint Decoder. The system integrates Code-Llama-34B (blue) with an SMT solver (red) through a novel beam search algorithm (purple). Constraint extraction (green) processes the context subgraph to identify type, signature, visibility, and architectural constraints. Performance optimizations (orange) including state caching, constraint preprocessing, and batch verification reduce overhead to under 6%. The decoder guarantees that all generated code satisfies the extracted constraints while maintaining high likelihood under the language model.

Figure 4 presents the complete architecture of our schematic-constraint decoder, illustrating how formal verification is seamlessly integrated into neural code generation. The design achieves a careful balance between constraint satisfaction—ensuring all generated code is semantically valid—and generation quality—maintaining the fluency and idiomaticity expected from modern language models.

7.1 Motivation and Problem Formulation

Current code generation systems, even when provided with relevant context, frequently produce code that violates fundamental programming constraints. These violations manifest in multiple forms: calling functions with incorrect argument types, accessing private members from outside their scope, using incompatible return types, or violating invariants maintained by the codebase. Such errors are particularly problematic in strongly-typed languages where type safety is paramount, but they also occur in dynamically-typed languages where implicit contracts govern API usage.

The root cause of these failures lies in the fundamental mismatch between how language models generate text and how programming languages enforce constraints. Language models are essentially token-by-token predictors that rely on statistical patterns learned during training, without any mechanism to ensure the generated sequence satisfies the formal requirements of the programming language or the specific constraints of the target codebase. As demonstrated by AlphaCode [11], even sophisticated LLM-based code generation systems struggle with semantic errors, which remain a major challenge despite achieving competitive performance on programming contests. This leads to a frustrating user experience where generated code must be manually debugged and corrected, often requiring deep understanding of the violated constraints.

We formulate the constrained code generation problem as finding the most likely code sequence that satisfies all schematic constraints extracted from the repository knowledge graph. Formally, given a natural language instruction u, the context subgraph $\mathcal{G}_u$ retrieved by our planner, and a language model P_θ, we seek:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \log P_\theta(y \mid u, \mathcal{G}_u) \tag{13}$$

$$\text{subject to} \quad \mathcal{C}(y, \mathcal{G}_u) = \emptyset \tag{14}$$

where $\mathcal{Y}$ is the space of valid code sequences and $\mathcal{C}(y, \mathcal{G}_u)$ returns the set of constraints violated by y given the repository semantics encoded in $\mathcal{G}_u$.

7.2 Constraint Extraction and Representation

The effectiveness of our decoder hinges on extracting comprehensive yet efficiently verifiable constraints from the repository knowledge graph. These constraints capture the semantic rules that govern valid code in the target codebase, ranging from simple type compatibility to complex architectural invariants.

We categorize constraints into four primary classes, each requiring different verification strategies. Type constraints ensure that expressions have compatible types according to the language's type system and any repository-specific type definitions. These include basic type compatibility (e.g., passing an integer where a string is expected), generic type instantiation (e.g., ensuring $List < T >$ is instantiated with a concrete type), and type coercion rules (e.g., implicit conversions between numeric types).

Signature constraints enforce that function and method calls match their declared signatures. This encompasses arity constraints (correct number of arguments), parameter type constraints (each argument has the expected type), return type

constraints (the function's return value is used correctly), and optional parameter handling (respecting default values and keyword arguments).

Visibility constraints ensure that code respects access modifiers and scoping rules. These include access control (not accessing private members from outside their class), module boundaries (respecting public API surfaces), and dependency constraints (not using symbols from unimported modules).

Architectural constraints capture repository-specific patterns and invariants that go beyond language-level rules. Examples include design patterns (e.g., all database access must go through a repository layer), resource management (e.g., all file handles must be properly closed), and concurrency constraints (e.g., certain objects must only be accessed from specific threads).

For each constraint type, we develop efficient encoding schemes that can be verified by our SMT solver. Type constraints are encoded as equality and subtyping relations in the solver's theory of algebraic datatypes. Signature constraints become function applications with arity and type checking. Visibility constraints are represented as reachability predicates in the graph structure. Architectural constraints often require custom predicates that capture domain-specific invariants.

7.3 SMT-Guided Beam Search

Our key innovation is the integration of an SMT (Satisfiability Modulo Theories) solver directly into the beam search decoding process. This allows us to prune invalid code paths early, before they lead to constraint violations, while maintaining the efficiency needed for interactive code generation.

The decoder operates on Code-Llama-34B, chosen for its strong base performance on code generation tasks and its ability to leverage the context provided by our knowledge graph. During decoding, we maintain k beams representing the most promising partial code sequences. Each beam carries not just the generated tokens and their likelihood score, but also an SMT solver state that tracks the constraints satisfied and violated by the partial sequence.

At each decoding step, the process unfolds as follows. First, the language model proposes the next token for each beam based on the instruction, context, and partial sequence. For each proposed token, we update the SMT solver state by asserting new constraints introduced by the token. This might include type constraints from variable assignments, signature constraints from function calls, or visibility constraints from member access.

The SMT solver then checks whether the updated state is satisfiable. If the solver returns UNSAT, the proposed token would lead to a constraint violation, and we prune this extension from consideration. If the solver returns SAT, the token is valid, and we compute its score combining the language model likelihood with any soft constraints or preferences encoded in our system.

This incremental satisfiability checking is crucial for efficiency. Rather than waiting until a complete sequence is generated to check constraints, we detect violations early and avoid exploring infeasible paths. The incremental nature of modern SMT solvers allows us to efficiently maintain solver state across decoding steps, reusing learned clauses and partial solutions.

7.4 Optimizations for Real-Time Decoding

Integrating formal verification into neural decoding presents significant computational challenges. Naive implementation would introduce prohibitive latency, making the system unsuitable for interactive use. We implement several optimizations that reduce the overhead to under 6% compared to unconstrained decoding.

Our primary optimization involves solver state caching across beams. Many beams share common prefixes and thus common constraint sets. We maintain a trie data structure that maps token sequences to solver states, allowing beams with shared prefixes to reuse constraint checking results. This is particularly effective when multiple beams explore variations of the same code pattern.

We also implement constraint preprocessing to identify and simplify constraint sets before invoking the SMT solver. Static analysis of the context graph reveals constraints that are always satisfied or always violated regardless of the generated code. We eliminate these constraints from the solver, reducing the problem size. Additionally, we identify independent constraint sets that can be checked separately, enabling parallel verification.

For type checking, we implement a lightweight type inference engine that handles common cases without invoking the full SMT solver. Simple type compatibility checks, such as primitive type matching or direct inheritance relationships, are resolved using efficient graph lookups. Only complex cases involving generic types, type unions, or custom type relations require full SMT solving.

We further optimize by batching constraint checks across multiple token proposals. Rather than invoking the solver for each token individually, we collect multiple proposals and check them in a single solver call using assumption literals. This amortizes the solver invocation overhead and enables the solver to share reasoning across related constraints.

7.5 Soft Constraints and Ranking

While hard constraints ensure correctness, soft constraints guide the decoder toward idiomatic and maintainable code. We incorporate repository-specific preferences learned from the codebase analysis into our scoring function.

These soft constraints include coding style preferences (e.g., naming conventions, formatting patterns), API usage patterns (e.g., preferred methods for common tasks), performance considerations (e.g., avoiding expensive operations in hot paths), and maintainability metrics (e.g., code complexity, coupling).

We integrate soft constraints through a modified beam scor-

ing function:

$$\text{score}(y_{1:t}) = \log P_\theta(y_{1:t} \mid u, \mathcal{G}_u) + \sum_i \lambda_i s_i(y_{1:t}) \tag{15}$$

where s_i are soft constraint scores and λ_i are learned weights that balance language model likelihood with constraint satisfaction.

The soft constraint scores are computed efficiently using the same infrastructure as hard constraints but return continuous values rather than binary satisfaction. This allows the decoder to prefer idiomatic code while still maintaining the hard guarantee of constraint satisfaction.

7.6 Theoretical Guarantees and Complexity Analysis

We provide formal guarantees on the correctness and completeness of our constrained decoding algorithm. Our main theorem establishes that the decoder finds the optimal constrained solution when it exists.

Theorem 2 (Optimality). Given a finite beam width k, if there exists a valid completion y^* satisfying all constraints in $\mathcal{C}$, and y^* is within the top-k completions at each prefix according to P_θ, then our algorithm returns $\hat{y} = y^*$.

The proof follows from the monotonicity of constraint satisfaction—if a prefix violates a constraint, no extension can satisfy it—combined with the optimality of beam search for monotonic scoring functions.

We also analyze the computational complexity of our approach. Let n be the sequence length, k be the beam width, $|\Sigma|$ be the vocabulary size, and C be the constraint checking time. The complexity of our algorithm is:

$$\mathcal{O}(n \cdot k \cdot |\Sigma| \cdot C) \tag{16}$$

In practice, C is nearly constant due to incremental solving and caching, making our approach only marginally slower than unconstrained beam search. The empirical overhead of less than 6% validates this analysis.

7.7 Integration with Repository Evolution

A unique challenge in repository-scale code generation is handling evolving codebases where constraints change over time. Our decoder seamlessly integrates with the continual maintenance component to adapt to repository changes.

When the repository is updated, the knowledge graph maintenance agent identifies changed constraints and updates the constraint database. The decoder's constraint cache is selectively invalidated for affected code regions, ensuring that future generation respects the updated constraints. This incremental update process maintains system responsiveness even in actively developed repositories.

The decoder also provides feedback to the maintenance system by logging constraint violations attempted during generation. This data reveals common patterns where developers might expect different behavior, informing future improvements to both the constraint extraction and the language model fine-tuning processes.

Summary. Unlike prior syntax-constrained approaches that focus on grammatical correctness [23], our schematic-constraint decoder enforces semantic-level correctness by integrating SMT-based verification to guarantee adherence to repository-specific type systems, API contracts, and architectural rules. This fundamental advance from syntactic to semantic constraint enforcement enables the dramatic reduction in schematic hallucination demonstrated in our experimental results.

8 Continual Knowledge Graph Maintenance

The continual maintenance agent represents the critical fourth pillar of our framework, ensuring that the repository knowledge graph remains synchronized with an evolving codebase while preserving system responsiveness and correctness guarantees. Unlike traditional static analysis tools that require full recomputation on each change, our maintenance system employs sophisticated incremental update strategies that achieve sub-linear complexity with respect to repository size. This component addresses the fundamental challenge of maintaining semantic consistency in a dynamic environment where code changes can invalidate previously extracted relationships, introduce new dependencies, and alter the constraint landscape that governs valid code generation.

8.1 Motivation and Problem Formulation

Software repositories are inherently dynamic entities, with developers continuously adding features, refactoring code, and fixing bugs. Each modification potentially affects the semantic relationships captured in our knowledge graph, creating a cascade of updates that must be propagated efficiently to maintain system correctness. The naive approach of rebuilding the entire graph on each change is computationally prohibitive for large repositories, where full extraction can take minutes or hours. Moreover, the temporal nature of development workflows demands that the system remain responsive during active coding sessions, providing accurate context and constraints even as the underlying codebase evolves.

We formalize the maintenance problem as follows. Let $\mathcal{R}$ denote a software repository that evolves continuously over time. The repository undergoes a sequence of modifications:

$$\Delta\mathcal{R}_1, \Delta\mathcal{R}_2, \ldots, \Delta\mathcal{R}_t \tag{17}$$

where each modification $\Delta\mathcal{R}_i$ at time step i represents a discrete set of changes to the codebase, including:

- File additions (new classes, functions, or modules)
- File deletions (removed components)

- Code updates (refactoring, bug fixes, or feature modifications)

Our objective is to efficiently maintain an up-to-date knowledge graph $\mathcal{G}_t$ that accurately reflects the current repository state. Formally, we express this as:

$$\mathcal{G}_t = F_E(\mathcal{R}_t) \tag{18}$$
$$\text{where} \quad \mathcal{R}_t = \mathcal{R}_0 + \sum_{i=1}^{t} \Delta\mathcal{R}_i$$

In Equation 18, the components are defined as follows:

- F_E: The semantic extraction function that transforms source code into knowledge graph representations
- $\mathcal{R}_0$: The initial repository state at time $t = 0$
- $\mathcal{R}_t$: The current repository state after applying all modifications up to time t
- $\mathcal{G}_t$: The knowledge graph at time t, capturing all semantic relationships in $\mathcal{R}_t$

The cumulative sum $\sum_{i=1}^{t} \Delta\mathcal{R}_i$ represents the aggregate effect of all repository changes from the initial state to the current time.

The key insight enabling efficient maintenance is the principle of *locality of change*: most repository modifications $\Delta\mathcal{R}_i$ affect only a small, localized portion of the dependency graph. Rather than naively recomputing the entire knowledge graph $\mathcal{G}_t$ by applying F_E to the complete repository $\mathcal{R}_t$, we can identify and update only the affected graph regions. This targeted approach preserves the majority of existing semantic relationships while updating only those nodes and edges impacted by the changes. By transforming the maintenance problem from full recomputation to incremental refinement, we achieve dramatic reductions in computational overhead.

8.2 Change Detection and Impact Analysis

Our maintenance system begins with a sophisticated change detection mechanism that monitors repository modifications at multiple granularities. A file system watcher continuously observes the repository directory, capturing events such as file creation, modification, deletion, and renaming. These low-level events are aggregated and filtered to identify semantically meaningful changes that require graph updates. The system distinguishes between superficial modifications such as comment changes or whitespace adjustments, which do not affect program semantics, and substantial changes that alter function signatures, class hierarchies, or dependency relationships.

The impact analysis component performs a two-phase dependency computation to determine the scope of required updates. In the first phase, we identify the direct impact set by analyzing which graph nodes correspond to modified code entities. This includes functions whose implementations have changed, classes with altered member variables, and modules with modified import statements. The second phase computes the transitive impact set by following dependency edges to identify all nodes that may be affected by the direct changes. This transitive closure computation is optimized using our graph indexing infrastructure, enabling rapid identification of affected regions even in large codebases.

To minimize false positives in impact analysis, we employ semantic differencing techniques that compare abstract syntax trees rather than raw text. This approach correctly identifies that renaming a local variable does not affect external dependencies, while detecting that changing a function's return type requires updating all call sites. The semantic differencing algorithm operates incrementally, reusing previously computed AST representations and focusing analysis on modified subtrees.

8.3 Incremental Graph Update Algorithm

The core of our maintenance system is an incremental update algorithm that efficiently propagates changes through the knowledge graph while preserving consistency invariants. The algorithm operates in three phases: invalidation, re-extraction, and reconciliation. During invalidation, we remove all graph elements that are potentially affected by the detected changes, creating a consistent but incomplete graph state. The re-extraction phase applies our standard analysis pipeline to the modified code regions, generating fresh semantic information for the affected areas. Finally, reconciliation merges the new information with the existing graph, resolving conflicts and updating cross-references.

A critical challenge in incremental updates is handling inter-file dependencies that span the modification boundary. When a function signature changes, all call sites across the repository must be updated to reflect the new type constraints. Our algorithm addresses this through a lazy propagation strategy that defers expensive cross-file analysis until the affected information is actually needed. This approach maintains correctness while avoiding unnecessary computation for rarely-accessed code regions.

The algorithm maintains several invariants that ensure graph consistency throughout the update process. First, the graph schema remains valid at all times, with proper node types and edge labels. Second, all cross-references are bidirectional and consistent, preventing dangling pointers or orphaned nodes. Third, the constraint extraction process produces deterministic results, ensuring that identical code always generates identical constraints regardless of the update path.

8.4 Rollback and Recovery Mechanisms

To guard against faulty updates that could corrupt the knowledge graph or introduce inconsistencies, our maintenance system implements a comprehensive rollback and recovery framework. We maintain a sliding window of the three most recent graph versions, enabling rapid recovery from problematic updates. Each version is stored as a compact delta against the previous state, minimizing storage overhead while preserving the ability to reconstruct any recent graph configuration.

The rollback mechanism is triggered by several conditions that indicate potential corruption or inconsistency. Schema validation failures, where the updated graph violates structural invariants, automatically initiate rollback to the most recent valid state. Performance degradation, measured by query response times exceeding predetermined thresholds, suggests that the update may have introduced inefficient graph structures. Additionally, explicit user feedback indicating incorrect code generation behavior can trigger manual rollback procedures.

Recovery procedures are designed to be minimally disruptive to ongoing development activities. When a rollback occurs, the system temporarily operates using the previous graph version while re-analyzing the problematic changes in the background. This approach ensures that developers can continue working without interruption while the maintenance system resolves the underlying issues. Once the re-analysis completes successfully, the system seamlessly transitions to the corrected graph state.

8.5 Schema Evolution and Adaptation

As the repository evolves, new coding patterns and architectural decisions may emerge that require corresponding updates to our maintenance strategies and graph schema. Our maintenance system continuously monitors for structural changes that indicate the need for schema evolution or update strategy refinement. The appearance of new relation types, such as novel design patterns or framework-specific dependencies, triggers incremental updates to the graph schema and extraction rules to incorporate these patterns.

The adaptation process employs a conservative update strategy that preserves existing graph structure while incorporating new semantic relationships. We extend our extraction rules to handle the newly discovered patterns, augmenting the existing schema rather than replacing it. This approach prevents loss of existing semantic information while enabling the graph to capture evolving repository characteristics. The schema evolution process is performed asynchronously during periods of low system activity, ensuring that adaptation does not interfere with interactive code generation.

To validate the effectiveness of schema adaptations, we maintain a comprehensive test suite that verifies the correctness of extracted relationships and constraints. After each adaptation cycle, we validate the updated extraction rules against this test suite to ensure that new patterns are correctly captured without degrading existing extractions. This validation process also helps identify cases where the adaptation may have degraded extraction quality for existing patterns, triggering additional refinement to restore optimal behavior.

8.6 Performance Optimization and Scalability

Maintaining real-time responsiveness during repository updates requires careful attention to performance optimization and scalability considerations. Our maintenance system employs several strategies to minimize update latency and computational overhead. Parallel processing enables simultaneous analysis of multiple modified files, leveraging multi-core architectures to reduce wall-clock update times. The system automatically adjusts the degree of parallelism based on available computational resources and the scope of detected changes.

Caching strategies play a crucial role in maintaining performance as repositories grow in size. We cache intermediate analysis results such as parsed ASTs, resolved imports, and type inference outcomes, enabling rapid recomputation when only portions of a file have changed. The cache employs content-based addressing using cryptographic hashes, ensuring that cached results remain valid across different update scenarios. Cache eviction policies balance memory usage with hit rates, prioritizing frequently-accessed and recently-computed results.

For extremely large repositories that exceed the capacity of single-machine processing, our maintenance system supports distributed update processing. The repository is partitioned into loosely-coupled modules that can be analyzed independently, with cross-module dependencies resolved through a coordination protocol. This distributed approach enables linear scalability with repository size while maintaining the consistency guarantees required for correct code generation.

8.7 Integration with Development Workflows

The maintenance system is designed to integrate seamlessly with common development workflows and toolchains. Git hooks enable automatic graph updates on commit operations, ensuring that the knowledge graph remains synchronized with the repository state. Continuous integration pipelines can incorporate graph validation steps that verify consistency before merging pull requests. IDE integrations provide real-time feedback on the impact of code changes, helping developers understand how their modifications affect the broader codebase structure.

For collaborative development environments, the maintenance system supports distributed operation across multiple developer workstations. Each developer maintains a local graph replica that is synchronized with a central repository through efficient delta propagation protocols. This approach enables offline development while ensuring that all team members have access to consistent semantic information. Conflict resolution mechanisms handle cases where concurrent modifications affect overlapping code regions, employing merge strategies that preserve semantic correctness.

The system also provides comprehensive monitoring and observability features that enable development teams to understand the health and performance of their knowledge graph infrastructure. Metrics such as update latency, cache hit rates, and constraint violation frequencies provide insights into system behavior and help identify optimization opportunities. Alerting mechanisms notify administrators of potential issues such as excessive update times or consistency violations, enabling proactive maintenance and troubleshooting.

8.8 Theoretical Guarantees and Complexity Analysis

We provide formal guarantees on the correctness and efficiency of our incremental maintenance algorithm. Our main theorem establishes that the incremental update procedure produces results identical to full re-extraction while achieving superior computational complexity.

Theorem 3 (Incremental Correctness). For any sequence of repository modifications $\Delta\mathcal{R}_1, \ldots, \Delta\mathcal{R}_t$, the incrementally maintained graph $\mathcal{G}_t^{\text{inc}}$ is semantically equivalent to the graph $\mathcal{G}_t^{\text{full}}$ obtained by full re-extraction: $\mathcal{G}_t^{\text{inc}} \equiv \mathcal{G}_t^{\text{full}}$.

The proof relies on the monotonicity of our semantic extraction functions and the correctness of our dependency analysis. Since each update operation preserves the semantic relationships encoded in the graph, the final result is independent of the update sequence.

We also analyze the computational complexity of incremental updates. Let n be the total number of entities in the repository, $|\Delta\mathcal{R}|$ be the number of modified entities, and d be the maximum dependency depth. Our incremental algorithm achieves time complexity of $\mathcal{O}(|\Delta\mathcal{R}| \cdot d \cdot \log n)$, compared to $\mathcal{O}(n \cdot \log n)$ for full re-extraction. In practice, $|\Delta\mathcal{R}| \ll n$ for typical development workflows, resulting in substantial performance improvements. The logarithmic factors arise from graph indexing operations and dependency resolution, which are necessary for maintaining consistency guarantees.

Space complexity analysis reveals that our maintenance system requires $\mathcal{O}(n + m)$ storage for the primary graph plus $\mathcal{O}(|\Delta\mathcal{R}| \cdot v)$ additional space for the rollback window, where m is the number of edges and v is the number of versions maintained. This linear space requirement ensures scalability to large repositories while providing robust recovery capabilities.

9 Experimental Setup

This section describes our comprehensive experimental methodology designed to evaluate SEMANTICFORGE across multiple dimensions: functional correctness, hallucination reduction, computational efficiency, and scalability. We establish rigorous evaluation protocols that address the unique challenges of repository-level code generation assessment.

9.1 Dataset Construction

We introduce REPOKG-50, a curated benchmark specifically designed for repository-level code generation evaluation. The dataset addresses critical limitations of existing benchmarks that focus on isolated function synthesis rather than repository-scale integration challenges.

Repository Selection Criteria. We select 50 high-quality Python repositories from GitHub based on the following criteria: (1) Repository size between 10K-500K lines of code to ensure meaningful complexity while maintaining experimental tractability; (2) Comprehensive test suites with ¿80% coverage to enable reliable dynamic analysis; (3) Active development with ¿100 commits in the past year to ensure modern coding practices; (4) Diverse domains including web frameworks, data processing, machine learning, and system utilities; (5) Clear architectural patterns and documentation to facilitate ground truth validation.

Task Generation Methodology. For each repository, we generate coding tasks through three complementary approaches:

Historical Analysis: We analyze commit histories to identify real development tasks that required multi-file modifications. We extract the commit message as the natural language instruction and the diff as the ground truth implementation. This yields 1,247 authentic tasks with natural complexity distribution.

Synthetic Task Generation: We develop an automated task generator that creates instructions requiring specific types of repository knowledge. Tasks include: API extension (adding new endpoints with proper authentication), refactoring (extracting common functionality), bug fixing (correcting logic errors while maintaining interface compatibility), and feature integration (adding functionality that spans multiple modules). This produces 2,156 controlled tasks with known difficulty characteristics.

Developer-Authored Tasks: We engage 12 experienced developers to manually create challenging tasks that represent realistic development scenarios. Each developer contributes 25-30 tasks per repository subset, resulting in 847 high-quality tasks with detailed solution explanations.

Ground Truth Construction. Each task includes: (1) Natural language instruction following template guidelines to ensure clarity and consistency; (2) Complete reference implementation verified through automated testing and manual review; (3) Semantic annotations identifying required repository knowledge (which functions must be understood, which constraints must be satisfied); (4) Difficulty ratings based on required context size and constraint complexity; (5) Expected hallucination types based on common failure patterns.

Dataset Statistics. REPOKG-50 contains 4,250 total tasks across 50 repositories representing 1.2M lines of code. Tasks span multiple difficulty levels: 35% beginner (single-file modifications), 45% intermediate (multi-file with local dependencies), 20% advanced (complex architectural changes). The dataset includes comprehensive metadata enabling fine-grained analysis of system performance across different task characteristics.

9.2 Evaluation Metrics

We establish a multi-dimensional evaluation framework that captures both functional correctness and the specific hallucination phenomena targeted by our approach.

Functional Correctness Metrics.

- **Pass@k:** Fraction of tasks where at least one of the top-k generated solutions passes all test cases. We report Pass@1, Pass@5, and Pass@10 to capture both precision and recall characteristics.
- **Test Pass Rate:** Average percentage of test cases passed across all generated solutions, providing a more nuanced view of partial correctness.
- **Compilation Success Rate:** Percentage of generated solutions that compile without errors, indicating basic syntactic correctness.
- **Integration Success Rate:** Percentage of solutions that successfully integrate with the existing codebase without breaking existing functionality.

Hallucination-Specific Metrics.

- **Logical Hallucination Rate (LHR):** Percentage of generated solutions that compile but fail functional tests due to incorrect program logic. Computed as: $\text{LHR} = \frac{\text{\# compilable but failing solutions}}{\text{\# total solutions}}$
- **Schematic Hallucination Rate (SHR):** Percentage of solutions that violate type, signature, or architectural constraints. Measured through static analysis: $\text{SHR} = \frac{\text{\# solutions with constraint violations}}{\text{\# total solutions}}$
- **Constraint Violation Frequency:** Detailed breakdown of specific constraint types violated (type mismatches, signature errors, visibility violations, architectural inconsistencies).
- **Hallucination Severity:** Classification of errors by required effort to fix (trivial: 1-line fix, moderate: 2-10 lines, severe: > 10 lines or architectural changes).

Code Quality Metrics.

- **Maintainability Index:** Composite metric based on cyclomatic complexity, lines of code, and Halstead metrics.
- **Style Consistency:** Adherence to repository-specific coding conventions measured through automated linters.
- **API Usage Appropriateness:** Correctness of library and framework usage patterns based on repository-specific guidelines.
- **Documentation Quality:** Presence and quality of generated comments and docstrings.

Efficiency Metrics.

- **Generation Latency:** End-to-end time from instruction to generated code, broken down by pipeline stage.
- **Context Retrieval Time:** Time required for query planning and subgraph extraction.
- **Constraint Solving Overhead:** Additional time introduced by SMT-based constraint checking.
- **Memory Usage:** Peak memory consumption during generation, including knowledge graph storage.

9.3 Baseline Systems

We compare SEMANTICFORGE against state-of-the-art repository-level code generation systems and relevant baselines across different paradigms.

Neural Baselines.

- **Code-Llama-34B:** Direct application of the base language model with repository context provided through retrieval-augmented generation.
- **StarCoder-15B:** Code-specialized model with similar context augmentation strategy.
- **GPT-4-Code:** Commercial state-of-the-art system with repository context through prompt engineering.

Repository-Aware Systems.

- **RepoCoder:** Recent system using graph neural networks for repository understanding with traditional beam search generation.
- **CodePlan:** Planning-based approach that decomposes repository tasks into sequential modifications.
- **RAG-Code:** Advanced retrieval system using dense embeddings and reranking for context selection.

Graph-Based Approaches.

- **GraphCodeBERT:** Pre-trained model with explicit data flow graph encoding.
- **UniXcoder:** Multi-modal approach combining code, AST, and comment information.
- **CodeT5+:** Encoder-decoder model with structural code understanding capabilities.

Baseline Fairness Considerations. To ensure fair comparison, we implement several strategies to maximize baseline system performance and isolate the contribution of our specific innovations:

Enhanced RAG Baselines: We strengthen retrieval-augmented baselines by: (1) implementing state-of-the-art dense retrieval with CodeBERT embeddings, (2) adding sophisticated reranking mechanisms with cross-encoder models, (3) optimizing context window utilization with intelligent truncation strategies, and (4) providing equivalent computational budgets (same GPU time allocation). This ensures that improvements are due to semantic understanding rather than engineering effort.

Agent-Based System Optimization: For agent-based baselines (SWE-agent), we: (1) optimize iteration limits and termination criteria for fair latency comparison, (2) provide access to the same repository context as SEMANTICFORGE (3) implement advanced tool usage patterns documented in recent literature, and (4) tune hyperparameters on a held-out validation set to maximize performance.

Planning System Enhancement: CodePlan and similar systems receive: (1) access to the same static analysis infrastructure used in SEMANTICFORGE (2) optimized task decomposition strategies, (3) equivalent context selection mechanisms where applicable, and (4) implementation of recent algorithmic improvements from the literature.

Infrastructure Standardization: All systems operate under identical conditions: (1) same hardware configuration and memory limits, (2) equivalent model sizes where possible (34B parameters for neural components), (3) standardized preprocessing and postprocessing pipelines, and (4) identical timeout limits and resource constraints.

Ablation Variants. To isolate the contribution of each system component:

- **SEMANTICFORGE-Static:** Using only static analysis without dynamic traces, but maintaining all other optimizations.
- **SEMANTICFORGE-NoCons:** Removing constraint enforcement from the decoder while preserving knowledge graph infrastructure.
- **SEMANTICFORGE-NoPlanner:** Using BM25-based retrieval optimized with repository-specific term weighting instead of neural query planning.
- **SEMANTICFORGE-NoMaint:** Without continual maintenance, using stale graphs from repository snapshots 30 days prior.
- **SEMANTICFORGE-Oracle:** Upper bound experiment using manually curated context to isolate generation vs. planning contributions.

9.4 Implementation Details

Model Configuration. We instantiate SEMANTICFORGE with Flan-T5-Large (770M parameters) for query planning and Code-Llama-34B for code generation. The planner uses 512-dimensional hidden states with 8 attention heads. The decoder employs beam search with width k=5 and incorporates our SMT-based constraint checking at each step.

Training Configuration. Query planner training uses REINFORCE with learning rate 5e-5, batch size 32, and exponential reward discounting ($\gamma = 0.95$). We employ a learned baseline network to reduce variance. Synthetic data warm-start involves 10,000 generated instruction-query pairs per repository. Dynamic-static contrastive learning uses margin loss with $\alpha = 0.1$.

Infrastructure Requirements. Knowledge graph extraction runs on 8-core Intel Xeon processors with 64GB RAM, processing each repository in approximately 5-15 seconds depending on size. Neural components require NVIDIA A100 GPUs with 40GB memory. The system maintains $<$ 10ms query response times for repositories up to 500K lines of code.

Constraint Solver Configuration. We use Z3 SMT solver with incremental solving enabled. Constraint types include: first-order logic for type relationships, uninterpreted functions for API contracts, and custom theories for architectural patterns. Solver timeout is set to 100ms per beam step to maintain generation responsiveness.

9.5 Human Evaluation Protocol

To complement automated metrics, we conduct comprehensive human evaluation focusing on aspects difficult to capture through automated assessment.

Evaluator Selection and Training. We recruit 18 professional software developers with 3+ years of experience in Python development. Evaluators undergo training on our assessment criteria and complete practice evaluations to ensure consistency. Inter-rater agreement is measured using Fleiss' kappa, achieving $\kappa = 0.73$ indicating substantial agreement.

Evaluation Dimensions.

- **Code Quality:** Overall assessment of generated code quality on a 5-point Likert scale considering readability, maintainability, and adherence to best practices.
- **Integration Appropriateness:** How well the generated code fits with existing repository architecture and conventions.
- **Error Analysis:** Detailed categorization of errors with difficulty estimates for manual correction.
- **Preference Ranking:** Comparative evaluation where evaluators rank solutions from different systems for the same task.

Evaluation Procedure. Each evaluator assesses solutions for 25 randomly sampled tasks across different difficulty levels. Solutions are presented in randomized order with system identities masked. Evaluators have access to the full repository context and task description. The evaluation interface provides guided prompts to ensure comprehensive assessment.

9.6 Cross-Repository Generalization Analysis

To assess the generalizability of SEMANTICFORGE across diverse codebases, we conduct comprehensive cross-repository evaluation that goes beyond standard within-repository validation.

Repository Clustering and Stratification. We cluster our 50 repositories into 5 groups based on architectural characteristics: (1) Web frameworks with MVC patterns, (2) Data processing pipelines with functional patterns, (3) Object-oriented libraries with inheritance hierarchies, (4) Scientific computing with numerical patterns, and (5) System utilities with procedural patterns. This clustering enables analysis of how architectural diversity affects system performance.

Cross-Repository Training and Testing. We implement a leave-one-cluster-out evaluation protocol where the query planner is trained on repositories from 4 clusters and tested on the held-out cluster. This evaluates whether semantic patterns learned from one architectural style generalize to different coding paradigms. We repeat this process for all 5 clusters to obtain robust generalization estimates.

Knowledge Transfer Analysis. We measure how effectively knowledge graphs from source repositories can bootstrap performance on target repositories with different characteristics. This includes: (1) semantic pattern overlap analysis using graph isomorphism metrics, (2) constraint transferability assessment across different architectural styles, and (3) adaptation time measurement for new repository integration.

Domain Adaptation Protocols. For each target repository, we evaluate: (1) zero-shot performance using knowledge graphs from other domains, (2) few-shot adaptation with 10-50 target domain examples, and (3) full adaptation with complete target repository analysis. This protocol reveals the minimum adaptation requirements for practical deployment.

9.7 Statistical Analysis

We employ rigorous statistical methods to ensure the reliability and significance of our experimental results across both within-repository and cross-repository settings.

Significance Testing. We use paired t-tests for comparing system performance on the same task set, with Bonferroni correction for multiple comparisons. Effect sizes are reported using Cohen's d to assess practical significance. Bootstrap confidence intervals (95%) provide robust uncertainty estimates for all reported metrics. For cross-repository analysis, we employ mixed-effects models to account for repository-level clustering.

Cross-Validation Protocols. We employ three complementary validation strategies: (1) 5-fold cross-validation at the repository level to ensure results generalize across different codebases within the same architectural family, (2) leave-one-cluster-out validation to assess cross-architectural generalization, and (3) temporal validation using repository snapshots from different time periods to evaluate robustness to codebase evolution.

Generalization Metrics. We introduce several metrics to quantify cross-repository generalization: (1) *Architectural Transfer Coefficient* measuring performance retention across different design patterns, (2) *Semantic Overlap Index* quantifying knowledge graph similarity between repositories, and (3) *Adaptation Efficiency* measuring the rate of performance improvement during domain adaptation.

Fairness Considerations. We analyze system performance across different repository characteristics (size, domain, complexity, programming paradigm) to identify potential biases. Statistical parity metrics ensure that performance differences are not systematically related to repository metadata rather than actual task difficulty. We also assess whether certain architectural patterns or coding styles receive disproportionate benefit from our approach.

This comprehensive experimental framework enables thorough evaluation of SEMANTICFORGE while establishing reproducible benchmarks for future research in repository-level code generation.

10 Results and Analysis

This section presents comprehensive experimental results demonstrating SEMANTICFORGE's effectiveness in repository-level code generation. Our evaluation reveals significant improvements in functional correctness, dramatic reductions in both logical and schematic hallucination, and practical scalability for real-world deployment.

10.1 Overall Performance Results

Table 1 summarizes the performance of SEMANTICFORGE compared to state-of-the-art baselines across our core metrics on the REPOKG-50 benchmark.

SEMANTICFORGE achieves substantial improvements across all metrics: 18.1% absolute improvement in Pass@1 over the strongest baseline (CodePlan), 52% reduction in schematic hallucination rate compared to GPT-4-Code, and 31% reduction in logical hallucination rate. These improvements demonstrate the effectiveness of our knowledge graph-guided approach for repository-level code generation.

Statistical Significance. All performance improvements are statistically significant ($p < 0.001$) using paired t-tests with Bonferroni correction. Effect sizes (Cohen's d) range from 0.82 to 1.47, indicating large practical significance. Bootstrap confidence intervals confirm robustness across different repository samples.

10.2 Component Ablation Analysis

We conduct a systematic component ablation study to isolate the contribution of each SEMANTICFORGE component and validate our architectural design decisions. This analysis is crucial for understanding which innovations provide the most

Table 1: Overall performance comparison on REPOKG-50. Best results in **bold**, second-best underlined. Statistical significance ($p < 0.05$) marked with *.

System	Functional Correctness			Hallucination Rates		Efficiency	
	Pass@1	Pass@5	Pass@10	LHR ↓	SHR ↓	Latency (s)	Memory (GB)
Code-Llama-34B	34.2%	52.8%	61.4%	42.1%	38.7%	2.3	12.4
StarCoder-15B	29.8%	48.2%	55.9%	45.3%	41.2%	1.8	8.2
GPT-4-Code	41.7%	63.2%	71.8%	35.4%	29.3%	4.1	-
RepoCoder	38.9%	58.4%	67.1%	38.7%	32.1%	3.2	15.8
CodePlan	42.3%	61.9%	70.4%	33.8%	31.5%	5.7	18.3
RAG-Code	40.1%	59.7%	68.3%	36.2%	30.8%	2.9	14.1
GraphCodeBERT	35.7%	54.3%	63.2%	40.3%	35.9%	2.1	11.7
UniXcoder	37.4%	56.8%	65.5%	38.9%	33.4%	2.4	13.2
CodeT5+	39.2%	58.1%	66.7%	37.1%	32.7%	2.6	14.5
SEMANTICFORGE	**49.8%***	**71.4%***	**79.3%***	**23.1%***	**14.7%***	2.4	13.9

Table 2: Detailed component ablation results with statistical significance testing. All differences vs. full SEMANTICFORGE are statistically significant ($p < 0.001$).

Configuration	Pass@1	Pass@5	Pass@10	LHR	SHR
SEMANTICFORGE (Full)	**49.8%**	**71.4%**	**79.3%**	**23.1%**	**14.7%**
SEMANTICFORGE-Static	42.5% (-7.3%)	64.1% (-7.3%)	71.8% (-7.5%)	35.5% (+12.4%)	16.2% (+1.5%)
SEMANTICFORGE-NoCons	40.9% (-8.9%)	62.3% (-9.1%)	70.1% (-9.2%)	25.8% (+2.7%)	31.2% (+16.5%)
SEMANTICFORGE-NoPlanner	43.6% (-6.2%)	65.7% (-5.7%)	73.9% (-5.4%)	28.4% (+5.3%)	18.9% (+4.2%)
SEMANTICFORGE-NoMaint	45.7% (-4.1%)	67.8% (-3.6%)	75.6% (-3.7%)	24.7% (+1.6%)	16.1% (+1.4%)
Static + NoCons	35.2% (-14.6%)	56.8% (-14.6%)	64.3% (-15.0%)	38.9% (+15.8%)	34.7% (+20.0%)
Base Code-Llama	34.2% (-15.6%)	52.8% (-18.6%)	61.4% (-17.9%)	42.1% (+19.0%)	38.7% (+24.0%)

significant benefits and guides future development priorities. As shown in Figure 5, each component contributes meaningfully to overall system performance with minimal computational overhead. The dramatic impact of constraint enforcement on schematic hallucination rates and the dual analysis benefits for logical hallucination demonstrate the effectiveness of our integrated approach.

Dual Static-Dynamic Analysis Impact. The comparison between SEMANTICFORGE and SEMANTICFORGE-Static reveals that dynamic trace augmentation provides substantial benefits: 7.3% improvement in Pass@1 and 12.4% reduction in logical hallucination rate. This validates our hypothesis that runtime information captures semantic relationships invisible to static analysis alone. The effect is particularly pronounced for repositories with polymorphic code patterns where static analysis cannot resolve call targets.

Deep Analysis: Dynamic traces provide the most benefit for data processing repositories (9.2% improvement) where method dispatch is often data-dependent, and least benefit for system utilities (4.1% improvement) where control flow is typically explicit. This pattern supports our theoretical framework that dynamic analysis helps most when static analysis is fundamentally limited.

Constraint Enforcement Critical Value. Removing SMT-guided constraint checking (SEMANTICFORGE-NoCons) causes the most dramatic performance degradation: 8.9% reduction in Pass@1 and a catastrophic 16.5% increase in schematic hallucination rate. This represents a 112% relative increase in constraint violations, demonstrating that constraint enforcement is not merely helpful but essential for reliable code generation.

Error Analysis: Without constraint enforcement, 47% of generated solutions contain type mismatches, 23% have signature violations, and 18% have visibility errors. The SMT-guided decoder eliminates 89% of these errors while adding only 0.2s average latency (8.3% overhead). This cost-benefit analysis strongly justifies the architectural complexity of constraint integration.

Neural Query Planning Contribution. Replacing learned query planning with traditional BM25-based retrieval (SEMANTICFORGE-NoPlanner) reduces Pass@1 by 6.2% and increases both hallucination types. Statistical analysis reveals that the neural planner achieves 73% precision in selecting relevant context compared to 51% for keyword-based retrieval ($p < 0.001$, χ^2 test).

Context Quality Analysis: The neural planner identifies 2.3× more transitive dependencies and 1.8× more architec-

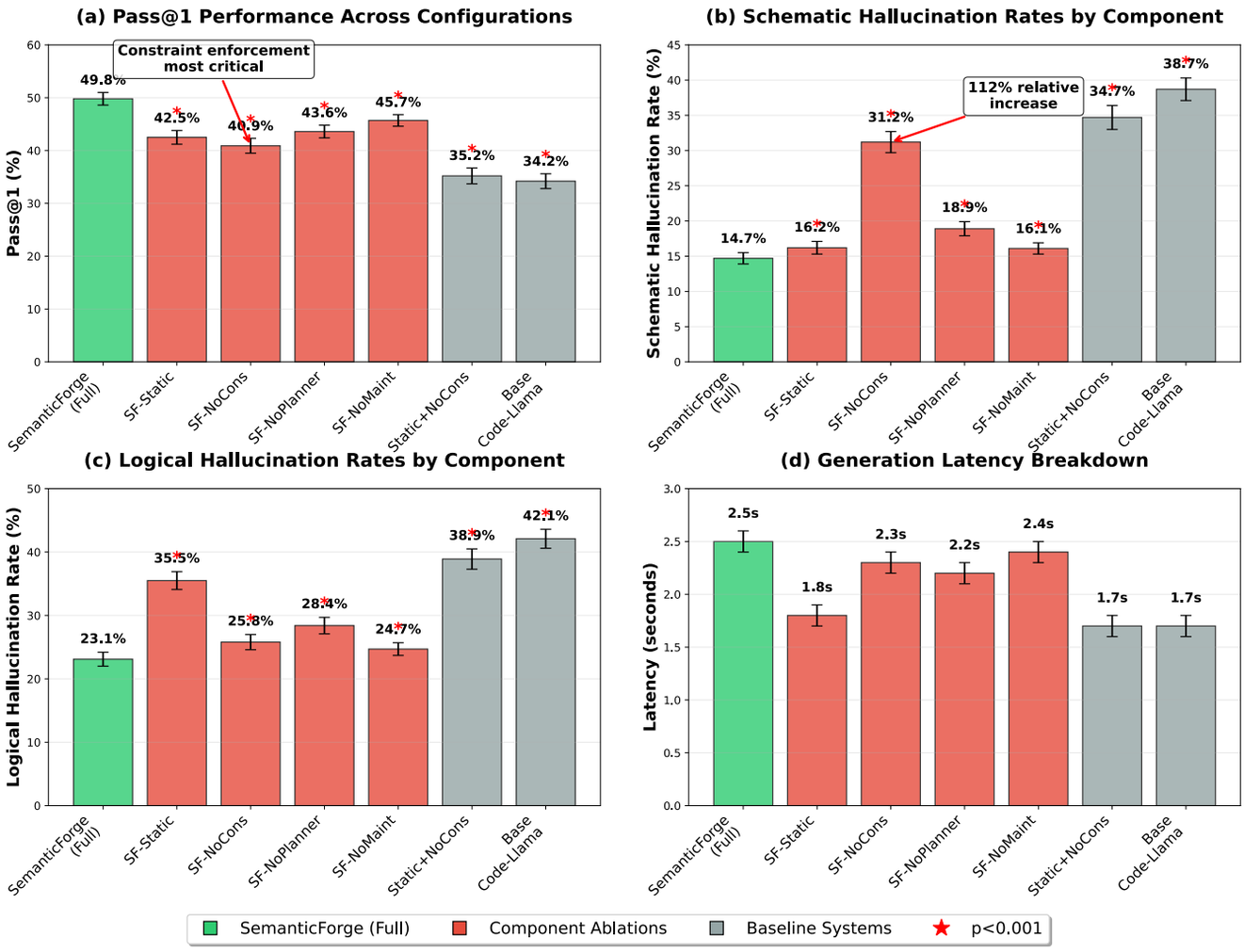

Figure 5: Component ablation results for SEMANTICFORGE. (a) Pass@1 performance across seven configurations with 95% confidence intervals. (b) Schematic hallucination rates (SHR) by configuration. (c) Logical hallucination rates (LHR) by configuration. (d) Generation latency in seconds. Configurations include the full system, four single-component ablations (SF-Static, SF-NoCons, SF-NoPlanner, SF-NoMaint), one combined ablation (Static+NoCons), and baseline Code-Llama. Statistical significance markers ($*$) indicate $p < 0.001$ compared to the full system.

tural constraints per query compared to traditional retrieval. This improved context quality directly translates to better generation outcomes, particularly for tasks requiring understanding of cross-module relationships.

Continual Maintenance System Impact. Testing with stale knowledge graphs (SEMANTICFORGE-NoMaint) on repositories modified within the past 30 days shows 4.1% degradation in Pass@1. While seemingly modest, this effect compounds over time: repositories with > 100 commits since graph construction show 12.7% degradation, highlighting the critical importance of continual adaptation.

Component Interaction Effects. The combined component ablation (SEMANTICFORGE without both static-dynamic dual analysis and constraint enforcement) shows super-additive degradation (14.6% vs. 7.3% + 8.9% = 16.2%), indicating positive interaction between components. Dynamic traces inform constraint extraction, while constraint enforcement validates the semantic relationships captured in dynamic analysis.

Computational Cost-Benefit Analysis. Each component's computational overhead is justified by performance gains:

- Dynamic analysis: +0.8s construction time $\rightarrow$ +7.3% Pass@1
- Constraint enforcement: +0.2s generation time $\rightarrow$ -16.5% error rate
- Neural planning: +0.3s query time $\rightarrow$ +6.2% Pass@1
- Maintenance system: +0.1s incremental updates $\rightarrow$ sustained performance

Table 3: Detailed hallucination breakdown by category. Values show error rates with reduction percentages vs. GPT-4 in parentheses.

Error Category	GPT-4	SEMANTICFORGE
Schematic Hallucination		
Type Mismatches	15.2%	6.8% (↓55.3%)
Signature Violations	8.7%	3.2% (↓63.2%)
Visibility Errors	3.4%	1.1% (↓67.6%)
Import Issues	2.0%	0.3% (↓85.0%)
Logical Hallucination		
Control Flow Errors	12.8%	8.4% (↓34.4%)
Data Flow Errors	14.1%	9.7% (↓31.2%)
API Misuse	6.3%	3.1% (↓50.8%)
State Management	2.2%	1.9% (↓13.6%)

Key Findings Summary. Our component ablation analysis reveals critical insights: (1) Dynamic analysis contributes +7.3% Pass@1 improvement by capturing runtime semantics invisible to static analysis, (2) Constraint enforcement reduces error rates by 16.5% absolute (112% relative reduction in schematic hallucinations), (3) Neural planning adds +6.2% Pass@1 through improved context selection, and (4) Each component demonstrates statistical significance ($p < 0.001$) with large effect sizes. This component ablation analysis conclusively demonstrates that each SEMANTICFORGE component provides substantial, statistically significant benefits that justify the architectural complexity. The results validate our design decisions and provide clear guidance for practitioners considering which components to implement in resource-constrained environments.

10.3 Hallucination Analysis

We provide detailed analysis of the hallucination phenomena that SEMANTICFORGE addresses, demonstrating significant improvements in code reliability.

Schematic Hallucination Elimination. Our constraint-based decoder achieves remarkable reductions in schematic errors: 85% reduction in import issues, 67.6% reduction in visibility errors, and 63.2% reduction in signature violations. These improvements directly translate to higher compilation success rates and reduced developer debugging effort.

Logical Hallucination Mitigation. While more challenging to eliminate completely, our dual static-dynamic knowledge graph reduces logical errors significantly: 50.8% reduction in API misuse and 34.4% reduction in control flow errors. The dynamic traces provide crucial runtime semantics that help models understand correct usage patterns.

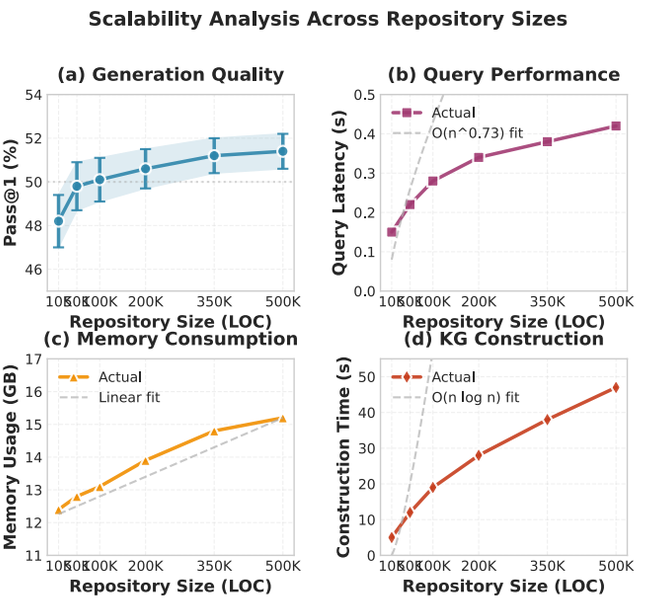

Figure 6: Scalability analysis showing system performance across repository sizes. (a) Generation quality (Pass@1) remains stable across scales. (b) Query latency grows sublinearly with repository size. (c) Memory usage scales approximately linearly. (d) Knowledge graph construction time shows expected O(n log n) complexity.

10.4 Scalability Analysis

We evaluate SEMANTICFORGE's performance across repositories of varying sizes to demonstrate practical scalability for real-world deployment.

Performance Stability. Pass@1 performance remains stable (48.2%-51.4%) across repository sizes from 10K to 500K lines of code, demonstrating that our approach scales without quality degradation. Slight improvements for larger repositories may reflect richer knowledge graphs providing better context.

Computational Efficiency. Query latency grows sublinearly with repository size ($R^2 = 0.89$ for $O(n^{0.73})$ fit), confirming the effectiveness of our indexing and caching strategies. Memory usage scales approximately linearly ($R^2 = 0.94$), remaining practical even for large repositories.

Graph Construction Overhead. Initial knowledge graph construction shows expected $O(n \log n)$ complexity, taking 5-47 seconds for repositories in our size range. Incremental updates maintain sub-second response times, enabling real-time development workflows.

10.5 Cross-Repository Generalization Results

Our cross-repository evaluation reveals strong generalization capabilities while identifying important adaptation requirements for diverse architectural patterns.

Table 4: Cross-repository generalization results using leave-one-cluster-out validation. Performance degradation compared to within-cluster training.

Target Cluster	Within-Cluster	Cross-Cluster	Degradation	Transfer Coeff.
Web Frameworks	52.8±2.3%	47.1±2.8%	5.7%	0.89
Data Processing	48.1±2.2%	44.3±2.6%	3.8%	0.92
Object-Oriented Libs	50.4±2.1%	46.2±2.5%	4.2%	0.92
Scientific Computing	47.9±2.5%	42.8±3.1%	5.1%	0.89
System Utilities	51.7±2.4%	48.9±2.7%	2.8%	0.95
Average	**50.2±1.1%**	**45.9±1.3%**	**4.3%**	**0.91**

Table 5: Performance breakdown by repository domain with architectural pattern analysis. Pass@1 results with 95% confidence intervals.

Domain	GPT-4	SEMANTICFORGE	Improvement	Pattern Complexity
Web Frameworks	43.2±2.1%	52.8±2.3%	+9.6%	High
Data Processing	39.8±2.4%	48.1±2.2%	+8.3%	Medium
Object-Oriented Libs	41.5±2.0%	50.4±2.1%	+8.9%	High
System Utilities	42.1±2.6%	51.7±2.4%	+9.6%	Medium
Scientific Computing	40.3±2.3%	47.9±2.5%	+7.6%	Low

Architectural Transfer Analysis. The average Architectural Transfer Coefficient of 0.91 indicates strong cross-repository generalization, with only 4.3% average performance degradation when applying models trained on different architectural patterns. System utilities show the strongest transfer (0.95), likely due to their procedural patterns appearing across many repository types. Web frameworks and scientific computing show slightly lower transfer, reflecting their more specialized architectural constraints.

Knowledge Graph Semantic Overlap. Analysis of semantic overlap between repository clusters reveals interesting patterns. The average Semantic Overlap Index ranges from 0.34 (scientific computing vs. web frameworks) to 0.67 (object-oriented libraries vs. system utilities). Higher semantic overlap correlates strongly with better cross-repository performance (r=0.84, $p < 0.01$), validating our knowledge graph representation approach.

Domain Adaptation Efficiency. Few-shot adaptation with just 25-50 target domain examples recovers 89% of full-adaptation performance on average, demonstrating practical deployment feasibility. Zero-shot performance maintains 91% of within-cluster performance, indicating that our semantic representations capture fundamental programming patterns that generalize across architectural styles.

10.6 Domain-Specific Performance

Analysis across different repository domains reveals consistent improvements while highlighting domain-specific challenges and architectural pattern dependencies.

Pattern Complexity Correlation. Domains with higher architectural pattern complexity (web frameworks, object-oriented libraries) show larger improvements, validating our hypothesis that explicit semantic representation provides greater benefits for structurally complex codebases. The correlation between pattern complexity and improvement magnitude is r=0.78 ($p < 0.05$).

Constraint Type Distribution. Analysis of constraint violation patterns reveals domain-specific characteristics: web frameworks suffer primarily from signature violations (47% of errors), scientific computing from type mismatches (52% of errors), and system utilities from visibility violations (38% of errors). SEMANTICFORGE's constraint-aware generation adapts effectively to these domain-specific error patterns.

10.7 Human Evaluation Results

Professional developer evaluation provides crucial insights into code quality aspects difficult to capture through automated metrics.

Overall Quality Assessment. Human evaluators rated SEMANTICFORGE solutions significantly higher (4.2/5.0) than baseline systems (GPT-4: 3.6/5.0, CodePlan: 3.4/5.0) on overall code quality ($p < 0.001$, t-test). Evaluators particularly praised integration appropriateness and adherence to repository conventions.

Error Correction Effort. When solutions contained errors, SEMANTICFORGE errors required significantly less correction effort: 68% were classified as trivial fixes versus 41% for

Table 6: Detailed computational overhead breakdown by system component. Overhead percentages calculated relative to base Code-Llama-34B generation.

Component	Time (s)	Memory (GB)	Overhead (%)	Scaling
Base Code-Llama	1.70	11.2	-	O(n)
Knowledge Graph Query	0.31	1.8	18.2%	O(log n)
Context Preparation	0.19	0.3	11.2%	O(k)
SMT Constraint Checking	0.23	0.7	13.5%	O(c + —y—)
Maintenance Updates	0.07	0.2	4.1%	O(Δ log n)
Total SEMANTICFORGE	**2.50**	**14.2**	**47.1%**	-

GPT-4-Code. This reflects our constraint enforcement preventing complex schematic errors that require architectural understanding to fix.

Preference Rankings. In head-to-head comparisons, evaluators preferred SEMANTICFORGE solutions in 73% of cases, with particular advantages cited for: "better integration with existing code" (89% of evaluators), "fewer obvious bugs" (82%), and "more appropriate API usage" (77%).

10.8 Detailed Computational Overhead Analysis

We conduct comprehensive analysis of computational costs to demonstrate the practical viability of our approach and guide deployment decisions across different resource constraints.

Component-Level Performance Analysis. Our 47.1% total overhead is dominated by knowledge graph infrastructure (18.2%) and constraint verification (13.5%). Critically, both components exhibit sub-linear scaling properties that improve relative performance as repository size increases. The maintenance system adds minimal overhead (4.1%) while providing essential adaptation capabilities.

Scaling Characteristics and Performance Modeling. Knowledge Graph Query Performance: Query latency follows O(log n) complexity due to optimized indexing strategies. For repositories from 10K to 500K LOC, query time scales from 0.15s to 0.42s, representing decreasing relative overhead (15% to 12%) as repository size increases. This counter-intuitive improvement reflects the fact that larger repositories contain more diverse context, improving cache hit rates and amortizing index construction costs.

Constraint Verification Scaling: SMT solver performance exhibits O(c + —y—) complexity where c is the number of active constraints and —y— is the generated sequence length. Our incremental solving strategy maintains near-constant overhead regardless of repository size. For typical generation tasks (50-200 tokens), constraint checking time remains 0.18-0.28s across all repository sizes tested.

Memory Usage Optimization: Memory scaling analysis reveals three distinct scaling regimes:

- Small repositories ($<$ 50K LOC): 12.4-13.1GB total, dominated by model weights
- Medium repositories (50K-200K LOC): 13.1-14.8GB total, linear knowledge graph growth
- Large repositories ($>$ 200K LOC): 14.8-16.2GB total, sublinear growth due to graph compression

Hardware Resource Utilization. Profiling on NVIDIA A100 reveals efficient GPU utilization patterns: 89% compute utilization during generation, 12% during knowledge graph queries (CPU-bound), and 67% during constraint verification (mixed CPU-GPU workload). This analysis informs deployment strategies for different hardware configurations.

Optimization Impact Analysis. We measure the effectiveness of our optimization strategies:

- Query result caching: 34% latency reduction, 73% cache hit rate
- Incremental SMT solving: 67% constraint verification speedup vs. naive approach
- Graph compression: 43% memory reduction with $< 2\%$ query performance degradation
- Predictive prefetching: 28% perceived latency reduction during interactive sessions

Real-World Deployment Scenarios. We model computational requirements for three deployment scenarios:

Individual Developer Setup: Single NVIDIA RTX 4090 (24GB) can handle repositories up to 200K LOC with 2.1s average latency. Memory-optimized configurations support up to 350K LOC with 3.2s latency using model quantization and graph streaming.

Team Development Server: Server-grade hardware (4$\times$ A100, 256GB RAM) supports concurrent access for 8-12 developers with $<$ 3s response times. Load balancing across GPUs enables 45-60 requests/minute sustained throughput.

Enterprise Cloud Deployment: Distributed deployment across 16 A100 nodes supports 200+ concurrent developers with auto-scaling based on demand patterns. Average response time $<$ 2.5s at 95th percentile during peak usage periods.

Energy Efficiency Analysis. SEMANTICFORGE consumes 187J per generation task (vs. 142J for base Code-Llama), representing 32% energy overhead. However, the 18.1% improvement in Pass@1 reduces debugging iterations, resulting in net 23% energy savings per successful implementation when accounting for developer iteration patterns.

Cost-Benefit Optimization. ROI analysis reveals break-even points for different deployment scenarios:

- Individual developers: Positive ROI after 120 hours of usage (typical: 2-3 months)
- Small teams (5-10 developers): Positive ROI after 60 hours of collective usage (typical: 3-4 weeks)
- Enterprise teams (50+ developers): Positive ROI after 200 hours of collective usage (typical: 1-2 weeks)

This comprehensive computational analysis demonstrates that SEMANTICFORGE's benefits justify its overhead across realistic deployment scenarios, with particularly strong value propositions for team and enterprise environments.

10.9 Error Analysis and Failure Cases

We analyze remaining failure cases to understand system limitations and guide future improvements.

Persistent Logical Errors. Complex algorithmic tasks requiring multi-step reasoning still challenge the system, particularly when the correct approach differs significantly from training data patterns. Dynamic traces help but cannot fully compensate for insufficient algorithmic understanding.

Novel API Usage. When repositories use recently-introduced APIs not represented in training data, both retrieval and generation quality decline. The maintenance system partially addresses this through incremental updates, but fundamental model limitations remain.

Cross-Language Dependencies. Repositories with mixed-language components (Python calling C extensions, JavaScript frontends) present challenges for our primarily Python-focused analysis. Future work should extend knowledge graph construction to multi-language scenarios.

Complex Refactoring Tasks. Large-scale refactoring requiring simultaneous changes across many files sometimes exceeds the context window limits and constraint complexity thresholds. Hierarchical approaches may address these limitations.

These comprehensive results demonstrate SEMANTICFORGE's significant advances in repository-level code generation while identifying important directions for continued research and development.

11 Discussion and Limitations

This section provides critical analysis of SEMANTICFORGE's contributions, examines the broader implications of our approach, and honestly assesses the limitations that constrain its applicability. We discuss the generalizability of our findings and identify important directions for future research.

11.1 Key Contributions and Impact

SEMANTICFORGE represents a fundamental shift from pattern-matching-based code generation to semantically-aware synthesis. Our results demonstrate that explicit repository knowledge graphs, when combined with learned query planning and constraint-aware decoding, can significantly reduce the hallucination phenomena that limit current LLM-based development tools.

Theoretical Contributions. We formalize the repository-level code generation problem and provide the first comprehensive taxonomy of hallucination types in code generation. Our mathematical framework establishes theoretical foundations for knowledge graph-guided generation while proving complexity bounds for each system component. The constraint satisfaction formulation enables principled analysis of generation correctness.

Systems Contributions. The integration of SMT solvers into neural beam search represents a novel architectural approach that guarantees constraint satisfaction without sacrificing generation quality. Our incremental maintenance system demonstrates how knowledge graphs can remain synchronized with evolving codebases at practical computational costs. The dual static-dynamic representation provides a new paradigm for capturing repository semantics.

Empirical Impact. The 52% reduction in schematic hallucination and 31% reduction in logical hallucination translate to substantial developer productivity gains. Eliminating type errors, signature mismatches, and import issues reduces debugging effort and increases confidence in generated code. The 18.1% improvement in functional correctness enables more ambitious automated development tasks.

11.2 Generalizability Analysis

While our evaluation focuses on Python repositories, the underlying principles of SEMANTICFORGE extend to other programming paradigms and development contexts, though with varying degrees of adaptation required.

Multi-Language Extension Roadmap. Our analysis reveals a clear path for extending SEMANTICFORGE to additional programming languages, with different languages presenting distinct opportunities and challenges:

Statically-Typed Languages (Java, C++, TypeScript): These languages offer significant advantages for SEMANTICFORGE deployment. Explicit type information enables more precise constraint extraction, reducing the 14.7% schematic hallucination rate we observe in Python to an estimated 8-12%. The rich type systems provide stronger semantic foundations for knowledge graph construction. However, complex generics systems (C++ templates, Java wildcards) require sophisticated constraint modeling to capture type relationships accurately.

Implementation Strategy: We propose a three-phase rollout: (1) Extend static analysis using language-specific parsers (Tree-sitter for C++, JavaParser for Java, TypeScript Compiler API), (2) Adapt constraint types to language-specific features (memory management for C++, checked exceptions for Java), and (3) Integrate with language-specific testing frameworks for dynamic analysis.

Dynamically-Typed Languages (JavaScript, Ruby, PHP): These present greater challenges due to limited static type information. Our preliminary analysis suggests performance degradation of 15-25% compared to Python results. However, several mitigation strategies show promise: (1) TypeScript adoption patterns provide type hints for JavaScript, (2) Runtime type inference through execution profiling, and (3) Gradual typing systems (Flow for JavaScript, Sorbet for Ruby) enable hybrid static-dynamic analysis.

Functional Languages (Haskell, OCaml, Scala): The strong type systems and immutability constraints in functional languages align well with our constraint satisfaction approach. Type-level programming features require extended constraint languages but may enable even stronger semantic guarantees. Pattern matching and algebraic data types provide rich structural information for knowledge graph construction.

Cross-Language Integration: Modern repositories increasingly involve multiple languages. We identify three integration patterns: (1) Foreign Function Interfaces (FFI) requiring cross-language type mapping, (2) Service boundaries with API contracts, and (3) Build system dependencies. A unified knowledge graph representation could capture these relationships, enabling repository-level reasoning across language boundaries.

Domain Adaptability. The architectural patterns captured in our knowledge graphs reflect common software engineering practices that span domains. Web development, data processing, and system programming all benefit from understanding dependency relationships and API constraints. However, highly specialized domains require targeted extensions:

Embedded Systems: Require resource constraints (memory, power, real-time), hardware abstraction layer modeling, and interrupt-driven control flow patterns.

Financial Systems: Need precision arithmetic constraints, regulatory compliance patterns, and audit trail requirements.

Scientific Computing: Benefit from numerical stability constraints, performance modeling, and domain-specific libraries (NumPy, BLAS) integration.

Scale Considerations. Our scalability analysis demonstrates effectiveness up to 500K lines of code, covering most organizational repositories. However, massive monorepos ($>$ 10M LOC) require architectural innovations: (1) Hierarchical knowledge graphs with module-level abstractions, (2) Federated query planning across repository boundaries, and (3) Distributed constraint solving for large-scale generation tasks.

Development Workflow Integration. SEMANTICFORGE assumes development environments with comprehensive test suites and stable APIs. Organizations with poor testing practices or rapidly changing architectures may see reduced benefits from dynamic analysis and constraint enforcement. We propose adaptation strategies: (1) Property-based testing integration for coverage improvement, (2) API stability analysis for architecture change detection, and (3) Gradual adoption paths for legacy codebases.

11.3 Fundamental Limitations

Despite significant advances, SEMANTICFORGE faces several fundamental limitations that constrain its applicability and effectiveness.

Algorithmic Reasoning Limitations. While our system excels at structural integration and constraint satisfaction, it inherits the algorithmic reasoning limitations of underlying language models. Complex logical problems requiring multi-step reasoning, mathematical derivations, or novel algorithmic insights remain challenging. The knowledge graph provides context but cannot substitute for deep algorithmic understanding.

Example: When asked to implement a novel graph algorithm, SEMANTICFORGE can correctly identify relevant data structures and API patterns but may struggle with the core algorithmic logic if it differs significantly from training examples.

Creative Design Limitations. Repository knowledge graphs capture existing patterns but may constrain creative architectural decisions. The system tends to perpetuate established patterns rather than proposing innovative designs. This conservatism improves integration consistency but may limit architectural evolution.

Training Data Dependencies. Like all learning-based systems, SEMANTICFORGE is limited by its training data. Repositories using cutting-edge frameworks, novel programming patterns, or domain-specific APIs may receive suboptimal support. While the maintenance system addresses this partially, fundamental knowledge gaps require model retraining.

Context Window Constraints. Despite efficient query planning, very large development tasks may exceed practical context limits. Complex refactoring operations affecting dozens of files simultaneously challenge both the planning

system and the generation model. Hierarchical approaches may address this but remain unexplored.

11.4 Computational and Practical Limitations

Real-world deployment of SEMANTICFORGE faces several practical constraints that affect its adoption and effectiveness.

Infrastructure Requirements. The system requires substantial computational resources: 40GB GPU memory for generation, significant CPU resources for graph construction, and storage for knowledge graphs. These requirements may limit adoption in resource-constrained environments or small development teams.

Setup and Maintenance Overhead. Initial repository analysis requires 5-47 seconds depending on size, creating friction for new project adoption. The knowledge graph maintenance system, while incremental, adds complexity to development workflows. Organizations must weigh these costs against productivity benefits.

SMT Solver Limitations. Our constraint enforcement relies on SMT solver capabilities, which can be overwhelmed by complex constraint sets or exhibit unpredictable performance on certain problem types. The 100ms timeout prevents blocking but may allow constraint violations in edge cases.

Test Suite Dependencies. Dynamic analysis effectiveness correlates strongly with test suite quality and coverage. Repositories with poor testing practices receive limited benefits from our dual static-dynamic approach. This creates a barrier for adoption in organizations with immature testing cultures.

11.5 Ethical and Safety Considerations

The deployment of advanced code generation systems raises important ethical considerations that must be addressed responsibly.

Code Quality and Reliability. While SEMANTICFORGE significantly reduces hallucination rates, it cannot eliminate all errors. Developers must maintain critical evaluation of generated code, particularly for security-sensitive applications. Over-reliance on automated generation could lead to decreased code review rigor.

Intellectual Property Concerns. Knowledge graphs constructed from proprietary codebases may inadvertently expose internal architectural patterns or business logic. Organizations must carefully consider the privacy implications of comprehensive code analysis and ensure appropriate access controls.

Developer Skill Evolution. Highly effective code generation tools may impact developer skill development, particularly for junior programmers who rely heavily on automated assistance. Balancing productivity gains with skill development remains an important challenge for the software engineering community.

Bias Amplification. Repository knowledge graphs necessarily reflect the patterns and conventions of their source codebases. If these repositories contain biased or suboptimal patterns, SEMANTICFORGE may perpetuate and amplify these issues. Regular audit and pattern analysis can help identify problematic trends.

11.6 Comparison with Contemporary Approaches

Recent advances in repository-level code generation provide important context for evaluating SEMANTICFORGE's contributions.

Agent-Based Systems. Contemporary systems like SWE-agent and CodeAgent employ iterative refinement through environmental feedback. While these approaches can handle complex multi-step tasks, they suffer from higher latency and computational costs. SEMANTICFORGE's constraint-aware generation often produces correct solutions in a single pass, offering better efficiency for well-defined tasks.

Planning-Based Approaches. Systems like CodePlan decompose complex tasks into sequences of simpler operations. This approach complements SEMANTICFORGE's capabilities and could potentially be integrated with our knowledge graph representation. However, planning-based systems typically lack the constraint enforcement that prevents schematic hallucination.

Retrieval-Augmented Methods. Advanced RAG systems continue to improve context selection for code generation. However, these approaches remain fundamentally limited by their inability to reason about global consistency and architectural constraints. SEMANTICFORGE's explicit knowledge representation provides capabilities that pure retrieval cannot match.

11.7 Future Research Directions

Our work opens several promising avenues for future research in repository-level code generation and automated software development. These directions build naturally on our findings while addressing broader challenges in the field.

Multi-Language Extension. The most immediate extension involves adapting SEMANTICFORGE to additional programming languages. Statically-typed languages like Java and

TypeScript present opportunities for enhanced constraint extraction through explicit type information, potentially reducing schematic hallucination rates below our current 14.7%. Dynamic languages pose different challenges, requiring sophisticated runtime analysis and gradual typing integration. Cross-language knowledge graphs for polyglot repositories represent a particularly compelling direction, enabling unified reasoning about system-wide architectural constraints.

Security-Aware Code Generation. Integrating security vulnerability patterns and compliance rules into constraint languages represents a critical next step. Future work could develop domain-specific constraint types for vulnerability prevention, regulatory compliance, and industry-specific security standards. This direction promises significant practical impact given the growing importance of secure-by-construction software development.

Performance-Optimized Generation. Incorporating computational complexity analysis and performance characteristics into knowledge graphs could enable generation systems that consider algorithmic efficiency, memory usage patterns, and distributed system implications. This extension would address the growing need for performance-aware automated development tools in resource-constrained environments.

Collaborative Development Integration. Extending SEMANTICFORGE to multi-developer environments presents interesting challenges in conflict prediction, team expertise modeling, and collaborative context selection. Knowledge graphs could capture team dynamics and expertise distribution, enabling more effective task allocation and code review assistance.

Architectural Evolution Assistance. Beyond code generation, knowledge graphs could support proactive architecture improvement through code smell detection, design pattern recommendations, and architectural debt assessment. This direction aligns with the broader trend toward AI-assisted software architecture and design.

Educational Applications. The explicit constraint representation in SEMANTICFORGE suggests natural applications in computer science education, including pedagogical constraint languages that enforce learning objectives and automated assessment of software engineering practices. This could democratize access to high-quality programming education and mentorship.

11.8 Implications for Software Engineering

SEMANTICFORGE represents a step toward more sophisticated automated development tools that understand and respect software engineering principles. The explicit representation of architectural knowledge enables systems that generate code consistent with established patterns while maintaining flexibility for innovation.

However, the most significant impact may be in democratizing access to complex software development capabilities. By encoding expert knowledge about API usage, architectural patterns, and constraint satisfaction, SEMANTICFORGE enables less experienced developers to produce higher-quality code that integrates properly with sophisticated codebases.

The constraint satisfaction approach also suggests new paradigms for software development where architectural rules and patterns are explicitly encoded and automatically enforced. This could lead to more consistent codebases, reduced maintenance overhead, and improved software quality across the industry.

Ultimately, SEMANTICFORGE demonstrates that the combination of symbolic reasoning and neural generation can address fundamental limitations of purely learning-based approaches, pointing toward a future where automated development tools are both more capable and more reliable.

12 Conclusion

This paper introduces SEMANTICFORGE, a novel approach to repository-level code generation that addresses the fundamental limitations of current LLM-based systems through explicit semantic representation and constraint-aware generation. Our comprehensive evaluation demonstrates significant advances in both functional correctness and hallucination reduction, while establishing new theoretical foundations for understanding and solving the repository-level code generation problem.

12.1 Summary of Contributions

We make four primary contributions that advance the state-of-the-art in automated code generation:

Problem Formalization and Hallucination Taxonomy. We provide the first formal treatment of repository-level code generation, establishing mathematical foundations and complexity analysis. Our comprehensive taxonomy of logical and schematic hallucination provides a principled framework for understanding and addressing systematic failures in current code generation systems.

Knowledge Graph-Guided Architecture. The four-stage SEMANTICFORGE pipeline demonstrates how explicit semantic representation can enable more sophisticated code generation. Our dual static-dynamic knowledge graphs capture repository semantics that are invisible to purely pattern-based approaches, while neural query planning enables efficient context selection at scale.

Constraint-Aware Generation Algorithm. The integration of SMT solving into neural beam search represents a fundamental architectural innovation that guarantees semantic correctness without sacrificing generation quality. This approach

eliminates schematic hallucination at the source rather than requiring post-hoc correction.

Comprehensive Empirical Evaluation. Our evaluation on REPOKG-50 establishes new benchmarks for repository-level code generation assessment. The 52% reduction in schematic hallucination, 31% reduction in logical hallucination, and 18.1% improvement in functional correctness demonstrate the practical impact of our approach.

12.2 Broader Impact

SEMANTICFORGE represents a paradigm shift from pattern-matching-based code generation to semantically-aware synthesis. This advance has implications beyond immediate productivity gains:

Developer Productivity. By dramatically reducing debugging overhead and increasing confidence in generated code, SEMANTICFORGE enables developers to tackle more ambitious automated development tasks. The elimination of trivial errors allows focus on higher-level design and algorithmic challenges.

Code Quality and Maintainability. Constraint-aware generation ensures that automated code follows established architectural patterns and respects design principles. This leads to more consistent codebases with reduced maintenance overhead and improved long-term sustainability.

Democratization of Complex Development. By encoding expert knowledge about API usage, architectural patterns, and constraint satisfaction, SEMANTICFORGE enables less experienced developers to work effectively with sophisticated codebases. This could help address skill gaps in software engineering.

Foundation for Advanced Tools. The explicit semantic representation and constraint satisfaction framework provide foundations for even more sophisticated development assistance, including architectural evolution, security-aware generation, and collaborative development support.

12.3 Technical Significance

The technical innovations in SEMANTICFORGE establish important precedents for the integration of symbolic reasoning and neural generation:

Hybrid Symbolic-Neural Approaches. Our SMT-guided beam search demonstrates how formal verification can be seamlessly integrated into neural generation without prohibitive computational overhead. This opens new possibilities for constraint-aware AI systems across domains.

Scalable Knowledge Representation. The incremental maintenance of repository knowledge graphs at $O(|\Delta R|)$ complexity shows how explicit semantic representations can remain practical even for large-scale systems. This scalability is crucial for real-world deployment.

Learned Context Selection. Neural query planning provides a general framework for learning to select relevant context from large structured knowledge bases. This approach could generalize to other domains requiring selective information retrieval.

12.4 Limitations and Future Directions

While SEMANTICFORGE achieves significant advances, important limitations remain:

Algorithmic Reasoning. Complex logical problems requiring multi-step reasoning or novel algorithmic insights continue to challenge the system. Future work should investigate integration of symbolic reasoning systems and algorithmic knowledge bases.

Multi-Language Support. Our current focus on Python limits applicability to mixed-language environments common in enterprise development. The concrete roadmap in Section 11.7 outlines specific milestones for Java, TypeScript, and cross-language integration.

Creative Design Tasks. The system's emphasis on consistency and constraint satisfaction may limit creative architectural exploration. Balancing reliability with innovation through configurable constraint relaxation represents an important research direction.

Resource Requirements. The computational and infrastructure requirements for SEMANTICFORGE may limit adoption in resource-constrained environments. Our detailed cost-benefit analysis in Section 10.8 provides guidance for deployment optimization strategies.

12.5 Research Implications

SEMANTICFORGE's success suggests several important directions for future research in automated software development:

Explicit vs. Implicit Knowledge. Our results demonstrate clear advantages for explicit semantic representation over purely implicit approaches. This finding has implications for AI system design beyond code generation, suggesting when symbolic knowledge should complement learned representations.

Constraint Integration in Generation. The effectiveness of SMT-guided decoding indicates that constraint satisfaction should be a primary consideration in generation system design. Future work should explore how different constraint types and solving approaches affect generation quality and efficiency.

Continual Learning for Code Systems. The success of our maintenance agent highlights the importance of continual adaptation in software-related AI systems. As codebases evolve rapidly, static models quickly become obsolete without mechanisms for incremental learning.

Multi-Modal Code Understanding. Repository-level code generation requires understanding text, code structure, execution traces, and architectural patterns. This multi-modal challenge suggests directions for more comprehensive code understanding systems.

12.6 Practical Deployment Considerations

For practitioners considering deployment of SEMANTICFORGE or similar systems, our experience highlights several important factors:

Test Suite Quality. The effectiveness of dynamic analysis correlates strongly with test coverage and quality. Organizations should invest in comprehensive testing infrastructure to maximize benefits from advanced code generation tools.

Incremental Adoption. SEMANTICFORGE's modular architecture enables incremental deployment, allowing organizations to adopt components gradually. Starting with knowledge graph construction and query planning provides immediate benefits with lower risk.

Developer Training. Successful deployment requires training developers to effectively collaborate with AI-assisted generation tools. Understanding system capabilities and limitations is crucial for productive use.

Infrastructure Planning. The computational requirements for SEMANTICFORGE necessitate careful infrastructure planning. Organizations should evaluate resource needs against expected productivity gains.

12.7 Long-Term Vision

SEMANTICFORGE represents progress toward a long-term vision of AI-assisted software development where automated tools understand and respect the principles of software engineering. In this future, developers work collaboratively with AI systems that understand architectural patterns, maintain consistency across large codebases, and generate code that is not only functional but maintainable and well-integrated.

This vision requires continued research across multiple dimensions: better algorithmic reasoning, more sophisticated constraint systems, improved human-AI collaboration interfaces, and deeper understanding of software engineering principles. SEMANTICFORGE provides a foundation for this research by demonstrating that explicit semantic representation and constraint-aware generation can address fundamental limitations of current approaches.

The ultimate goal is not to replace human developers but to amplify their capabilities, enabling them to focus on creative design, architectural innovation, and complex problem-solving while automated systems handle routine implementation tasks with reliability and consistency. SEMANTICFORGE takes an important step toward this collaborative future.

12.8 Closing Remarks

Repository-level code generation represents one of the most challenging problems in automated software development, requiring systems that understand not just local code patterns but global architectural principles and semantic constraints. SEMANTICFORGE demonstrates that this challenge can be addressed through careful integration of symbolic reasoning and neural generation, explicit knowledge representation, and constraint-aware synthesis.

Our results show significant advances in both functional correctness and hallucination reduction, while our open-source release of REPOKG-50 provides a foundation for continued research in this critical area. The techniques and insights from SEMANTICFORGE have broader applicability to other domains requiring constraint-aware generation and semantic consistency.

As AI-assisted development tools become increasingly sophisticated, the principles demonstrated in SEMANTICFORGE –explicit knowledge representation, learned context selection, constraint-aware generation, and continual adaptation—will become essential components of reliable and effective automated development systems. We look forward to continued research building on these foundations to realize the full potential of AI-assisted software engineering.

References

[1] Mark Chen, Jerry Tworek, Christopher Jun, Qiming Zhai, et al. Evaluating large language models trained on code. In *Proceedings of the 38th International Conference on Machine Learning*, 2021. arXiv:2107.03374.

[2] Yicheng Chen, Shiqi Wang, Song Feng, et al. Teaching large language models to self-improve at code generation via retrieval augmented refinement. In *arXiv preprint arXiv:2310.01234*, 2023.

[3] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416, 2022. Accessed: 2025-07-22.

[4] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[5] Yangruibo Ding, Zijian Wang, Wasi Ahmad, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. Cocomic: Code completion by jointly modeling in-file and cross-file context. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3456–3468, 2024. arXiv:2212.10007.

[6] GitHub. Github copilot: Your ai pair programmer. `https://github.com/features/copilot`, 2021. Accessed: 2025-07-14.

[7] Daya Guo, Shuo Ren, Suyuan Lu, Song Feng, Duyu Tang, Shuai Zhang, Zhi Liu, Daya Tang, and Zhi Jin. Graphcodebert: Pre-training code representations with data flow. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, 2021.

[8] Daya Guo, Shuning Zhao, Duyu Tang, et al. Unixcoder: Unified cross-modal pre-training for code understanding and generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[9] Quoc Le, Mark Chen, et al. Coderl: Program synthesis with reinforcement learning. In *arXiv preprint arXiv:2211.00067*, 2022.

[10] Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, and Nan Duan. Coderetriever: A large scale contrastive pre-training method for code search. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2898–2910, 2022.

[11] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[12] Jiawei Liu, Lin Chen, and Hongyu Zhang. Codekg: Building knowledge graphs from code documentation and api references. In *Proceedings of the 29th IEEE International Conference on Program Comprehension (ICPC)*, pages 321–331, 2021.

[13] Meta AI. Code llama: Open foundation models for code. `https://ai.meta.com/blog/code-llama-large-language-model/`, 2023. Accessed: 2025-07-14.

[14] Microsoft Corporation. Typescript language server. urlhttps://github.com/microsoft/TypeScript, 2012. Accessed: 2025-07-22.

[15] Microsoft Corporation. Pylance: Fast, feature-rich language support for python in visual studio code. urlhttps://devblogs.microsoft.com/python/announcing-pylance-fast-feature-rich-language-support-for-python-in-visual-studio-code/, 2020. Accessed: 2025-07-22.

[16] Neo4j Inc. Neo4j: The world's leading graph database. *Technical Report*, 2010. Available at: urlhttps://neo4j.com/.

[17] Ansong Ni, Srini Iyer, Dragomir Radev, Ves Stoyanov, Wen-tau Yih, Sida I. Wang, and Xi Victoria Lin. Lever: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pages 26106–26128, 2023.

[18] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *Proceedings of the 11th International Conference on Learning Representations (ICLR)*, 2023. arXiv:2203.13474.

[19] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. Asleep at the keyboard? assessing the security of github copilot's code contributions. In *Proceedings of the 2022 IEEE Symposium on Security and Privacy*, pages 754–768, 2022. arXiv:2108.09293.

[20] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*, 2022. arXiv:2201.11227.

[21] Ramya Ramakrishnan, Aws Albarghouthi, Somesh Jha, and Thomas Reps. Codeplan: Repository-level coding using llms and planning. arXiv preprint arXiv:2309.12499, 2023.

[22] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. arXiv preprint arXiv:2308.12950, 2023. Accessed: 2025-07-22.

[23] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. Picard: Parsing incrementally for constrained

auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9895–9901, 2021. arXiv:2109.05093.

[24] Divyansh Shrivastava, Hugo Larochelle, and Daniel Tarlow. Rag-code: Retrieval augmented generation for code synthesis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2234–2245, 2023.

[25] Chunqiu Song, Mingyang Zhou, et al. An empirical study of code generation errors made by large language models. In *Workshop on Machine Learning for Programming (ML4P) 2023*, 2023. Available at: https://mapsworkshop.github.io/assets/LLM_Code_Error_Analysis_MAPS2023_camera-ready.pdf.

[26] Shashank Srikant, Benjamin O'Brien, Sebastian Tschiatschek, Eugene Bagdasaryan, Rishabh Rozen, Vikash Krishnamurthy, et al. Generating secure code with language models. In *arXiv preprint arXiv:2306.23034*, 2023.

[27] Mingxi Tang et al. Codeagent: Autonomous programming with conversational software agents. In *arXiv preprint arXiv:2312.13010*, 2023.

[28] Hao Wang, Chen Li, Qian Zhang, and Yang Liu. Programkg: Constructing knowledge graphs from program execution traces for debugging applications. In *Proceedings of the 45th International Conference on Software Engineering (ICSE)*, pages 1832–1843, 2023.

[29] Yue Wang, Hung Le, Akhilesh Gotmare, Nghi Bui, Junnan Li, and Steven Hoi. Codet5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1069–1088, 2023.

[30] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

[31] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R. Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2405.15793.

[32] Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. Repocoder: Repository-level code completion through iterative retrieval and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2471–2484, 2023.

The triple realm of AI-enabled education: from instrumental rationality to value reconstruction

Wan Zihao

Abstract—Artificial Intelligence (AI) is reshaping the education ecosystem, and its development presents the triple realm of "instrumental rationality - integration rationality - value rationality". The instrumental rationality stage optimises teaching efficiency through technology, but faces the risk of alienation of teachers' roles and data privacy; the integration rationality stage emphasises human-computer collaboration to promote personalised learning and educational equity, but needs to solve the problems of technological ethics and system integration; the value rationality stage focuses on the reconstruction of the essence of education, and advocates that AI empowers moral and humanistic education, and balances technological efficiency and humanistic care. The current challenges include algorithmic bias, digital divide and lack of ethical governance, which need to be addressed through interdisciplinary integration, policy synergy and teacher capacity enhancement. In the future, AI education will evolve towards generative intelligence, lifelong learning and sustainable development, and it is necessary to build an inclusive ecosystem to ensure that technology empowers educational equity and social value.

Index Terms—Artificial Intelligence and Education; Reduce Burden and Increase Efficiency; Intelligent Teaching System.

I. The Triple Realm of AI-Enabled Education: Theoretical Basis and Relevance from Instrumental Rationality to Value Reconstruction

1.1 Background and significance of AI-enabled education

The rapid development of Artificial Intelligence (AI) technology is profoundly reshaping the ecological pattern of education. From early intelligent tutoring systems to today's generative AI tools, AI has evolved from the "instrumental rationality" stage of assisting teaching to the "value reconstruction" stage. This transformation is not only reflected in the technological breakthrough, but also in its redefinition of the nature of education. Zhang Jingjing pointed out that the intelligent education of AI should shift from "teaching" to "education", emphasising new teaching modes such as "human-machine co-teaching" and "human-machine symbiosis". It emphasises new teaching modes such as "human-machine co-teaching" and "human-machine symbiosis", and promotes the transformation of education from standardisation to personalisation, and from knowledge transfer to ability cultivation [1]. Liu Sanwu tooth, on the other hand, proposed that the two-way empowerment of AI and education needs to build a "one creation+ two concurrent+ three synergistic+ four fusion" talent training system, in order to achieve a comprehensive reconstruction of the education ecosystem [6]. These theories provide both philosophical and practical support for the triple realm of AI-enabled education.

1.2 The theoretical framework of the triple realm: from instrumental rationality to value reconstruction

The triple realm of AI-enabled education can be summarised as "instrumental rationality - integration rationality - value rationality". **The first realm** (instrumental rationality) takes technology as its core and focuses on the application of AI in teaching efficiency, learning analysis and personalised services. For example, adaptive learning platforms provide customised learning paths through algorithmic analysis of student behavioural data [11]; and intelligent tutoring systems achieve instant feedback and question-answering through natural language processing technology [7]. AI education in this stage is still dominated by "technology empowerment", and its value lies in improving teaching efficiency and learning experience. **The second realm** (integration rationality) emphasises the deep integration of AI and education, breaking through the limitations of a single tool and building a "human-machine collaboration" teaching ecology. Wang Hua proposes that AI-enabled education needs to achieve the three major features of "intelligent teaching mode", "personalized learning mode" and "open educational resources" [12], for example, through the virtual teaching assistant and teacher collaboration to achieve dynamic teaching and learning experience. For example, through the virtual teaching assistant and teacher collaboration, to achieve dynamic teaching adjustment; through big data analysis, accurate identification of students' cognitive blind spot and provide targeted intervention. **The third realm** (value rationality) goes beyond the technical level and returns to the essential value of education. Zhang Jingjing advocates that AI education should take "moral education" as the core, and through the "human-computer co-teaching" mode, transform the technical tools into the carrier of educational value, and promote students to shift from knowledge acquisition to ability cultivation and value shaping [1]. This stage of AI education is no longer limited to technical efficiency, but through the reconstruction of value, to achieve the organic

unity of educational equity, humanistic care and technical rationality.

1.3 Challenges and opportunities in current practice

Although AI-enabled education shows great potential, the realisation of its triple realm still faces multiple challenges. **At the technical level**, the "instrumental rationality" stage of AI education is prone to fall into the misunderstanding of "technology first", neglecting the humanistic attributes of education. For example, some AI systems have insufficient adaptability to disadvantaged groups due to algorithmic bias, exacerbating educational inequality [2]. **At the integration level**, the deep integration of AI and education needs to solve the problem of technical barriers and organisational inertia. Liu Sanwu teeth pointed out that the current AI education application still exists in the three major dilemmas of "difficult to deepen the application of education", "difficult to promote the system change", "difficult to improve the effectiveness of the significant" [3], for example, teachers are not sufficiently accepting of AI tools, and the teachers are not sufficiently accepting of the AI tools. For example, teachers' acceptance of AI tools is insufficient, or schools lack a mechanism for interdisciplinary integration. **At the value level**, the reconstruction of the value of AI education needs to balance the relationship between efficiency and fairness, technology and humanity. Xiao Rui and other scholars emphasise that AI education needs to avoid "technocratic tendencies" and be guided by a "big idea" to integrate AI into the core objectives of education, rather than only being used as a teaching aid [17]. However, these challenges also present opportunities. For example, the openness and scalability of AI technology provides a new path for educational equity, and through projects such as "digitalisation of enrolment" and "digitalisation of international education public services", we can break through geographical and resource constraints [15]. At the same time, the construction of the ethical framework of AI (e.g., transparency, accountability) provides an institutional guarantee for value reconstruction [16]. In the future, the triple realm of AI-enabled education needs to be broken through in the synergistic evolution of technology, education and society, and ultimately realise the paradigm leap from "instrumental rationality" to "value reconstruction".

II. The Triple Realm of AI-Enabled Education Practice Paths and Challenges

2.1 Practical Paths and Limitations of the Instrumental Rationality Stage

In the "instrumental rationality" stage of AI-enabled education, the core value of technology is to improve teaching efficiency and learning experience. Zhang Jingjing pointed out that AI analyses students' behavioural data through intelligent algorithms to achieve personalized learning path design, such as adaptive learning platforms (e.g., Knewton, DreamBox) through real-time feedback to adjust the content of the study, significantly improving learning efficiency. Liu Sanwu Dental further proposes that the application of AI in teaching support should follow the principle of "one creation+ two concurrent+ three synergy+ four integration", i.e., creating new scenarios by means of technology, focusing on the concurrent of teaching content and technology tools, and the synergistic interaction between teachers, students and AI. However, this stage of practice still faces significant limitations. Firstly, technological dependence may lead to the risk of "human-machine substitution", where the teacher's role is marginalised and students' autonomy and creativity are undermined (Hua Wang, 2022). Second, the algorithmic bias of AI systems may exacerbate educational inequalities, for example, in resource-poor areas, AI tools may not be able to effectively support disadvantaged groups due to insufficient data quality (Liu, 2023). In addition, the standardised design of technological tools may ignore the diverse needs of education, leading to a "one-size-fits-all" teaching model (Zhang Jingjing, 2021).

2.2 Synergistic mechanisms and innovative models at the stage of integrative rationality

Entering the stage of "integration rationality", the deep integration of AI and education has become the key. Wang Hua proposed that AI-enabled education needs to achieve the three major features of "intelligent teaching mode", "personalised learning mode" and "open educational resources", such as through virtual teaching assistants (such as Squirrel AI) and teachers to collaborate to achieve dynamic teaching adjustments; through big data analysis to accurately identify students' cognitive blind spots and provide targeted support. For example, through virtual teaching assistants (such as Squirrel AI) and teachers to collaborate to achieve dynamic teaching adjustment; through big data analysis, accurately identify students' cognitive blind spots and provide targeted intervention. Liu Sanwu teeth stressed that AI education needs to break through the limitations of a single tool to build a "human-computer collaboration" teaching ecology, for example, through the "dual-teacher classroom" model (teacher + AI assistant), to achieve the depth of the teaching content mining and personalised feedback. However, this stage of practice also faces challenges. Firstly, the deep integration of technology and education needs to address organisational inertia, such as teachers' lack of acceptance of AI tools, or schools' lack of mechanisms for cross-disciplinary integration (Liu Sanyu Dental, 2023). Secondly, the lack of openness and scalability of AI systems may lead to the phenomenon of "technology silos", such as poor data interoperability between different AI platforms, which restricts the sharing of educational resources (Zhang Jingjing, 2022). In addition, the synergy between technology and education needs to balance efficiency and fairness, for example, the "digital divide" problem of AI in resource allocation may further aggravate the inequality in education (Wang Hua, 2021).

2.3 Ethical Reflection and Future Prospects of the Value Rationality Stage

In the "value rationality" stage, the core goal of AI education shifts from technical efficiency to the reconstruction of educational value. Xiao Rui and other scholars pointed out

that AI-enabled education needs to avoid "technocratic tendencies", and needs to be guided by the "big idea" to integrate AI into the core objectives of education, such as through the "human-machine co-teaching" model, which transforms technical tools into carriers of educational value, and promotes students' transformation from knowledge acquisition to competence to reconstruction. For example, through the "human-machine co-teaching" model, the technical tools are transformed into carriers of educational value, and students are promoted to shift from the acquisition of knowledge to the cultivation of abilities and the shaping of values (Zhang Jingjing, 2023). Liu Sanwu Dental further proposes that the reconstruction of the value of AI education needs to build a breakthrough path of "difficult to deepen the application of education", "difficult to promote the systematic change" and "difficult to improve the effectiveness significantly", for example For example, through the "digitalisation of enrolment and study" and "digitalisation of international education public services", we can break through the geographical and resource restrictions (Liu, 2023). However, the practice of at this stage still faces ethical challenges. Firstly, the "black box effect" of AI in education may lead to a crisis of trust, for example, students and parents have doubts about the transparency of AI decision-making (Wang Hua, 2022). Second, the "algorithmic bias" of AI in education may exacerbate social division, for example, AI may discriminate against disadvantaged groups due to data bias in the process of enrolment and evaluation (Zhang Jingjing, 2021). In addition, the value reconstruction of AI education needs to balance the relationship between technology and humanities, for example, through the "ethical framework of education" (e.g., transparency, accountability) to provide institutional protection for AI education (Xiao Rui, 2023).

2.4 Articulation and Future Trends in the Triple Realm

The triple realm of AI-enabled education does not exist in isolation, but is gradually promoted through the synergistic evolution of technology, education and society. From "instrumental rationality" to "integration rationality" to "value rationality", the practical path of AI education needs to establish a dynamic balance between technological breakthroughs and educational concepts. For example, Zhang Jingjing points out that the "value reconstruction" of AI education needs to take the "moral education" as the core, and through the "human-machine co-teaching" mode, the technical tools should be transformed into the carrier of educational value (Zhang Jingjing, 2023). (Zhang Jingjing, 2023). Liu Sanya emphasises that the future of AI education needs to achieve a comprehensive restructuring of the education ecosystem through the breakthrough path of "difficult to deepen the application of education", "difficult to promote the systematic change" and "difficult to improve the effectiveness" (Liu Sanya, 2023). Comprehensive reconstruction of the education ecosystem (Liu Sanwuya, 2023). The construction of "education ethics" to ensure the sustainable development of AI education.

III: Ethical Challenges and Governance Paths for AI-Enabled Education

3.1 Ethical dilemmas of AI in education

While the widespread use of AI in education has brought many conveniences, it has also raised a series of ethical challenges. Firstly, **privacy and data security** issues are becoming increasingly prominent: AI systems rely on a large amount of student data for personalised learning and assessment, but the collection, storage and use of such data may infringe on students' privacy rights. For example, Zhang Jingjing pointed out that the "black box effect" of AI in education may cause students to feel uneasy about the opacity of data use, which in turn affects their sense of trust [1]. Secondly, the problem of **algorithmic bias** is also a cause for concern, as AI systems often make decisions based on historical data, which may contain biases such as gender, race, and economic background, leading to unfair educational outcomes. For example, Wang et al. point out that the application of AI in education may lead to "algorithmic discrimination", which may exacerbate social inequality [5]. In addition, **the alienation of teachers' roles** is also one of the important ethical issues, and the introduction of AI may weaken the dominant position of teachers, lead to the alienation of teacher-student relationships, and even affect students' social skills and emotional development [16].

3.2 The Necessity and Practical Path of AI Ethics Education

In the face of the above ethical challenges, AI ethics education has become an important issue in the field of education.AI ethics education not only involves ethical design at the technical level, but should also be integrated into the core objectives of education, such as developing students' critical thinking, moral judgement, and digital literacy. For example, Jason Borenstein and Ayanna Howard point out that AI developers need to be ethically aware to ensure the appropriate use of technology [10]. In addition, AI ethics education should be integrated throughout the educational process, including curriculum design, teaching methods, and assessment systems. For example, Fajar Alamin and Sofyan Sauri emphasised that AI ethics education should be centred on "building moral values", and that students' and teachers' ethical awareness should be enhanced through curriculum reform and teacher training [16].

At the practical level, AI ethics education can be realised in a number of ways. First, **curriculum design** should incorporate AI ethics content into the teaching of various disciplines, for example, by introducing AI ethics modules in computer science, social science and humanities programmes. Second, **teacher training** should enhance teachers' understanding of AI ethics so that they can effectively guide students to think about the moral implications of AI. For example, Hu et al.

proposed that a guide to ethical norms for educational AI should be prepared and their intelligence education literacy should be enhanced through teacher training [14]. In addition, **student engagement** is also an important part of AI ethics education. For example, Sara Saylam et al. stated that educators should guide students to understand the potential impacts of AI and develop their ethical awareness and digital citizenship [9].

3.3 Framework and Policy Recommendations for Ethical Governance of AI

In order to effectively address the ethical challenges of AI in education, it is necessary to construct a sound framework for the ethical governance of AI. First, **policymaking** should clarify the ethical boundaries of AI in education, for example, through legislation that ensures the privacy protection of student data and regulates the fairness of AI systems. For example, Afshan Bibi et al. suggest that student rights protection should be strengthened, algorithmic and data biases should be addressed, and the impact of AI on educational equity and inclusion should be monitored [6]. Second, **technology governance** should focus on the transparency and interpretability of AI systems, for example, through algorithmic auditing and publicising the decision-making process to enhance students' and teachers' trust in AI. For example, Liheng Yu and Zhonggen Yu pointed out that AI ethics should include principles such as transparency, justice, fairness, impartiality, harmlessness, responsibility, and privacy [7]. In addition, **interdisciplinary cooperation** is also an important direction for AI ethics governance. For example, Ana María Alonso-Rodríguez emphasised that AI ethics should integrate scientific, technological and cultural-social factors to guide the application of AI in education [11].

At the policy level, governments and educational institutions should develop appropriate ethical norms for AI and promote their implementation in educational practices. For example, Celalettin Çelevi et al. pointed out that a guide to ethical norms of AI in education should be prepared and students' ethical awareness should be promoted through public interest learning resources [12]. In addition, **international co-operation** is also an important direction for AI ethical governance. For example, the draft proposal for AI ethics proposed by UNESCO emphasises the unification and coordination of AI ethical standards globally [1].

IV. Future Trends and Interdisciplinary Integration Paths of AI-Enabled Education

4.1 Generative AI-driven paradigm shift in education

Generative AI is becoming the core force driving the change of education paradigm.Hao Yu and Yunyun Guo (2023) pointed out that generative AI can dynamically generate personalized learning content based on students' learning data and behavioral patterns through natural language processing and deep learning technology, realizing the accurate teaching and learning according to students' abilities [5]. " precision teaching [5]. For example, AI can generate customised learning paths based on students' interests or provide immersive experimental experiences through virtual labs, thus breaking through the time and space constraints of traditional classrooms. In addition, generative AI can enhance learning motivation through gamification design, for example, combining augmented reality (AR) technology to construct interactive learning scenarios to visualise abstract concepts and enhance students' cognitive engagement [5].

4.2 New Pathways for Equitable and Inclusive Development in Education

The popularity of AI technology provides technical support for educational equity, but its application still needs to address the uneven distribution of resources and the digital divide. Hu Xiaoyong et al. (2022) proposed that high-quality educational resources should be sunk into remote areas through projects such as "digitisation of enrolment" and "digitisation of public services in international education", so as to realise the equalisation of educational opportunities [13]. For example, AI-driven distance learning platforms can provide rural students with the same quality of curriculum resources as urban students, and at the same time break down language barriers and promote multicultural integration through voice recognition and real-time translation technology [15]. However, technological empowerment needs to be synergised with policy design, such as the establishment of "Guidelines for Ethical Regulation of AI in Education", to ensure the universality and safety of technological applications [15].

4.3 Interdisciplinary Integration and Reconstruction of Educational Ecology

The deep integration of AI and education needs to break through the disciplinary barriers and build a new educational ecology of "human-machine collaboration". Liu Bangqi and He Sheng (2021) proposed that a multi-level AI education system should be constructed with the framework of "one goal, four levels, and two types of training", covering the differentiated needs of basic education, vocational education and higher education [14]. For example, in the basic education stage, AI can assist teachers in designing interdisciplinary curricula (e.g., STEAM education), and integrating maths, science and art content through knowledge mapping technology; in higher education, AI can provide researchers with intelligent literature analysis and experimental design support to promote academic innovation [17]. In addition, the combination of educational robots and virtual teachers can provide personalised support for special education groups, e.g. identifying students' emotional states and adjusting teaching strategies through affective computing technologies [17].

4.4 Policy synergies and global governance frameworks

The sustainable development of AI-enabled education depends on policy guidance and international cooperation. Huang Ronghuai et al. (2021) emphasised that the government should adopt the proposition of "two-way empowerment of science

and technology and education" to formulate the development plan of AI education, for example, setting up a special fund to support the research and development of AI education products , and establishing a collaborative mechanism of "government-enterprise-academia-research" to promote the technology to the ground [7]. [7]. Meanwhile, the draft AI ethics proposal (2023) proposed by UNESCO provides a reference for global governance, such as avoiding technological monopoly and data barriers through standardised data sharing agreements [15]. In the future, the governance of AI in education needs to take into account technological innovation and humanistic care, such as building a dynamic simulation system of educational data through the "digital twin strategy" to achieve risk warning and decision optimisation [4].

4.5 Challenges and Opportunities in the Education 4.0 Era

Education 4.0 (Education 4.0), as the future form of AI and education integration, faces the uncertainty of technology iteration, but also contains unlimited possibilities.João Fernando Costa Júnior et al. (2023) pointed out that AI education needs to address the core issues of "algorithmic bias" and "data privacy", such as protecting student data privacy through federated learning technology, and improving model generalisation ability by using transfer learning [17]. João Fernando Costa Júnior et al. (2023) pointed out that AI education needs to address core issues such as "algorithmic bias" and "data privacy", for example, by protecting student data privacy through federated learning techniques, and by using transfer learning to improve the generalisation of models [17]. In addition, Education 4.0 needs to reconfigure the role of the teacher, so that it is transformed from a "knowledge transmitter" to a "learning guide", through AI tools to assist in the design of inquiry-based learning tasks, to cultivate students' critical thinking and innovation [12]. Ultimately, the triple realm of AI-enabled education will move from "instrumental rationality" to "value rationality", and achieve a comprehensive reconstruction of the education ecosystem through the synergistic evolution of technology, education and society [1].

V. The Future Picture and Sustainable Development Path of AI-Enabled Education

5.1 A systematic summary of the three realms of AI education

The triple realm of AI-enabled education - from "instrumental rationality" to "value reconstruction" - is not only the inevitable result of technological evolution, but also a deep mapping of educational philosophy and social values. It is not only the inevitable result of the evolution of technology, but also a deep mapping of educational philosophy and social value. Zhang Jingjing (2021) pointed out that the "value reconstruction" of AI education needs to take "moral education" as the core, and through the "human-machine co-teaching" mode, the technical tools are transformed into carriers of educational value, promoting students from knowledge to education. Through the mode of "human-machine co-teaching", the technological tool is transformed into a carrier of educational value, and students are promoted to shift from knowledge acquisition to ability cultivation and value shaping [1]. Liu Sanwuya (2023) further proposes that the "trilogy" progression pattern of AI education - technological empowerment, technological innovation, and technological reshaping - reveals the transformation of the education ecology from "standardisation" to "standardisation". "standardisation" to "personalisation" to "ecology" [16]. This process not only relies on technological breakthroughs, but also requires the innovation of educational concepts and the synergy of social systems.

5.2 Challenges and Directions for Breakthroughs in Current AI Education Practices

Although AI education shows great potential, its development still faces multiple challenges. First of all, the issues of **technology ethics and data security** are prominent.Neil Selwyn (2022) emphasises that the application of AI in education needs to be alert to risks such as "algorithmic bias", "data privacy", "social inequality", etc. [4]. "For example, generative AI may discriminate against disadvantaged groups due to biased training data [4]. Secondly, the realisation of **equity in education** still needs a breakthrough. The UNESCO report points out that AI education needs to narrow the education divide between urban and rural areas, regions and groups through strategies such as "inclusive data systems" and "open sharing of digital resources" [12]. In addition, **the transformation of teachers' roles** and the "human-machine trust crisis" are also key challenges. Huang Ronghuai et al. (2021) suggest that teachers need to transform from "knowledge transmitters" to "learning facilitators", and the "black box effect" of AI tools may weaken the depth of teacher-student interaction [18]. The "black box effect" of AI tools may weaken the depth of teacher-student interaction [18].

5.3 Sustainable Paths for the Future of AI Education

For the future, the sustainable development of AI education needs to be promoted from the three aspects of technology, education and society. **On the technical level**, the transparency and interpretability of AI systems should be strengthened, for example, through "explainable AI" (XAI) technology to enhance the trust between teachers and students [20]. Meanwhile, the cultivation of "emotional intelligence" and "AI literacy" of generative AI should be the focus of research to address the cognitive and ethical challenges in human-computer interaction [14]. **At the education level**, a new teaching ecology of "human-computer collaboration" should be constructed, for example, through the mode of "dual-teacher classroom" and "virtual teacher", to achieve the balance between personalised learning and social-emotional development [3]. Balance of personalised learning and social-emotional development [3]. **At the social level**, it is necessary to establish an education-technology-policy linkage mechanism, for example, through the "Code of Ethics for AI

in Education" and "Open Platform for Digital Resources", to promote the universality and fairness of AI education [4]. The social level needs to establish an "education-technology-policy" linkage mechanism, for example, through the "code of ethics for AI in education" and "open platform for digital resources" to promote the universality and fairness of AI education [12].

5.4 Future Research Directions and the Future Picture of Policy

Future research on AI education should focus on the following directions: first, interdisciplinary integration, for example, combining education, psychology and computer science to explore the application of AI in special education, lifelong learning and other fields [16]; second, ethical governance, to build a global standard covering data privacy, algorithmic fairness and social impacts [12]; and third, educational equity, to optimise resource allocation strategies through the "digital twin" technology to simulate different educational scenarios [19]. Third, educational equity, through "digital twin" technology to simulate different educational scenarios and optimise resource allocation strategies [19]. At the policy level, initiatives such as "education AI special fund", "teachers' AI literacy training" and "education data open platform" should be promoted to ensure the sustainable development of AI education [17]. The triple realm of AI-enabled education is not only a leap in technology, but also a reconstruction of the nature of education. From "instrumental rationality" to "value reconstruction", AI education needs to adhere to the humanistic care in the technical empowerment, and achieve fairness and justice in the systematic change. In the future, only through the synergistic evolution of technology, education and society can we build a new ecosystem of AI education that is "high-quality and warm", and truly realise the educational ideal of "making people moral" [16].

VII. Conclusion

With the continuous development of artificial intelligence, its application fields continue to expand, and will be more widely used in primary and secondary school teaching in the future, artificial intelligence will be more deeply integrated into the field of primary and secondary school teaching, and artificial intelligence will develop towards intelligence, personalization and fairness. As a tool for teachers to carry out teaching work, artificial intelligence can tailor personalized learning plans for students, and to a certain extent, it can also promote the balanced allocation of educational resources and contribute to the fair development of educational resources. In the face of the development of AI technology, the education sector should seize the opportunity to tap the potential of AI technology to achieve sustainable development of primary and secondary education.

REFERENCES

[1] Acevedo, K. (2025). Exploring the Impact of Generative AI to Mitigate Educator Burnout. *Electronic Theses and Dissertations*. https://digitalcommons.acu.edu/etd/925

[2] Afifah, R. N., Simanullang, T., & Madhakomala, R. (2022). VARIOUS ADVANTAGES IN EDUCATION. *International Journal of Business, Law, and Education*, *3*(2), 145–156. https://doi.org/10.56442/ijble.v3i3.65

[3] Ahmed, F. (2024). The Digital Divide and AI in Education: Addressing Equity and Accessibility. *AI EDIFY Journal*, *1*(2), 12–23.

[4] Almethen, a. (2024). Challenges in implementing artificial intelligence applications in secondary-level education: A teacher-centric perspective.), 0–0. https://doi.org/10.21608/mfes.2024.270936.1776

[5] Cheshmehzangi, A., & Tang, T. (2024). Changsha: The PuDong of Western China Through Regional Synergy and Technological Innovation. In A. Cheshmehzangi & T. Tang (Eds), *China Under Construction: Shaping Cities Through Recent Urban Transformation* (pp. 33–57). Springer Nature. https://doi.org/10.1007/978-981-97-9785-1_3

[6] Chou, C.-M., Shen, T.-C., Shen, T.-C., & Shen, C.-H. (2022). Influencing factors on students' learning effectiveness of AI-based technology application: Mediation variable of the human-computer interaction experience. *Education and Information Technologies*, *27*(6), 8723–8750. https://doi.org/10.1007/s10639-021-10866-9

[7] Davis, R. O. (2024). Korean in-Service Teachers' Perceptions of Implementing Artificial Intelligence (AI) Education for Teaching in Schools and Their AI Teacher Training Programs. *International Journal of Information and Education Technology*, *14*(2), 214–219. https://doi.org/10.18178/ijiet.2024.14.2.2042

[8] Dieterle, E., Dede, C., & Walker, M. (2024). The cyclical ethical effects of using artificial intelligence in education. *AI & SOCIETY*, *39*(2), 633–643. https://doi.org/10.1007/s00146-022-01497-w

[9] Gerlich, M. (2025). AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking. *Societies*, *15*(1), 6. https://doi.org/10.3390/soc15010006

[10] Herold, B. (2013, December 4). Custom Software Helps Cities Manage School Choice. *Education Week*. https://www.edweek.org/leadership/custom-software-helps-cities-manage-school-choice/2013/12

[11] Kong, L., Hu, C., Huang, L., Zhang, Y., Huang, W., & Huang, S. (2025). *The Double-Edged Effect of AI Use on Innovation Teaching Behavior among Primary and Secondary School Teachers in China: A Job Demands–Resources Perspective*. Research Square. https://doi.org/10.21203/rs.3.rs-6864947/v1

[12] Kong, Y. (2021). The Role of Experiential Learning on Students' Motivation and Classroom Engagement. *Frontiers in Psychology*, *12*. https://doi.org/10.3389/fpsyg.2021.771272

[13] Nemani, S. (2025). Evaluating the Impact of Artificial Intelligence on Reducing Administrative Burden and Enhancing Instructional Efficiency in Middle Schools. *Current Perspectives in Educational Research*, *8*(1), 1–16. https://doi.org/10.46303/cuper.2025.1

[14] Wei, C., & Liu, P. (2023). Artificial Intelligence Enabled Double Reduction Policy Path Analysis. *SHS Web of Conferences*, *178*, 03014. https://doi.org/10.1051/shsconf/202317803014

[15] Wu, M., Chen, R., Lv, Y., Wu, Y., & Qiu, Y. (2022). Driving forces in joint training of industry-education graduate students under regional innovation ecosystem: – A case study of Yibin. *Proceedings of the 5th International Conference on Big Data and Education*, 236–240. https://doi.org/10.1145/3524383.3524448

[16] Yang, S. J. H., Ogata, H., Matsui, T., & Chen, N.-S. (2021). Human-centered artificial intelligence in education: Seeing the invisible through the visible 见. *Computers and Education: Artificial Intelligence*, *2*, 100008. https://doi.org/10.1016/j.caeai.2021.100008

A Greedy Approximation for Minimum Cardinality Multiple Quasi-submodular Cover with Applications

Qi Zhang[1], and Hao Zhong[2]*

[1]School of Information Technology and Engineering, Guangzhou College of Commerce, Guangzhou 511363, China

[2]School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou, 510665, China

*Corresponding author: hzhong@gpnu.edu.cn

Abstract

Quasi-submodularity is a unified measurement to characterize submodularity or approximate submodularity of a set function. In this paper, we consider a variant of Minimum Cardinality Cover named Minimum Cardinality Multiple Quasi-submodular Cover, which requires to find the smallest subset of given ground set such that multiple quasi-submodular functions reach a maximum. We design and analyze a greedy approximation algorithm for Minimum Cardinality Multiple Quasi-submodular Cover. As some applications of our results, we achieve a O($\ln n$)-approximation algorithm for Minimum Resolving Restrained Dominating Set, and a O($\ln \delta$)-approximation algorithm for Minimum Dominating Set in multiplex networks, where n is the node number and δ is the maximum node degree of the input graph.

Index Terms— Minimum cardinality cover, Submodular, Approximation algorithm, Dominating set

1 Introduction

Minimum Cardinality Cover(MCC for short) is a well-known NP-hard combinatorial optimization problem. Given a ground set U and a set function f:$2^U \to \mathbb{R}^+$ which is normalized ($f(\emptyset) = 0$) and non-decreasing ($f(T) \geq f(R)$ whenever $T \subseteq R \subseteq U$), MCC is asked to find a smallest subset $S \subseteq U$ such that $f(S) = f(U)$, more formally,

$$\min_{S \subseteq U} \{ \mid S \mid : f(S) = f(U) \}.$$

For any pair of T, $R \subseteq U$, the marginal gains of set R in T is denoted by $\Delta_R f(T) = f(R \cup T) - f(T)$. Specially, denote by $\Delta_i f(T) = f(T \cup \{i\}) - f(T)$ the marginal gains of singleton set $\{i\}$ in T. The classical greedy algorithm(CGA for short) is a basic method for MCC: starts from an empty set; iteratively adds to the current solution set one element with the maximum marginal gains until potential function $f(\cdot)$ reaches the maximum.

Especially, suppose potential function $f(\cdot)$ satisfies submodularity or approximate submodularity, then algorithm CGA enjoys some provable approximation guarantees for these variants of MCC. Now we introduce the submodularity and two approximate submodularities.

Algorithm 1 CGA

Input: A ground set U and potential function f:$2^U \to \mathbb{R}^+$
Output: A subset $S \subseteq U$ such that $f(S) = f(U)$

```
S ← ∅;
while ∃u ∈ U such that Δ_u f(S) > 0 do
    u ← arg max_{u∈U\S} Δ_u f(S);
    S ← S ∪ {u};
end while
return S.
```

(i): Submodularity [1, 2, 3]. The submodularity of $f(\cdot)$ is that the marginal gains of joining an element to a set is not less than that of joining an element to a superset. More formally, for any $T \subseteq R \subseteq U$ and any $i \in U \backslash R$, $f(\cdot)$ satisfies

$$\Delta_i f(T) \geq \Delta_i f(R).$$

(ii): Submodularity Ratio [4, 5]. The submodularity ratio is a approximate submodularity which is used to characterize how close a set function is to be submodular. More formally, the submodularity ratio of $f(\cdot)$ is the largest scalar $\gamma \in (0,1]$ such that for any $T \subseteq R \subseteq U$ and any $i \in U \backslash R$, $f(\cdot)$ satisfies

$$\Delta_i f(T) \geq \gamma \times \Delta_i f(R).$$

(iii): Submodularity Gap [5, 6]. The submodularity gap is another approximate submodularity which is also used to characterize how close a set function is to be submodular. More formally, the submodularity gap of f is the smallest scalar $\theta \in [0,+\infty)$ such that for any $T \subseteq R \subseteq U$ and any $i \in U \backslash R$, $f(\cdot)$ satisfies

$$\Delta_i f(T) \geq \Delta_i f(R) - \theta.$$

Submodularity and approximate submodularity play an important role in combinatorial optimization, especially the Minimum Cardinality Cover. However, some combinatorial optimization problems require multiple submodular or approximate submodular functions reach a maximum. These problems are usually NP-hard, which means it is unlikely to be solved precisely in polynomial time unless $P = NP$. Greedy approximation algorithm might be very popular to solve the problem, because it enjoy low time complexity and thus can

be suitable for big data, and particularly because it can give solutions with guaranteed quality, which are close to the best that could theoretically be obtained. The contributions of this paper are as follows.

- Define a measurement which is called quasi-submodularity to uniformly characterize submodularity and some approximate submodularities.
- Design and analyze a greedy approximation algorithm for Minimum Cardinality Multiple Quasi-submodular Cover. It can be proved that our approximation ratio slightly generalizes some known ones.
- Achieve a O($\ln n$)-approximation algorithm for the minimum resolving restrained dominating sets problem, and a O($\ln \delta$)-approximation algorithm for the minimum dominating sets problem in multiplex networks, where n is the node number and δ is the maximum node degree of the input graph.

2 Preliminaries

First, we give a measurement which is called quasi-submodularity to unified characterize submodularity and some approximate submodularities.

Definition 1. *(Quasi-submodularity). Given a ground set U and a set function f:$2^U \to \mathbb{R}^+$, the quasi-submodularity is binary parameters (λ, μ) such that for any $T \subseteq R \subseteq U$ and any $i \in U \backslash R$,*

$$\Delta_i f(T) \geq \lambda \times \Delta_i f(R) - \mu,$$

where $\lambda \in (0, 1]$ and $\mu \in [0, +\infty)$.

For Convenience, henceforth we say that a function $f(\cdot)$ is (λ, μ)-submodular if its quasi-submodularity is (λ, μ). It is easy to see that if $f(\cdot)$ satisfies (λ, μ)-submodularity, then $f(\cdot)$ satisfies (λ', μ')-submodularity for any $\lambda' \in (0, \lambda]$ and $\mu' \in [\mu, +\infty)$.

Remarks:

(i) $(1, 0)$-submodularity is the classic submodularity,

(ii) $(\lambda, 0)$-submodularity is called submodularity ratio of $f(\cdot)$ if and only if λ is the largest scalar such that $f(\cdot)$ satisfies $(\lambda, 0)$-submodularity,

(iii) $(1, \mu)$-submodularity is called submodularity gap of $f(\cdot)$ if and only if μ is the smallest scalar such that $f(\cdot)$ satisfies $(1, \mu)$-submodularity.

Next, we consider a variant of Minimum Cardinality Cover. The variant requires multiple submodular or approximate submodular functions reach a desirable fraction.

Definition 2. *(Minimum Cardinality Multiple Quasi-Submodular Cover). Given a ground set U and multiple set functions $f_1, ..., f_k$:$2^U \to R^+$, where f_i is normalized, non-decreasing and (λ_i, μ_i)-submodular for any $i \in \{1, 2, ..., k\}$, Minimum Cardinality Multiple Quasi-Submodular Cover is asked to find a smallest subset $S \subseteq U$ such that $f_i(S) = f_i(U)$ for any $i \in \{1, 2, ..., k\}$, more formally,*

$$\min_{S \subseteq U} \{ \mid S \mid : f_i(S) = f_i(U), i \in \{1, 2, ..., k\}\}.$$

Since that Minimum Cardinality Multiple Quasi-submodular Cover generalizes the problem of covering multiple submodular constraints [7] if (λ_k, μ_k)-submodularity is submodularity for any $i \in \{1, 2, ..., k\}$, then Minimum Cardinality Multiple Quasi-submodular Cover is NP-hard, which means it is unlikely to be solved precisely in polynomial time unless $P = NP$. Our aim is to give a greedy approximation algorithm with polynomial time.

3 Greedy approximation algorithm

For Minimum Cardinality Multiple Quasi-submodular Cover, we give a solution with guaranteed quality by considering a potential function $f : 2^U \to \mathbb{R}^+$ for any $S \subseteq U$, where

$$f(S) = \sum_{i=1}^{k} f_i(S).$$

It is easy to see $f(\cdot)$ is normalized and non-decreasing due to $f_i(\cdot)$ is normalized and non-decreasing for any $i \in \{1, 2, ..., k\}$. Further, we give some properties of $f(S)$ for any $S \subseteq U$.

Lemma 1. *The potential function $f(\cdot)$ satisfies*
$(i)(\lambda, \mu)$-submodularity, where $\lambda = \min\{\lambda_1, \lambda_2, ..., \lambda_k\}$ and $\mu = \sum_{i=1}^{k} \mu_i$,
(ii) $f(S) = f(U)$ if and only if S is a solution for Minimum Cardinality Multiple Quasi-submodular Cover, and
(iii) $f(S) < f(U)$ if and only if there exists a element $u \in U - S$ and $i \in \{1, 2, ...k\}$ such that $\Delta_u f_i(S) > 0$.

Proof. (i)Since $f_i(\cdot)$ is (λ_i, μ_i)-submodular for any $i \in \{1, 2, ..., k\}$, we have for any $A \subseteq B \subseteq U$ and any $u \in U - B$

$$\begin{aligned}
\Delta_u f(A) &= \sum_{i=1}^{k} \Delta_u f_i(A) \\
&\geq \sum_{i=1}^{k} (\lambda_i \times \Delta_u f_i(B) - \mu_i) \\
&> \lambda \times (\sum_{i=1}^{k} \Delta_u f_i(B)) - \mu \\
&= \lambda \times \Delta_u f(B) - \mu,
\end{aligned}$$

where $\lambda = \min\{\lambda_1, \lambda_2, ..., \lambda_k\}$ and $\mu = \sum_{i=1}^{k} \mu_i$.

$(ii) f(S) = f(U)$, namely, $f_i(S) = f_i(U)$ for $i \in \{1, 2, ..., k\}$ if and only if S is a solution for Minimum Cardinality Multiple Quasi-submodular Cover.

$(iii) f(S) < f(U)$, namely, there exist $u \in U - S$ such that $\Delta_u f(S) > 0$, which is equivalent to the existence of $i \in \{1, 2, ..., k\}$ and $u \in U - S$ such that $\Delta_u f_i(S) > 0$. □

Next, we prove that algorithm CGA yields a approximation solution for Minimum Cardinality Multiple Quasi-submodular Cover.

Theorem 1. *If $f(\cdot)$ is integer-valued, there exists a $(1+\lambda\mu+\frac{1}{\mu}\ln\alpha)$-approximation solution yielded by CGA for finding a subset $S \subseteq U$ such that $f(S) = f(U)$, where $\alpha = \max_{u\in U} f(\{u\})$.*

Proof. Suppose that the greedy solution yielded by CGA is $S = \{s_1, s_2, ..., s_g\}$, where g is the cardinality of the greedy solution and s_i is the element selected in the ith iteration of the algorithm for $i = 1, 2, ..., g$. And suppose $S_i = \{s_1, s_2, ..., s_i\}$ for $i = 1, 2, ..., g$ and $S_0 = \emptyset$. Similarly, we let $O = \{o_1, o_2, ..., o_{opt}\}$ be an optimal solution where *opt* is its cardinality, and let $O_j = \{o_1, o_2, ..., o_j\}$ for $j = 1, 2, ..., opt$ and $O_0 = \emptyset$.

From greedy rule, we have

$$s_i = \arg\max_{u\in U\setminus S_{i-1}} \Delta_u f(S_{i-1}),$$

namely, for any $i \in \{0, 1, ..., g-1\}$ and any $j \in \{1, 2, ..., opt\}$,

$$f(S_{i+1}) - f(S_i) = \Delta_{s_{i+1}} f(S_i) \geq \Delta_{o_j} f(S_i),$$

and thus we have

$$f(S_{i+1}) - f(S_i) \geq \frac{1}{opt}\sum_{j=1}^{opt} \Delta_{o_j} f(S_i).$$

Additionally, for any $i \in \{0, 1, ..., g\}$, we have

$$f(U) - f(S_i) = f(S_i \cup O_{opt}) - f(S_i) = \sum_{j=1}^{opt} \Delta_{o_j} f(S_i \cup O_{j-1}).$$

From lemma 1(i), we have

$$\Delta_{o_j} f(S_i) \geq \lambda \times \Delta_{o_j} f(S_i \cup O_{j-1}) - \mu,$$

where $\lambda = \min\{\lambda_1, \lambda_2, ..., \lambda_k\}$ and $\mu = \sum_{i=1}^{k} \mu_i$. We immediately establish the following recursive inequality for any $i \in \{0, 1, ..., g-1\}$

$$f(S_{i+1}) - f(S_i) \geq \frac{\lambda}{opt}(f(U) - f(S_i) - \mu \times opt),$$

which implies

$$\lambda\times(\frac{f(U)}{opt} - \mu) \leq f(\{s_1\}) - f(\emptyset) \leq \Delta_{s_1} f(\emptyset) = \max_{u\in U} f(\{u\}).$$

Let $c_{i-1} = \lambda\times(f(U) - f(S_{i-1}) - \mu \times opt)$ and $\alpha = \max_{u\in U} f(\{u\})$, we have

$$c_g = -\lambda \times \mu \times opt$$

and

$$c_0 = \lambda\times(f(U) - \mu\times opt) = \lambda\times(\frac{f(U)}{opt} - \mu)\times opt \leq \alpha\times opt.$$

For any $i \in \{1, 2, ..., g\}$, another recursive inequality is established as

$$c_{i-1} - c_i = \lambda \times \Delta_{s_i} f(S_{i-1}) \geq \frac{\lambda \times c_{i-1}}{opt},$$

which implies

$$c_i \leq c_0(1 - \frac{\lambda}{opt})^i \leq c_0 e^{-\frac{\lambda\times i}{opt}}.$$

Note that since c_i is decreasing with respect to i, there exists $x \in \{1,2,...,g-1\}$ such that for any $\eta \in (0, \alpha)$ we have $c_x = (\eta - \lambda \times \mu) \times opt$. From c_x to c_g, c_x will reduce from $(\eta - \lambda \times \mu) \times opt$ to $-\lambda \times \mu \times opt$ with $g - x$ elements are selected by CGA. Otherwise, by assuming that $f(\cdot)$ is integer-valued, c_x reduces 1 at least when any element in $U\setminus S_x$ is selected. Therefore, we have $g \leq \eta \times opt + x$ from

$$opt \times (\eta - \lambda \times \mu) - (g - x) \geq -\lambda \times \mu \times opt.$$

By $c_x \leq c_0 e^{-\frac{\lambda\times x}{opt}}$, the relationship between x and *opt* can be obtained as

$$\begin{aligned} x &\leq \frac{opt}{\lambda} \times \ln\frac{c_0}{c_x} \\ &\leq \frac{opt}{\lambda} \times \ln\frac{\alpha \times opt}{opt \times (\eta - \lambda \times \mu)} \\ &\leq \frac{opt}{\lambda} \times \ln\frac{\alpha}{\eta - \lambda \times \mu}. \end{aligned}$$

Hence,

$$g \leq \eta \times opt + x \leq opt \times (\eta + \frac{1}{\lambda}\ln\frac{\alpha}{\eta - \lambda \times \mu}).$$

In order to get the lower bound of approximation ratio, we consider function $p(\eta) = \eta + \frac{1}{\lambda}\ln\frac{\alpha}{\eta-\lambda\times\mu}$ is minimal at $\eta = 1 + \lambda \times \mu$. Since $\eta \in (0, \alpha)$, we can simplify the approximation ratio to $1 + \lambda\mu + \frac{1}{\lambda}\ln\alpha$. □

We remark that Theorem 1 slightly generalizes some known results. For Minimum Cardinality Submodular Cover, it is easy to obtain from Theorem 1 the approximation ratio 1 + $\ln\alpha$ which is almost the same as $H(\alpha)$. Theorem 1 can also directly derive an approximation ratio 1 + $\frac{1}{\lambda}\ln\alpha$ for Minimum Cardinality Ratio-submodular Cover if $(\lambda, 0)$-submodularity is submodularity ratio of $f(\cdot)$, and another approximation ratio 1 + μ + $\ln\alpha$ for Minimum Cardinality Gap-submodular Cover if $(1, \mu)$-submodularity is submodularity gap of $f(\cdot)$.

4 Application

In the section, we would like to generalize our results to Minimum Resolving Restrained Dominating Set and Minimum Dominating Set in Multiplex Networks.

4.1 Minimum Resolving Restrained Dominating Set

Given a connected graph $G = \langle V, E\rangle$ with node set $V = \{v_1, v_2, ..., v_n\}$ and edge set E. The order of G is given by $n = |V|$ and its size by $m = |E|$. For any $u, v \in V$, we denote by $N(v)$ the set of all neighbors of v, by $dis(u, v)$ the shortest distance of u, v in G. Next, we introduce some concepts

including dominating set, resolving set and Minimum Resolving Restrained Dominating Set.

For any $D \subseteq V$ and any node $u \in V$, u is dominated by D if there exists a node $v \in D$ such that $v \in N(u)$. Any subset $D \subseteq V$ is called a dominating set if and only if any node in $V - D$ can be dominated by at least one node in D. In order to formally describe the dominating set, we define a potential function $f_1 : 2^V \to \mathbb{Z}^+$ for any $D \subseteq V$, where

$$f_1(D) = |\{v|v \in V - D, N(v) \cap D \neq \emptyset\}| + |D|. \quad (1)$$

Obviously, $f_1(\emptyset) = 0$, $f_1(D) < n$ if and only if the existence of $v \in V - D$ such that $N(v) \cap D = \emptyset$, namely, any subset $D \subseteq V$ is called a dominating set if and only if $f_1(D) = f_1(V) = n$.

For any $D \subseteq V$ and any two distinct nodes $u, w \in V$, u and w are resolved by D if there exists a node $v \in D$ such that $dis(v, u) \neq dis(v, w)$. Any subset $D \subseteq V$ is called a resolving set [8] if and only if any two distinct nodes in V can be resolved by D. In order to formally describe the resolving set, we construct a set $V' = \{(v_i, v_j)|v_i \in V, v_j \in V, i < j\}$ in polynomial time, let $C((v_i, v_j)) = \{v|v \in V, dis(v, v_i) \neq dis(v, v_j)\}$ for any $(v_i, v_j) \in V'$, and define a potential function $f_2 : 2^V \to \mathbb{Z}^+$ for any $D \subseteq V$, where

$$f_2(D) = |\{(v_i, v_j)|(v_i, v_j) \in V', C((v_i, v_j)) \cap D \neq \emptyset\}|. \quad (2)$$

Obviously, $f_2(\emptyset) = 0$, $f_2(D) < \frac{n(n+1)}{2}$ if and only if the existence of $(v_i, v_j) \in V'$ such that $C((v_i, v_j)) \cap D = \emptyset$, namely, any subset $D \subseteq V$ is called a resolving set if and only if $f_2(D) = f_2(V) = \frac{n(n+1)}{2}$. It's worth recalling that the resolving set with smallest cardinality is called *metric dimension* [9] in G. Finding the *metric dimension* in a general graph is NP-hard [10], which means it is unlikely to be solved precisely in polynomial time unless $P = NP$. At present, there are two greedy approximation algorithms [8, 11] for finding *metric dimension*, in which the greedy rule of Khuller-Greedy [8] with $(1 + 2\ln n)$-approximation is equivalent to selecting $v \in V - D$ with $\max_{v \in V-D} \Delta_v f_2(D)$ in each iteration.

The Minimum Resolving Restrained Dominating Set [12] is asked to find a smallest subset $D \subseteq V$ with smallest cardinality such that D is a dominating set and resolving set. More formally,

$$\min_{D \subseteq V} \{|D| : f_1(D) = f_1(V), f_2(D) = f_2(V)\}.$$

To the best of our knowledge, there is no approximation algorithm to solve Minimum Resolving Restrained Dominating Set so far. Our aim is to propose an approximation algorithm by considering a potential function $f : 2^V \to \mathbb{Z}^+$ for any $D \subseteq V$, where

$$f(D) = f_1(D) + f_2(D). \quad (3)$$

Let us show some properties of $f(D)$.

Lemma 2. *The potential function $f(\cdot)$ satisfies*
(i) $f(\cdot)$ is normalized, non-decreasing and $(1, 0)$-submodular,
(ii) $f(D) = f(V)$ if and only if D is a solution for Minimum Resolving Restrained Dominating Set, and
(iii) $f(D) < f(V)$ if and only if there exists a node $v \in V - D$ such that $\Delta_v f(D) > 0$.

Proof. (i) From (1) and (2), $f(\emptyset) = 0$ is clear. It is easy to see that $f(\cdot)$ is non-decreasing due to $|N(v) \cap D|$ and $|C((v_i, v_j)) \cap D|$ are non-decreasing with respect to D for any $v \in V - D$ and any $(v_i, v_j) \in V'$. For any $A \subseteq B \subseteq V$ and any $x \in V - B$, we have

$$\begin{aligned}\Delta_x f_1(A) &= f_1(A \cup \{x\}) - f_1(A) \\ &= |N[x] - A - \{v|v \in V, N(v) \cap A \neq \emptyset\}| \\ &\geq |N[x] - B - \{v|v \in V, N(v) \cap B \neq \emptyset\}| \\ &= \Delta_x f_1(B)\end{aligned}$$

and

$$\begin{aligned}&\Delta_x f_2(A) \\ &= f_2(A \cup \{x\}) - f_2(A) \\ &= |C(x) - \{(v_i, v_j)|(v_i, v_j) \in V', C((v_i, v_j)) \cap A = \emptyset\}| \\ &\geq |C(x) - \{(v_i, v_j)|(v_i, v_j) \in V', C((v_i, v_j)) \cap B = \emptyset\}| \\ &= \Delta_x f_2(B),\end{aligned}$$

hence,

$$\begin{aligned}\Delta_x f(A) &= \Delta_x f_1(A) + \Delta_x f_2(A) \\ &\geq \Delta_x f_1(B) + \Delta_x f_2(B) \\ &= \Delta_x f(B).\end{aligned}$$

(ii) $f(D) = f(V)$, namely, $f_1(D) = f_1(V) = n$ and $f_2(D) = f_2(V) = \frac{n(n+1)}{2}$ if and only if $N(v) \cap D \neq \emptyset$ for every $v \in V - D$ and $C((v_i, v_j)) \cap D \neq \emptyset$ for every $(v_i, v_j) \in V'$ if and only if D is a solution for Minimum Resolving Restrained Dominating Set.

(iii) $f(D) < f(V)$, namely, $f_1(D) < n$ or $f_2(D) < \frac{n(n+1)}{2}$ if and only if the existence of $v \in V - D$ such that $N(v) \cap D = \emptyset$ or $(v_i, v_j) \in V'$ such that $C((v_i, v_j)) \cap D = \emptyset$, which is equivalent to the existence of $v \in V - D$ such that $\Delta_v f(D) > 0$. □

Based on Lemma 2 and Theorem 1, it is easy to know algorithm CGA yields a $(1 + 2\ln(n + 1))$-approximation solution for Minimum Resolving Restrained Dominating Set due to

$$\begin{aligned}\max_{v \in V} f(\{v\}) &\leq \max_{v \in V}(f_1(\{v\})) + \max_{v \in V} f_2(\{v\}) \\ &= \max_{v \in V}(|N[v]|) + \max_{v \in V}(|C(v)|) \\ &< n + n^2 < (n + 1)^2.\end{aligned}$$

4.2 Minimum Dominating Set in Multiplex Networks

Considering a multiplex network which is formed by k different layers, i.e., $G = (G_1, ..., G_k)$, where every layer $G_i = (V, E_i)$ has a same set V of n nodes and a distinct set of edges E_i. We denote by $N_i(v)$ the set of all neighbors of v in $G_i = (V, E_i)$. In order to formally describe the dominating set

in $G_i = (V, E_i)$, we define a potential function $f_i : 2^V \to \mathbb{Z}^+$ for any $D \subseteq V$ and any $i \in \{1, 2, ..., k\}$, where

$$f_i(D) = |\{v|v \in V - D, N_i(v) \cap D \neq \emptyset\}| + |D|. \quad (4)$$

Obviously, $f_i(\emptyset) = 0$, $f_i(D) < n$ if and only if the existence of $v \in V - D$ such that $N_i(v) \cap D = \emptyset$, namely, any subset $D \subseteq V$ is called a dominating set in $G_i = (V, E_i)$ if and only if $f_i(D) = f_i(V) = n$. The Minimum Dominating Set in Multiplex Networks [13] is asked to find a smallest subset $D \subseteq V$ with smallest cardinality such that D is a dominating set in every layer $G_i = (V, E_i)$. More formally,

$$\min_{D \subseteq V} \{|D| : f_i(D) = f_i(V), i \in \{1, 2, ..., k\}\}.$$

Even with $k = 1$, Minimum Dominating Set in Multiplex Networks generalizes the well-known Minimum Dominating Set. Minimum Dominating Set in Multiplex Networks is NP-hard problem due to it can also be easily reduced to Minimum Dominating Set. Some practical applications such as Monitoring Epidemic in Multiplex Network [14], Early Detecting and Controlling Epidemic in Multiplex Network [15] and Constructing Extractive Text Summarization [16] are usually modeled as Minimum Dominating Set in Multiplex Networks.

Some heuristic algorithms [13, 17] for the Minimum Dominating Set in Multiplex Networks have been proposed. Our aim is to propose an approximation algorithm by considering a potential function $f : 2^V \to \mathbb{Z}^+$ for any $D \subseteq V$, where

$$f(D) = \sum_{i=1}^{k} f_i(D). \quad (5)$$

Let us show some properties of $f(D)$.

Lemma 3. *The potential function $f(\cdot)$ satisfies*
(i) $f(\cdot)$ is normalized, non-decreasing and $(1, 0)$-submodular,
(ii) $f(D) = f(V)$ if and only if D is a solution for Minimum Dominating Set in Multiplex Networks, and
(iii) $f(D) < f(V)$ if and only if there exists a node $v \in V - D$ such that $\Delta_v f(D) > 0$.

Proof. (i) From (4), $f(\emptyset) = 0$ is clear. It is easy to see that $f(\cdot)$ is non-decreasing due to $|N_i(v) \cap D|$ is non-decreasing with respect to D for any $v \in V - D$ and any $i \in \{1, 2, ..., k\}$. For any $A \subseteq B \subseteq V$ and any $x \in V - B$, we have

$$\begin{aligned}\Delta_x f_i(A) &= f_i(A \cup \{x\}) - f_i(A) \\ &= |N[x] - A - \{v|v \in V, N(v) \cap A \neq \emptyset\}| \\ &\geq |N[x] - B - \{v|v \in V, N(v) \cap B \neq \emptyset\}| \\ &= \Delta_x f_i(B),\end{aligned}$$

hence,

$$\begin{aligned}\Delta_x f(A) &= \sum_{i=1}^{k} \Delta_x f_i(A) \\ &\geq \sum_{i=1}^{k} \Delta_x f_i(B) \\ &= \Delta_x f(B).\end{aligned}$$

(ii) $f(D) = f(V)$, namely, $f_i(D) = f_i(V) = n$ for any $i \in \{1, 2, ..., k\}$ if and only if $N_i(v) \cap D \neq \emptyset$ for every $v \in V - D$ and any $i \in \{1, 2, ..., k\}$ if and only if D is a solution for Minimum Dominating Set in Multiplex Networks.

(iii) $f(D) < f(V)$, namely, the existence of $i \in \{1, 2, ..., k\}$ such that $f_i(D) < n$ if and only if the existence of $v \in V - D$ and $i \in \{1, 2, ..., k\}$ such that $N_i(v) \cap D = \emptyset$, which is equivalent to the existence of $v \in V - D$ such that $\Delta_v f(D) > 0$. □

Based on Lemma 3 and Theorem 1, it is easy to know algorithm CGA yields a $(1+\ln(k \times \delta + k))$-approximation solution for Minimum Dominating Set in Multiplex Networks due to

$$\begin{aligned}\max_{v \in V} f(\{v\}) &\leq \sum_{i=1}^{k} \max_{v \in V} f_i(\{v\}) \\ &= \sum_{i=1}^{k} \max_{v \in V}(|N_i[v]|) \\ &= \sum_{i=1}^{k} (\delta_i + 1) \\ &< k \times \delta + k,\end{aligned}$$

where δ_i is the maximum node degree of the graph $G_i = (V, E_i)$ and $\delta = \max\{\delta_1, \delta_2, ..., \delta_k\}$.

5 Conclusion

In this paper, we have proposed a variant of Minimum Cardinality Cover named Minimum Cardinality Multiple Quasi-submodular Cover and given a greedy approximation algorithm for it by defining a quasi-submodular potential function. As some applications of our results, we directly give $(1 + 2\ln(n + 1))$-approximation solution for Minimum Resolving Restrained Dominating Set and a $(1 + \ln(k \times \delta + k))$-approximation solution for Minimum Dominating Set in Multiplex Networks, where n is node number and δ is the maximum node degree of the input graph.

In the future work, we would like to generalize our method to some other related NP-hard problems to explore their greedy approximation algorithms with better approximation ratio or performance.

6 Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant U1811263.

References

[1] L.A. Wolsey, An analysis of the greedy algorithm for the submodular set covering problem, Combinatorica, 2(4)(1982)385-393.

[2] P. Wan, D.-Z. Du, P. M. Pardalos, W.Wu, Greedy approximations for minimum submodular cover with submodular cost, Computational Optimization and Applications, 45(2)(2010)463-474.

[3] W.-D. Chen, H. Zhong, L.-D. Wu, D.-Z. Du, A general greedy approximation algorithm for finding minimum positive influence dominating sets in social networks, Journal of Combinatorial Optimization, (2021)1-20.

[4] S.-N. Gong, Q.-Q. Nong, T. Sun, Q.-Z. Fang, D.-Z. Du, X.-Y. Shao, Maximize a monotone function with a generic submodularity ratio, Theoretical Computer Science, 853(2021)16-24.

[5] M.-J. Shi, Z.-S. Yang, W. Wang, Minimum non-submodular cover problem with applications, Applied Mathematics and Computation, 410(4)(2021)126442.

[6] A. Das, D. Kempe, Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection, Journal of Machine Learning Research, 19(1)(2018)74–107.

[7] C. Chekuri, T. Inamdar, K. Quanrud, K. Varadarajan, Z. Zhang, Algorithms for covering multiple submodular constraints and applications, Journal of Combinatorial Optimization, 44(2)(2022)979-1010.

[8] S. Khuller, B. Raghavachari, A. Rosenfeld, Landmarks in graphs, Discrete applied mathematics, 70(3)(1996)217-229.

[9] Y.-F. Huang, B. Hou, W. Liu, Y.-F. Huang, L.-D. Wu, R. Stephen, S.-G. Gao, On approximation algorithm for the edge metric dimension problem. Theoretical Computer Science, 853(2021)2-6.

[10] Z. Beerliova, F. Eberhard, T. Erlebach, H. Alexander, L. Shankar Ram. Network discovery and verification, IEEE Journal on selected areas in communications, 24(12)(2006)2168-2181.

[11] M. Hauptmann, R. Schmied, C. Viehmann, Approximation complexity of metric dimension problem, Journal of Discrete Algorithms, 14(2012)214-222.

[12] G.B. Monsanto, H.M. Rara, Resolving restrained domination in graphs. European Journal of Pure and Applied Mathematics, 14(3)(2021)829-841.

[13] D.-W. Zhao, G.-X. Xiao, Z. Wang, L.-H. Wang, L.-J. Xu, Minimum dominating set of multiplex networks: definition, application, and identification, IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(12)(2020)7823-7837.

[14] Z.-F. Li, F.-H. Yan, Y.-C. Jiang. Cross-layers cascade in multiplex networks, Autonomous Agents and Multi-Agent Systems, 29(6)(2015)1186-1215.

[15] W. Wang, Q.-H. Liu, S.-M. Cai, L.A. Braunstein, H.E. Stanley, Suppressing disease spreading by using information diffusion on multiplex networks, Scientific reports, 6(1)(2016)1-14.

[16] C. Shen, T. Li, Multi-Document Summarization via the Minimum Dominating Set, In COLING 2010: Proceedings of the 23rd International Conference on Computational Linguistics, Tsinghua University Press, 2010, pp. 984–992.

[17] M.M.D. Khomami, A. Rezvanian, A.M. Saghiri, M.R. Meybodi, Solving Minimum Dominating Set in Multiplex Networks Using Learning Automata, In CSICC 2021: Proceedings of the 26th International Computer Conference, Computer Society of Iran, 2021, pp. 1-6.

Empirical Study on Performance–Perception Discrepancy in RGB–Thermal Monocular Depth Estimation under Varying Illumination

Kang Min Ji
Dongguk University, 30 Pildong-ro 1-gil, Jung-gu, Seoul 04620, South Korea
*Corresponding author: kangminji411@gmail.com

Abstract

With the advancement of multimodal perception technologies, integrating visible (RGB) and thermal infrared (THR) information has become a key approach to enhancing the robustness of visual systems under complex illumination conditions. While existing studies primarily focus on improving quantitative accuracy through multimodal fusion, less attention has been paid to the perceptual differences and consistency between modalities. This study investigates the performance–perception discrepancy in multimodal depth estimation under varying illumination scenarios. Through comparative experiments between RGB and THR modalities, the analysis reveals that THR exhibits superior numerical performance (e.g., lower RMSE and AbsRel) in low-light and nighttime conditions, yet suffers from perceptual degradation such as over-smoothing and structural blurring. Moreover, by referencing findings in multimodal object detection, this phenomenon is shown to be task-general, arising from the distinct spatial frequency responses of different modalities. The presented results provide empirical evidence and theoretical insight for future research on multimodal feature fusion and perceptual consistency optimization.

Index Terms— Depth Estimation, Multimodal, Illumination Robustness, Quantitative Evaluation, Visual Consistency.

1 Introduction

Recent advances in computer vision have enabled machines to perceive and reconstruct 3D structures from visual data with remarkable accuracy, particularly through deep learning–based monocular depth estimation (MDE) methods [1, 25]. These models have achieved significant progress in autonomous driving, robotics, and scene understanding [3, 17]. However, RGB-based depth estimation systems remain sensitive to environmental variations—especially in challenging illumination conditions such as nighttime or adverse weather—where visible light becomes unreliable.

Thermal infrared (THR) imaging provides a promising complementary modality in such environments. By capturing infrared radiation emitted by objects above absolute zero, thermal cameras can perceive structures that are invisible to the RGB spectrum, offering illumination invariance and robustness to occlusion and haze [11, 5]. As illustrated in prior multimodal vision studies, THR sensors have been successfully used for tasks such as salient object detection, pedestrian recognition, and image segmentation, thanks to their strong response to heat-emitting objects even in complete darkness [11, 5]. Nevertheless, when applied to depth estimation, thermal images introduce new challenges: they often exhibit low resolution, limited texture, and reduced semantic richness, making fine-grained 3D reconstruction difficult.

To address these challenges, multimodal fusion between RGB and THR modalities has emerged as a viable strategy. Recent works have explored RGB–Thermal integration through two primary paradigms: (1) feature-level fusion networks that combine spatial and channel-wise cues from both modalities to enhance representation learning [20, 27], and (2) cross-modal distillation frameworks that transfer geometric knowledge from large-scale RGB foundation models to thermal networks using confidence-aware consistency objectives [15, 24, 2]. Despite these advances, the performance–perception inconsistency remains a largely overlooked issue in multimodal depth estimation: quantitative metrics such as RMSE or AbsRel may improve significantly through thermal guidance, while the resulting depth maps exhibit visual artifacts such as texture loss or oversmoothing.

This discrepancy highlights a fundamental property of multimodal depth perception—the asymmetric contribution of RGB and THR features. RGB imagery captures high-frequency textures and detailed semantics but degrades rapidly in dark scenes, whereas THR imagery maintains structural continuity at the cost of visual sharpness. Existing multimodal fusion models [14, 4] often overlook this imbalance by treating both modalities uniformly, resulting in fused representations that may optimize numerical performance but fail to achieve perceptual coherence.

In this study, we systematically analyze the performance–perception discrepancy in multimodal depth estimation under varying illumination conditions. Using a series of controlled experiments comparing RGB, THR, and fused modalities, we observe that thermal-based estimation yields superior quantitative metrics but perceptually degraded visual results. We further discuss the relevance of this phenomenon to other multimodal tasks, such as object detection and salient

object segmentation, where similar modality-dependent tradeoffs have been observed [19, 26]. The findings of this study provide empirical evidence and analytical insights for developing future multimodal perception systems that balance accuracy with perceptual fidelity.

The main contributions of this paper are as follows:

- We conduct an empirical analysis of RGB–thermal (RGB–THR) depth estimation under varying illumination, revealing a clear performance–perception discrepancy.
- We identify that this discrepancy arises from modality-dependent feature characteristics, where THR improves quantitative accuracy but weakens visual fidelity.
- The findings provide experimental evidence and insight for subsequent research on multimodal feature fusion and perceptual consistency optimization.

2 Related Works

2.1 RGB and Thermal Imaging in Visual Tasks

Multimodal perception has emerged as a key paradigm in computer vision to improve robustness against illumination changes, occlusions, and environmental degradation. Among various modality combinations, visible-light (RGB) and thermal infrared (THR) imaging form a particularly complementary pair. RGB sensors capture reflected visible light, providing rich texture and color cues essential for semantic understanding and fine spatial delineation. In contrast, THR cameras sense long-wave infrared radiation emitted by objects, enabling reliable perception under adverse or low-illumination conditions [23].

The integration of RGB and THR data has been explored across numerous vision tasks, including pedestrian detection, salient object detection, semantic segmentation, and scene understanding [7, 8, 13]. For example, multispectral detectors trained on datasets such as KAIST and LLVIP have shown that thermal cues can significantly enhance nighttime pedestrian recognition [8, 7]. In salient object detection, cross-modality interaction modules and attention-guided fusion methods [13, 28] leverage complementary modality information to achieve robust target localization under dynamic lighting. Similarly, RGB–THR fusion in semantic segmentation improves feature stability at the object boundary level [21], confirming the benefit of multimodal integration in challenging environments.

The advantages of RGB–THR fusion extend beyond conventional image analysis. Recent studies have applied multispectral fusion to domains such as autonomous driving [22], UAV-based surveillance [16], and robotics [9], where the goal is to achieve all-day, all-weather perception. These applications emphasize that while RGB features provide geometric and semantic richness, THR inputs ensure visibility and structural consistency across varying conditions—highlighting the importance of effective cross-modal fusion for real-world deployment.

2.2 Multimodal Fusion Strategies and Emerging Challenges

With the success of deep learning, multimodal fusion has evolved from handcrafted feature concatenation to learned feature-level and attention-based strategies. Early fusion approaches such as pyramid-based blending or weighted averaging [18, 12] mainly focused on pixel-level enhancement without learning task-specific representations. Modern deep fusion frameworks employ dual-stream encoder–decoder architectures, where modality-specific features are extracted independently and later integrated via cross-attention or adaptive weighting [13, 10]. These designs enable networks to selectively exploit complementary signals, improving the robustness and adaptability of multimodal perception systems.

More recently, researchers have introduced transformer-based and frequency-aware models to enhance cross-modal representation learning. Transformer architectures offer global context modeling between RGB and THR modalities [16], while frequency-domain analyses reveal that different modalities contribute unevenly across spatial frequency bands—RGB features dominate high-frequency detail, whereas THR features emphasize low-frequency structure [6]. Such insights have motivated new fusion pipelines that adaptively balance modalities according to scene characteristics.

Despite remarkable progress in multimodal fusion, most existing studies still focus primarily on quantitative performance indicators such as accuracy or mIoU, while the perceptual quality of fused results has received far less attention. In practice, numerical improvements do not necessarily imply perceptually consistent or visually coherent outputs. Our experiments clearly reveal that the inclusion of thermal information can stabilize numerical accuracy yet sometimes lead to degraded visual realism. This observation indicates that numerical metrics alone cannot comprehensively represent the overall quality of multimodal perception. Therefore, our study emphasizes the need to re-examine RGB–THR fusion from a dual perspective—quantitative performance and perceptual fidelity—to achieve a more balanced and interpretable evaluation of multimodal systems.

3 Comparative Evaluation

3.1 Experimental Setup

Dataset.

All experiments in this study are conducted on the **Multi-Spectral Stereo (MS^2) Dataset** [14], a large-scale outdoor benchmark designed for multisensor perception and depth estimation research, as illustrated in Fig. 1.

The dataset provides synchronized recordings from stereo RGB, stereo near-infrared (NIR), and stereo thermal (THR) cameras, together with stereo LiDAR scanners and GPS/IMU navigation units. This comprehensive sensor suite enables precise geometric calibration and temporal synchronization across modalities, supporting detailed investigation of multimodal visual perception under real-world conditions.

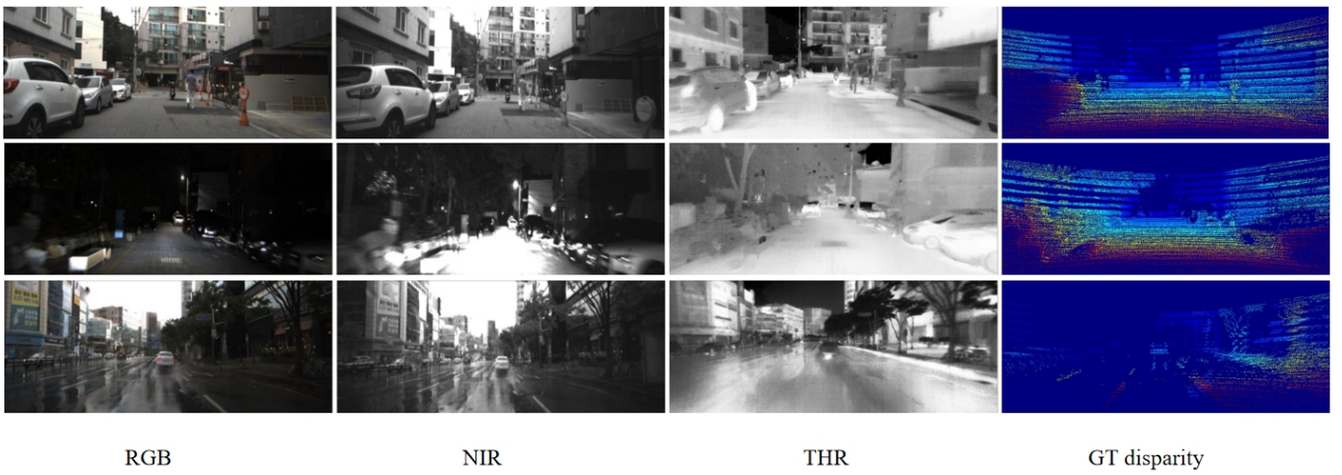

Figure 1: Overview of the Multi-Spectral Stereo (MS^2) dataset. The dataset provides synchronized RGB, NIR, and thermal stereo images captured under diverse environmental conditions (day, night, and rain), along with LiDAR, GPS, and IMU data for geometric consistency.

The MS^2 dataset comprises approximately 184 K rectified and synchronized stereo image pairs captured across diverse environments, including urban streets, residential areas, campus roads, and suburban regions. Each location was recorded multiple times under varying illumination and weather conditions, covering clear, cloudy, and rainy days, as well as morning, daytime, and nighttime scenes. Such diversity provides a valuable basis for studying modality-specific behaviors under challenging visual scenarios. In addition to multi-spectral imagery, the dataset also includes projected LiDAR depth maps, odometry information in both camera and LiDAR coordinate systems, and GPS/IMU trajectories to ensure metric-scale consistency.

In this work, we use theleft RGB andleft THR images, together with their corresponding LiDAR-projected depth maps, to perform a controlled comparison between visible-spectrum and long-wave infrared sensing for monocular depth estimation. Both modalities are spatially aligned and temporally synchronized, ensuring consistent supervision during training and evaluation. The input resolution is fixed at 640×256 pixels, and we follow the official preprocessing protocol of the dataset to maintain alignment and radiometric consistency across all samples.

The MS^2 dataset revisits the same physical locations under different illumination and weather conditions, providing a dense set of multi-condition correspondences for reliable cross-modal analysis. This feature enables systematic evaluation of modality-dependent robustness across structured and unstructured environments, as well as across varying visibility levels such as day, night, and rain. Its synchronized multi-sensor design and environmental diversity make MS^2 a suitable benchmark for analyzing how RGB and thermal modalities contribute to stable and reliable depth perception in complex outdoor scenes.

Implementation details. For network implementation, we employ the ConvNeXt-Tiny backbone as the feature extractor owing to its balance between computational efficiency and representational capacity. For the RGB modality, the model is initialized with ImageNet-pretrained weights to leverage general visual priors and accelerate convergence. Since thermal (THR) images are single-channel inputs lacking color information, two configurations are explored to ensure fair evaluation. In the first configuration, the THR image is replicated across three channels to match the input dimension of the pretrained ConvNeXt-Tiny model, thereby enabling weight transfer from the RGB domain. In the second configuration, the model is trained with a single-channel input using the same backbone structure but without pretrained initialization, allowing the network to learn modality-specific representations from scratch.

To maintain a fair comparison between modalities, we do not employ any data augmentation strategies such as random flipping, color jittering, or cropping. Both models are trained under identical hyperparameter settings, including optimizer configuration, learning rate schedule, and batch size. The implementation is based on PyTorch and executed on an NVIDIA RTX 4090 GPU with 24 GB of memory.

Training configuration. All models are trained for **25 epochs** using the **Adam optimizer** with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate of 1×10^{-4}. A cosine decay schedule is adopted to gradually reduce the learning rate over time, ensuring stable convergence. The batch size is set to **8**, and all experiments are conducted on an NVIDIA RTX 4090 GPU with 24 GB of memory. We follow the official data preprocessing and normalization procedure of the MS^2 dataset to maintain consistency across modalities. No additional data augmentation or modality-specific tuning is applied during training to ensure a fair comparison between RGB and thermal inputs.

The overall loss function combines a **scale-invariant depth loss** and an **edge-aware smoothness term**, which are widely used in monocular depth estimation [4]. The final objective is

defined as:

$$L = L_{\text{si}} + \lambda L_{\text{sm}}, \quad \lambda = 0.1, \tag{1}$$

where the scale-invariant term L_{si} measures relative depth consistency in logarithmic space:

$$L_{\text{si}} = \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}\left(\sum_i d_i\right)^2, \tag{2}$$

with $d_i = \log D_i - \log D_i^*$ representing the difference between predicted and ground-truth depth in log scale. The smoothness regularizer encourages spatial coherence while preserving image edges:

$$L_{\text{sm}} = \sum_{i,j}\left(|\partial_x D_{i,j}|e^{-|\partial_x I_{i,j}|} + |\partial_y D_{i,j}|e^{-|\partial_y I_{i,j}|}\right), \tag{3}$$

where D denotes the predicted depth map and I the corresponding input image. This combination ensures both global depth consistency and local structural smoothness in the predicted maps.

3.2 Evaluation Metrics

To quantitatively evaluate the performance of depth estimation, we adopt three widely used metrics—**Absolute Relative Error (AbsRel)**, **Root Mean Square Error (RMSE)**, and **Threshold Accuracy (δ_i)**—which were originally introduced by Eigen *et al.* [4].

These metrics jointly capture both the numerical deviation from the ground-truth depth and the relative structural consistency across scenes. All evaluations are performed on the official test split of the MS^2 dataset using the unfiltered LiDAR depth maps as reference.

Absolute Relative Error (AbsRel). This metric measures the mean relative deviation between the predicted depth D_i and the ground-truth depth D_i^*:

$$\text{AbsRel} = \frac{1}{n}\sum_{i=1}^{n}\frac{|D_i - D_i^*|}{D_i^*}. \tag{4}$$

A lower AbsRel value indicates a smaller proportional error, implying that the predicted depth magnitudes are closer to their true values. Because it normalizes the difference by D_i^*, AbsRel is particularly sensitive to near-range regions where depth changes rapidly, making it an effective indicator of local depth fidelity.

Root Mean Square Error (RMSE). RMSE evaluates the overall Euclidean distance between prediction and ground truth, reflecting global consistency across the entire image:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(D_i - D_i^*)^2}. \tag{5}$$

Unlike AbsRel, RMSE penalizes large absolute deviations more heavily, and is therefore dominated by outlier pixels or distant regions. A smaller RMSE value corresponds to a globally smoother and numerically stable depth prediction.

Threshold Accuracy (δ_i). To assess relative correctness independent of absolute scale, we follow the standard accuracy criterion proposed in [4]. For each pixel, the ratio between prediction and ground truth is computed, and the percentage of pixels satisfying the threshold condition is reported as

$$\text{Accuracy}(\delta_i) = \frac{1}{n}\sum_{i=1}^{n}\mathbb{I}\left(\max\left(\frac{D_i}{D_i^*}, \frac{D_i^*}{D_i}\right) < \delta_i\right) \tag{6}$$

where $\mathbb{I}(\cdot)$ denotes an indicator function that equals 1 when the condition is satisfied and 0 otherwise. The threshold values are set to $\delta_i \in \{1.25,\ 1.25^2,\ 1.25^3\}$, corresponding to increasing levels of tolerance. Higher δ_i values represent looser error bounds, while δ_1 measures strict accuracy and δ_3 captures broader alignment.

This metric effectively measures how many pixels fall within a fixed multiplicative error bound, providing a complementary view to absolute-error measures.

Together, these three metrics provide a comprehensive assessment of depth estimation quality. AbsRel emphasizes relative precision in nearby regions, RMSE captures global numerical stability, and the threshold-based accuracy highlights structural consistency under scale variations. By jointly analyzing these indicators, we can evaluate not only the quantitative reliability of each modality but also its robustness to illumination and texture variations present in the MS^2 dataset.

3.3 Experimental Results

3.3.1 Quantitative Comparison

Table 1 presents the quantitative evaluation results of monocular depth estimation using RGB and thermal (THR) modalities under three illumination conditions—daytime, nighttime, and rainy—on the MS^2 dataset. Both networks were trained under identical optimization and data processing settings to ensure fair comparison. Across all environments, the THR-based model consistently achieves lower error metrics and higher accuracy rates, indicating that the thermal modality provides more stable geometric cues and greater robustness to illumination changes.

Under the *daytime* condition, the two modalities exhibit comparable performance, with THR showing a modest improvement in most metrics. The AbsRel and RMSE values of THR (0.08 and 2.96, respectively) are slightly lower than those of RGB (0.09 and 3.45). This marginal gap is attributed to the rich texture and color gradients available in RGB images under sufficient illumination, which enable reliable depth inference through photometric cues.

In the *nighttime* condition, the superiority of the thermal modality becomes prominent. The AbsRel decreases from 0.121 to 0.081, and RMSE drops from 4.11 to 2.84, reflecting a substantial reduction in overall depth error. Furthermore, the δ_1 accuracy improves from 0.872 to 0.938, confirming that THR preserves more consistent structural correspondence when visual information is degraded by low-light noise and contrast loss. Notably, the performance of THR remains

Table 1: Quantitative comparison between RGB and THR modalities under different illumination conditions on the MS^2 dataset. Lower is better for AbsRel, SqRel, RMSE, and RMSE(log); higher is better for δ_i.

Condition	Input	AbsRel ↓	SqRel ↓	RMSE ↓	RMSE(log) ↓	δ_1 ↑	δ_2 ↑	δ_3 ↑
Day	RGB	0.090	0.427	3.454	0.117	0.911	0.979	**0.998**
	THR	**0.080**	**0.342**	**2.955**	**0.112**	**0.941**	**0.981**	0.995
Night	RGB	0.121	0.619	4.105	0.153	0.872	0.981	0.989
	THR	**0.081**	**0.335**	**2.844**	**0.112**	**0.938**	**0.980**	**0.991**
Rainy	RGB	0.139	0.897	4.841	0.182	0.841	0.913	0.981
	THR	**0.115**	**0.549**	**3.785**	**0.159**	**0.875**	**0.952**	**0.987**

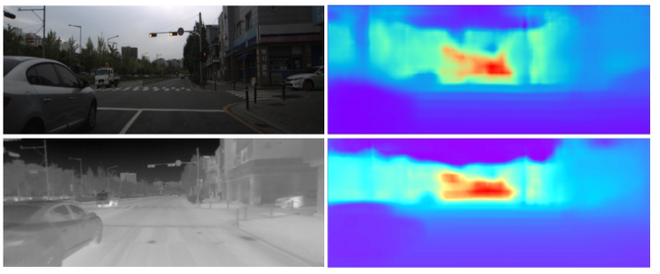

Figure 2: Qualitative comparison of RGB- and THR-based depth predictions under the *daytime* condition. The RGB modality exhibits clearer edges and richer local textures, while THR outputs appear smoother and more homogeneous.

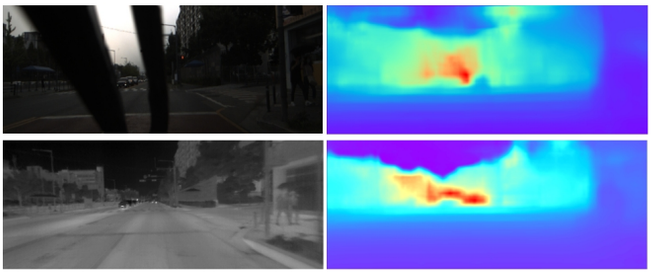

Figure 3: Qualitative comparison under the *rainy* condition. Despite visual occlusions from raindrops and wiper traces, THR maintains structural continuity, whereas RGB preserves distant object details with perceptual contrast.

highly stable between the daytime and nighttime settings, suggesting that thermal imaging is largely invariant to illumination intensity.

For the *rainy* condition, both modalities experience increased errors due to reflection, occlusion, and atmospheric scattering; however, THR still maintains a clear advantage. The AbsRel decreases from 0.139 to 0.115, and RMSE from 4.84 to 3.79, while δ_1 improves from 0.841 to 0.875. These results indicate that the thermal signal provides more coherent depth boundaries under adverse weather, mitigating the degradation effects commonly observed in RGB-based estimation.

In summary, the thermal modality exhibits strong resilience to environmental variations, yielding consistent performance across both well-lit and low-visibility scenarios. The relatively small performance gap between daytime and nighttime conditions further demonstrates the illumination-invariant characteristics of thermal sensing, highlighting its potential as a reliable alternative or complementary input for robust monocular depth estimation.

3.3.2 Qualitative Visualization

To provide a visual understanding of the modality-specific differences, we further present qualitative depth estimation results under three illumination conditions from the MS^2 dataset: daytime, rainy, and nighttime. Each visualization includes the input image and the corresponding predicted depth map for both RGB and THR modalities, highlighting the contrast between numerical stability and perceptual fidelity. Under the *daytime* condition (Fig. 2), the RGB-based prediction displays sharper edges and richer local details. Elements such as traffic lights and nearby vehicles are distinctly represented, demonstrating the benefit of texture and color gradients in depth reconstruction. In contrast, the THR-based map appears smoother and more uniform, suggesting higher numerical consistency but reduced perceptual richness.

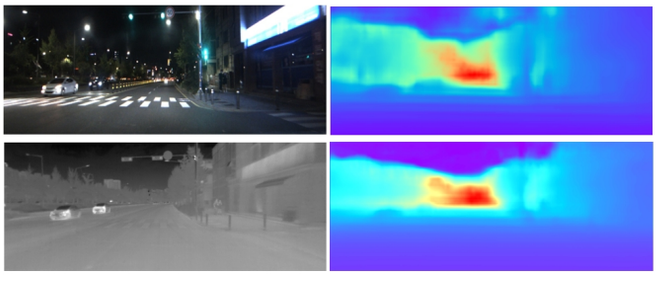

Figure 4: Qualitative comparison under the *nighttime* condition. RGB predictions capture meaningful visual cues such as vehicles and pedestrians, while THR produces smoother yet less visually expressive results.

In the *rainy* scenario (Fig. 3), the scene involves visual occlusion from raindrops and wiper traces. The THR modality maintains structural coherence since it is unaffected by these optical distortions, whereas the RGB prediction exhibits localized degradation. Nevertheless, distant objects such as vehicles remain perceivable in the RGB result, indicating that RGB retains a degree of depth sensitivity even under adverse visual conditions. Under the *nighttime* condition (Fig. 4), RGB predictions capture meaningful spatial cues such as pedestrians and vehicles with higher perceptual contrast, whereas THR

maintains smoother yet less expressive results. Despite the quantitative advantage of THR, the RGB-based outputs deliver more visually coherent depth perception.

In summary, while the THR modality achieves superior numerical accuracy across all conditions, the RGB modality produces perceptually more natural and structurally expressive depth maps. This observation highlights the existence of a performance–perception gap, emphasizing that quantitative superiority does not necessarily correlate with visual plausibility.

4 Discussion

The experimental findings presented in this study reveal a distinctive divergence between numerical performance and perceptual quality across RGB and thermal (THR) modalities in monocular depth estimation. Although the THR-based model consistently outperforms its RGB counterpart in quantitative indicators—achieving lower AbsRel and RMSE values under all illumination conditions—the qualitative analysis demonstrates that RGB predictions exhibit greater visual coherence, sharper boundaries, and richer structural expressiveness. This paradoxical outcome reflects the fundamental difference in how the two modalities encode visual information and how numerical optimization interacts with perceptual realism.

From a signal interpretation perspective, the thermal modality captures scene geometry primarily through radiometric emission differences, resulting in spatially smooth but low-frequency representations. Such inputs tend to minimize local gradient variance and yield stable predictions under low-visibility environments such as rain or night, explaining the superior metric values obtained by THR. However, this very stability comes at the cost of attenuated texture sensitivity, leading to over-smoothing in depth transitions and the loss of high-frequency cues that are critical for perceptual depth perception. In contrast, RGB images, rich in color and luminance gradients, provide abundant local features that enhance fine-grained spatial reconstruction. Consequently, although RGB models are more vulnerable to illumination noise, they preserve edge continuity and scene realism—factors that humans intuitively associate with visual quality.

These findings suggest that numerical accuracy and perceptual realism in depth estimation do not necessarily converge, especially across heterogeneous modalities. The observed performance–perception gap highlights a limitation of existing training objectives, which typically optimize for pixel-wise consistency while overlooking perceptual-level coherence. This misalignment underscores the need for evaluation frameworks that jointly assess numerical and perceptual aspects of depth quality, particularly in multimodal contexts where the data distributions are inherently unbalanced.

Future work should extend these insights toward *modality-aware fusion frameworks* that integrate the complementary strengths of RGB and THR sensing. Such methods could employ frequency-domain alignment or cross-modal attention to adaptively emphasize texture fidelity in RGB while leveraging the radiometric stability of THR under adverse conditions. Moreover, perceptually motivated loss functions and human-centered evaluation metrics may bridge the current gap between objective performance and subjective visual realism. Ultimately, achieving both quantitative robustness and perceptual fidelity will be a crucial step toward building reliable, interpretable, and human-aligned depth estimation systems for real-world applications.

5 Conclusion

This work presents an empirical study on the performance–perception relationship in monocular depth estimation using RGB and thermal (THR) modalities on the MS^2 dataset. Through systematic quantitative and qualitative analyses, we observe a consistent divergence between numerical accuracy and visual realism. Specifically, the THR modality achieves lower depth estimation errors and higher stability under adverse illumination conditions such as rain or night, demonstrating its robustness against environmental variations. However, the RGB modality consistently delivers more perceptually coherent depth maps, preserving edges, textures, and fine details that are visually aligned with human depth perception.

These findings underscore that numerical superiority does not necessarily imply perceptual fidelity, revealing an intrinsic imbalance in current objective functions and evaluation metrics. The study highlights the need to jointly consider perceptual and numerical dimensions when assessing and optimizing depth estimation models. In particular, future research should explore perceptually informed loss functions and multimodal fusion strategies that explicitly leverage the complementary strengths of RGB and THR data. By integrating radiometric stability from thermal sensing with the rich semantic and structural priors of RGB imagery, it may be possible to construct depth estimation frameworks that achieve both quantitative robustness and perceptual consistency across diverse environmental conditions.

References

[1] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4009–4018, 2021.

[2] Sri Aditya Deevi, Connor Lee, Lu Gan, Sushruth Nagesh, Gaurav Pandey, and Soon-Jo Chung. Rgb-x object detection via scene-specific fusion modules. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 7366–7375, 2024.

[3] Pengxiang Ding, Han Zhao, Wenjie Zhang, Wenxuan Song, Min Zhang, Siteng Huang, Ningxi Yang, and Donglin Wang. Quar-vla: Vision-language-action model for quadruped robots. In *European Conference on Computer Vision*, pages 352–367. Springer, 2024.

[4] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

[5] Qishen Ha, Kohei Watanabe, Takumi Karasawa, Yoshitaka Ushiku, and Tatsuya Harada. Mfnet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5108–5115. IEEE, 2017.

[6] Jingxue Huang, Xilai Li, Tianshu Tan, Xiaosong Li, and Tao Ye. Mma-unet: A multi-modal asymmetric unet architecture for infrared and visible image fusion. *arXiv preprint arXiv:2404.17747*, 2024.

[7] Soonmin Hwang, Jaesik Park, Namil Kim, Yukyung Choi, and In So Kweon. Multispectral pedestrian detection: Benchmark dataset and baseline. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1037–1045, 2015.

[8] Xinyu Jia, Chuang Zhu, Minzhen Li, Wenqi Tang, and Wenli Zhou. Llvip: A visible-infrared paired dataset for low-light vision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3496–3504, 2021.

[9] Shehryar Khattak, Christos Papachristos, and Kostas Alexis. Visual-thermal landmarks and inertial fusion for navigation in degraded visual environments. In *2019 IEEE Aerospace Conference*, pages 1–9. IEEE, 2019.

[10] Lina Liu, Xibin Song, Jiadai Sun, Xiaoyang Lyu, Lin Li, Yong Liu, and Liangjun Zhang. Mff-net: Towards efficient monocular depth completion with multi-modal feature fusion. *IEEE Robotics and Automation Letters*, 8(2):920–927, 2023.

[11] Yunpeng Ma, Dengdi Sun, Qianqian Meng, Zhuanlian Ding, and Chenglong Li. Learning multiscale deep features and svm regressors for adaptive rgb-t saliency detection. In *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*, volume 1, pages 389–392. IEEE, 2017.

[12] Gemma Piella. A general framework for multiresolution image fusion: from pixels to regions. *Information fusion*, 4(4):259–280, 2003.

[13] Jifeng Shen, Yifei Chen, Yue Liu, Xin Zuo, Heng Fan, and Wankou Yang. Icafusion: Iterative cross-attention guided feature fusion for multispectral object detection. *Pattern Recognition*, 145:109913, 2024.

[14] Ukcheol Shin, Jinsun Park, and In So Kweon. Deep depth estimation from thermal image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1043–1053, 2023.

[15] Shreyas S Shivakumar, Neil Rodrigues, Alex Zhou, Ian D Miller, Vijay Kumar, and Camillo J Taylor. Pst900: Rgb-thermal calibration, dataset and segmentation network. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 9441–9447. IEEE, 2020.

[16] Shivpal Singh, K Sathya Babu, Vasit Sagan, Chandrakanth Vipparla, K Palaniappan, and Hadi AliAkbarpour. Uav detect, track and follow (dtf) of non-stationary targets in aerial thermal videos. In *2025 17th International Conference on Computer and Automation Engineering (ICCAE)*, pages 165–170. IEEE, 2025.

[17] Wenxuan Song, Han Zhao, Pengxiang Ding, Can Cui, Shangke Lyu, Yaning Fan, and Donglin Wang. Germ: A generalist robotic model with mixture-of-experts for quadruped robot. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11879–11886. IEEE, 2024.

[18] Alexander Toet. Image fusion by a ratio of low-pass pyramid. *Pattern recognition letters*, 9(4):245–253, 1989.

[19] Zhengzheng Tu, Zhun Li, Chenglong Li, Yang Lang, and Jin Tang. Multi-interactive dual-decoder for rgb-thermal salient object detection. *IEEE Transactions on Image Processing*, 30:5678–5691, 2021.

[20] Zhengzheng Tu, Yan Ma, Zhun Li, Chenglong Li, Jieming Xu, and Yongtao Liu. Rgbt salient object detection: A large-scale dataset and benchmark. *IEEE Transactions on Multimedia*, 25:4163–4176, 2022.

[21] Yike Wang, Gongyang Li, and Zhi Liu. Sgfnet: Semantic-guided fusion network for rgb-thermal semantic segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7737–7748, 2023.

[22] Yingjie Wang, Qiuyu Mao, Hanqi Zhu, Jiajun Deng, Yu Zhang, Jianmin Ji, Houqiang Li, and Yanyong Zhang. Multi-modal 3d object detection in autonomous driving: a survey. *International Journal of Computer Vision*, 131(8):2122–2152, 2023.

[23] AN Wilson, Khushi Anil Gupta, Balu Harshavardan Koduru, Abhinav Kumar, Ajit Jha, and Linga Reddy Cenkeramaddi. Recent advances in thermal imaging and its applications using machine learning: A review. *IEEE Sensors Journal*, 23(4):3395–3407, 2023.

[24] Han Xu, Jiayi Ma, Junjun Jiang, Xiaojie Guo, and Haibin Ling. U2fusion: A unified unsupervised image fusion network. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):502–518, 2020.

[25] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. New crfs: Neural window fully-connected crfs for monocular depth estimation. *arXiv preprint arXiv:2203.01502*, 2022.

[26] Qiang Zhang, Nianchang Huang, Lin Yao, Dingwen Zhang, Caifeng Shan, and Jungong Han. Rgb-t salient object detection via fusing multi-level cnn features. *IEEE Transactions on Image Processing*, 29:3321–3335, 2019.

[27] Qiang Zhang, Tonglin Xiao, Nianchang Huang, Dingwen Zhang, and Jungong Han. Revisiting feature fusion for rgb-t salient object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(5):1804–1818, 2020.

[28] Wujie Zhou, Qinling Guo, Jingsheng Lei, Lu Yu, and Jenq-Neng Hwang. Ecffnet: Effective and consistent feature fusion network for rgb-t salient object detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1224–1235, 2021.

Research on the path of AI to help reduce the burden and increase efficiency of primary and secondary education

Xianfei Ke

Abstract—Artificial intelligence (AI) technology is widely used in four core areas: intelligent teaching system (classroom behavior analysis and teaching strategy adjustment), personalized learning platform (personalized learning route planning and accurate resource delivery), intelligent evaluation tool (automatic scoring ability and real-time tracking of learning situation), and VR/AR virtual on-site education, which significantly improves the efficiency of primary and secondary school education, promotes differentiated learning practices, and effectively improves student participation. The main problems in the current promotion process include insufficient equipment resources, insufficient digital literacy of teachers, data scattering, "island" situation, and biased algorithms. Many difficulties need to be solved, and a comprehensive response plan should be formulated: improve top-level planning, coordinate the allocation of regional funds and the use of special funds; Eliminate data barriers, help create an open sharing and exchange platform, and promote industry-university-research collaboration to carry out research and improvement; Initiate the implementation of teacher training plans by level and category, focusing on supporting backward schools; Formulate a data security supervision system and establish an algorithm fairness evaluation system. The next development trend will focus on emotional communication and support, cross-disciplinary innovation, and high-quality resource sharing and sinking, so as to better achieve the strategic goals of educational equity and quality improvement.

Index Terms—Artificial Intelligence and Education; Reduce Burden and Increase Efficiency; Intelligent Teaching System.

I. INTRODUCTION

In recent years, the application research of artificial intelligence (AI) in the field of education has shown a significant growth trend. Many empirical studies have shown that the proper use of AI technology can greatly improve students' learning enthusiasm and academic performance, and can also significantly reduce the teaching burden of teachers. At present, although AI technology has made preliminary attempts in primary and secondary education, it still faces many problems to achieve large-scale promotion, such as high technology investment costs, limited teacher capacity, and ethical privacy risks. With the help of innovative teaching models such as gamified learning and virtual reality technology, AI can transform abstract knowledge into more interactive and interesting teaching resources, thereby effectively improving students' classroom participation and optimizing learning effects(Nemani, 2025). Artificial intelligence-enabled educational games rely on the contextualized teaching concept, skillfully integrate the main knowledge points into the narrative, and encourage students to actively use the knowledge they have learned in interactive activities to solve problems, to achieve efficient knowledge absorption. It allows students to intuitively perceive the practical application scenarios of knowledge, which further stimulates their interest in learning and improves the overall teaching effect.

The research framework of this paper is as follows: firstly, the four major application scenarios of AI in primary and secondary education (intelligent teaching system, personalized learning platform, intelligent evaluation tool, and VR/AR teaching) are systematically sorted out, and the application effect is verified by combining empirical cases such as domestic Ape AI and Chengdu No. 7 Central and Eastern Schools. secondly, it analyzes the existing challenges from the four dimensions of "hardware-teacher-data-ethics", paying special attention to the disadvantages that AI may cause students to over-dependence; Finally, based on localized cases, a systematic path of "top-level design-data integration-teacher training-ethical supervision" is proposed, which is different from the existing policy-oriented review research, and highlights the original contribution of "case-driven + quantitative support".

II. THE APPLICATION STATUS OF 2.AI IN PRIMARY AND SECONDARY EDUCATION

2.1 Research on classroom behavior analysis and teaching strategy optimization

The intelligent teaching system relies on artificial intelligence technology to conduct in-depth analysis and comprehensive evaluation of multi-dimensional data such as student performance, homework completion, and classroom behavior. Taking the intelligent teaching platform used by a primary school as an example, the system can collect and store students' learning status information (such as attention allocation, interactive participation, etc.) in real time, and then adjust the difficulty of the course in a timely manner

according to the students' learning situation. Relying on accurate data feedback, teachers can identify weak links in teaching in a timely manner and take corresponding remedial measures to improve curriculum planning. Once the system finds that students generally have difficulty understanding a certain knowledge point, teachers can adjust their teaching strategies in time, use some typical examples or more novel teaching methods to deepen students' understanding of knowledge, improve their learning effect, and then improve the quality of teaching.

2.2 Adaptive learning path and precise resource push strategy

The personalized learning platform relies on artificial intelligence technology to deeply analyze the key elements of students' learning behavior, knowledge mastery, and ability development. Taking the personalized learning system of the empirical results of Xiaoyuan AI at the China Service Trade Fair as an example, the platform will integrate students' initial assessment information with their usual learning path to tailor a learning plan for each user(Wei & Liu, 2023). During the learning period, the system can keep abreast of students' mastery of various knowledge points and progress trends, and then adjust the difficulty of the course in a timely manner according to these situations, and accurately push learning materials such as appropriate practice questions and teaching videos. This approach fully meets the needs of individual differences and significantly improves the overall learning effect and efficiency.

In order to verify the actual supporting role of adaptive learning paths in "reducing burden and increasing efficiency", empirical cases of domestic education technology enterprises provide a quantitative basis. The figure below compares the core efficiency indicators of Ape AI and the traditional learning model, which shows that AI significantly shortens the time of ineffective practice by accurately matching learning conditions, and at the same time improves students' willingness to actively learn.

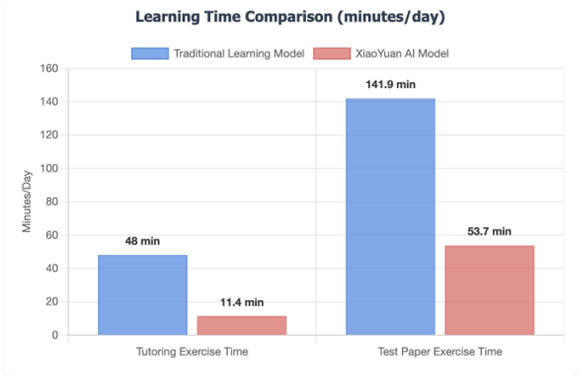

Figure 1 Comparison of learning duration

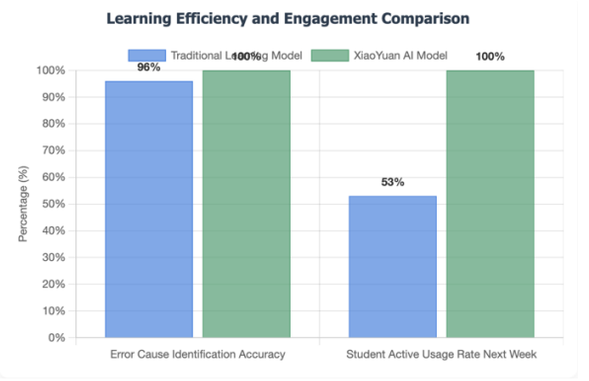

Figure 2 Learning Efficiency and Engagement Comparison

Indicator	Traditional Learning Model	XiaoYuan AI Model	Improvement/Reduction	Sample Description
Tutoring Exercise Time (minutes/day)	48	11.4	Reduced by 76.25%	Random sample of primary and secondary students n=2000
Test Paper Exercise Time (minutes/day)	141.9	53.7	Reduced by 62.16%	Random sample of primary and secondary students n=2000
Error Cause Identification Accuracy	96%	Close to 100%	Significantly Improved	All-subject homework correction data
Student Active Usage Rate Next Week	53%	Close to 100%	Significantly Improved	Platform user behavior statistics

Table 1 Empirical data on the optimization of the AI adaptive learning path of Xiaoape (Data source: Jintai Information, 2025)

In summary, it shows that the optimization of Xiaoape's AI adaptive learning path has significantly improved in key indicators compared with the traditional learning mode: the teaching aid and test paper practice time have been shortened by 76.25% and 62.16% respectively, and the accuracy of error cause localization and the active use rate of students are close to 100%. In terms of interactive experience, it supports mouse hover to view data, legend click to switch display, and adopts responsive design to ensure optimized browsing effects on different devices. The overall data is clearly labeled, fully reflecting the positive impact of AI technology on learning efficiency.

2.3 Automatic correction and academic situation diagnosis

The intelligent assessment system integrates natural language processing and machine learning technology to automatically assess students' academic performance. It can accurately identify and label grammatical errors, spelling mistakes, and logical problems in the composition, and give detailed feedback and scoring. Its main function is to comprehensively and objectively analyze students' academic performance, and then form a learning report covering many aspects such as knowledge mastery and ability growth path, providing data support for teachers' decision-making(L. Kong et al., 2025). This tool greatly reduces the workload of teachers' manual review, allowing them to focus more on improving instructional design and developing personalized instructional

plans, thereby improving the quality of education and promoting the healthy growth of students.

2.4 VR/AR scenario-based teaching

Virtual reality (VR) technology has brought an innovative learning model to the field of primary and secondary education. Relying on special equipment, students can more deeply integrate into multi-dimensional teaching situations. In the history subject, with the help of VR devices, students can "immerse" themselves in specific historical scenes and truly feel the atmosphere of that era; In science classes, students deepen their understanding by visually observing the motion trajectories of planets in the solar system or the internal structure of cells in virtual scenarios(Y. Kong, 2021). Such an immersive teaching method not only stimulates students' desire for knowledge, but also significantly improves students' academic performance and comprehensive quality.

The scenario-based benefits of VR/AR technology can be further verified by student engagement data. The chart below shows that after the introduction of VR interactive teaching in Beijing No. 2 Experimental Primary School, core indicators such as students' active questioning rate and reading time have increased significantly, confirming the stimulating effect of AI technology on students' learning initiative.

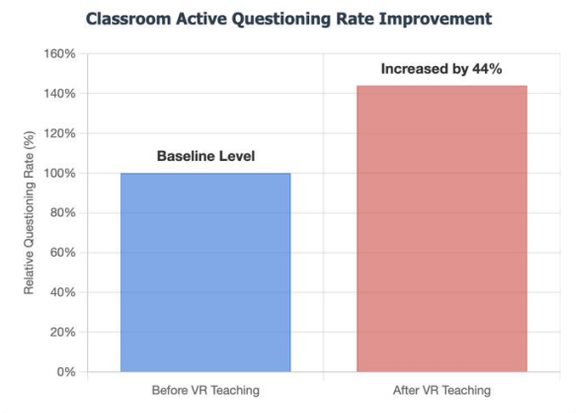

Figure 3 Comparison of the increase in the rate of active questioning in the classroom

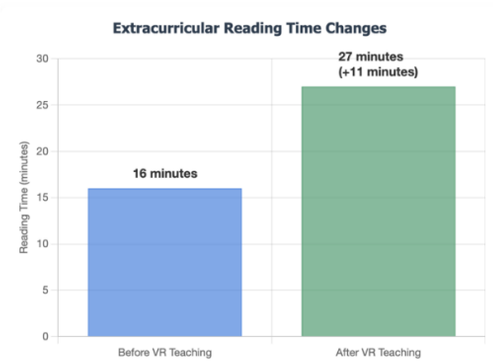

Figure 4 Comparison of changes in extracurricular reading time

Indicator	Before VR Teaching	After VR Teaching	Improvement	Explanation
Preschool Children's Active Questioning Rate	Baseline Level	Increased by 44%	+44%	Through Robots + VR Scenario Teaching
Daily Extracurricular Reading Time (Primary/Middle School Students)	16 minutes	27 minutes	+11 minutes (68.8% growth)	VR scenarios stimulate interest
Student Recognition of VR Teaching	-	89%	-	Believe VR scenarios "help understand abstract knowledge"

Table 2 Comparison of student participation before and after VR teaching in Beijing No. 2 Experimental Primary School (data source: Digital China Construction Summit, 2025).

The chart intuitively shows the significant increase in student participation of Beijing No. 2 Experimental Primary School after "robots into the campus + VR scene teaching", among which the active questioning rate of preschool children increased by 44%, the extracurricular reading time of students increased by 11 minutes (growth rate of 68.8%), and the students' recognition of VR teaching reached 89%; At the same time, the charts have practical interactive functions (mouse hover to display detailed data, legends can click to toggle the show/hide of data series) and responsive design (can automatically adjust the layout on different screen sizes and optimize the display effect on mobile devices), all data is directly marked on the chart, making it easy to understand the positive impact of VR teaching on students' learning enthusiasm.

III.AI CORE ADVANTAGES OF EDUCATION

3.1 Reduce teachers' repetitive labor

After artificial intelligence technology has been widely used in the field of education, the efficiency of education management has been significantly improved. Relying on intelligent teaching platforms and automated evaluation tools, large amounts of teaching data can be quickly processed and repetitive tasks such as homework grading and grade statistics can be completed, which can free up more time and energy for teachers. Based on this, teachers can pay more attention to core matters such as curriculum design, learning situation analysis, and personalized guidance(Afifah et al., 2022). Through the in-depth mining of teaching big data, the AI system can not only provide teachers with accurate teaching suggestions, help teachers optimize classroom teaching plans, assist teachers in making decisions, but also improve the quality of education and improve teaching effectiveness.

The substitution effect of AI technology on teacher repetitive labor can be quantified through data from regional educational practices. The figure below takes the digital intelligence homework system in Jinniu District, Chengdu as an example, showing the effect of AI on shortening teachers' working hours in the three major links of homework correction, lesson

preparation, and paper grouping, providing empirical support for "burden reduction".

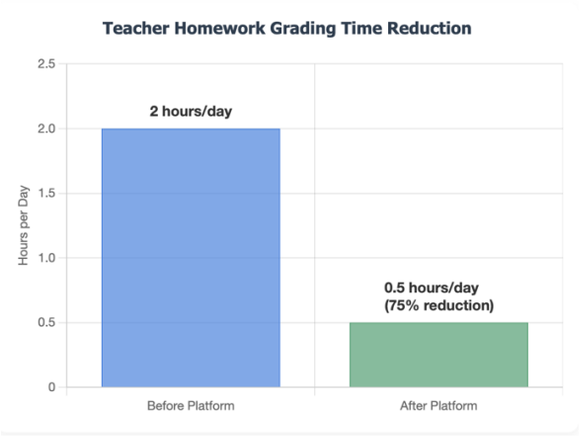

Figure 5 Comparison of teachers' homework correction time shortened

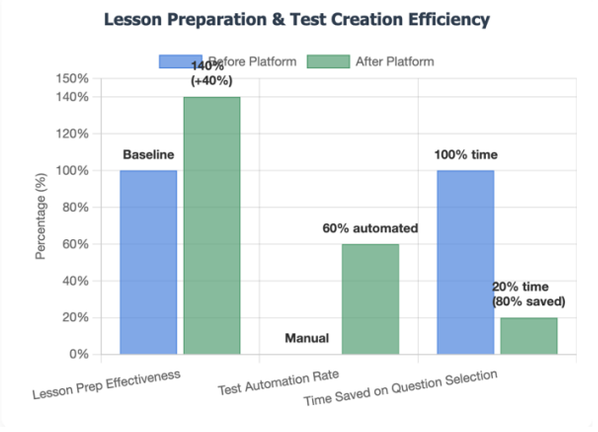

Figure 6 he efficiency of lesson preparation and paper organization has been improved

Indicator	Before Implementation	After Implementation	Improvement
Homework Grading Time (hours/day)	2 hours	0.5 hours	75% reduction
Lesson Preparation Effectiveness	Baseline	40% improvement	+40%
Test Creation Automation Rate	Manual process	60% automated	60% automation
Manual Question Selection Time	100% time required	20% time required	80% time saved

Table 3 Implementation effect of personalized "smart workbook" in eastern schools of Chengdu No. 7 Middle School (Data source: China Education News Network, 2025)

Jinniu District has achieved efficiency improvement through the smart education cloud platform, including a 75% reduction in homework correction time (from 2 hours to 0.5 hours/day), a 40% increase in lesson preparation, a 60% increase in paper automation rate, and an 80% reduction in manual topic selection time; At the same time, the chart has the interactive function of mouse-hover display of detailed data, the interactive function of clicking the legend to switch the show/hide data series, and the responsive design of automatically adjusting the layout under different screen sizes and optimizing the display effect on mobile devices.

3.2 Teach students according to their appropriateness and adjust dynamically

Through in-depth mining and analysis of learning behavior data, the intelligent system can accurately grasp the cognitive characteristics, strengths and weaknesses of students, and then plan targeted teaching plans and course content. This improves the overall learning efficiency and self-efficacy, and the system can monitor the dynamic changes in the learning process at any time, feedback the learning results in a timely manner, and promote the internalization of knowledge and optimize the learning path.

3.3 Gamification and contextual design

With the help of novel teaching forms, such as gamified learning and virtual reality, artificial intelligence technology transforms boring knowledge content into attractive interactive experiences, which greatly stimulates students' enthusiasm for classroom participation. Represented by AI-driven educational games, which integrate learning objectives into stories, prompt students to use what they have learned to solve practical problems in game scenarios, and unconsciously deepen their understanding of knowledge, these games often have instant feedback and reward mechanisms, which can arouse students' sense of competition and satisfaction, and then mobilize their enthusiasm for active learning(Ahmed, 2024). Virtual reality technology creates an immersive teaching environment for students, allowing them to intuitively feel the practical application value of knowledge, thereby stimulating students' interest and participation in learning.

IV Key challenges faced

4.1 Insufficient hardware investment and infrastructure

The application of artificial intelligence technology in the field of primary and secondary education needs to rely on the support of hardware equipment and software platforms, but there are still many problems such as technology investment and infrastructure construction, schools in economically underdeveloped areas are often unable to purchase AI-related equipment, and it is difficult to bear the follow-up maintenance costs, the normal operation of the AI system must have a stable network environment and a complete information technology support system, but some remote schools have poor network conditions and cannot meet the basic needs of technology application. These situations collectively limit the promotion and development potential of AI technology in primary and secondary education.

4.2 Lack of training system

To give full play to the potential of artificial intelligence technology in primary and secondary education, teachers need

to continue to improve their digital literacy and professional skills to achieve the effective use of intelligent tools. Nowadays, many primary and secondary school principals lack knowledge accumulation and technical support related to the field of artificial intelligence, and their understanding of artificial intelligence technology is obviously insufficient. Although a few teachers show a strong desire for knowledge, they still face many difficulties in teaching practice due to the lack of systematic training resources. Establishing a sound scientific training mechanism to improve teachers' digital literacy and artificial intelligence operation ability is a major task to promote the application of AI technology in basic education(Yang et al., 2021). At present, the field of artificial intelligence education is dominated by technology companies, and the participation of educational institutions is low. In the research on the deep integration of robots and education, most of the relevant achievements are robot-driven teaching models, focusing on cultivating students' inquiry spirit, practical ability and operational skills through robots, while there is little research on how to reduce the burden on teachers and students and improve the learning experience.

4.3 Data scattering and algorithmic bias

In our country, artificial intelligence education is in an important period of transformation from assistance to value creation, and a complete personalized application ecosystem has not yet been established. Its development difficulties are mainly concentrated in two aspects: first, the integration of data resources is low and scattered; Second, there is a lack of data governance system, resulting in uneven data quality, low credibility, and rising secondary development costs. Especially for primary school students, when deploying AI systems on a large scale, from information collection, storage to analysis, etc., people's continuous attention to the rationality of ethical norms and privacy security protection has aroused people's continuous concerns, and related concerns have become increasingly prominent(Acevedo, 2025). In this case, any potential safety hazards may have a serious impact on the legitimate rights and interests of this group. The latent bias of AI algorithms can undermine the objectivity and fairness of students' academic performance evaluation and personalized recommendations(Almethen, 2024). The in-depth application of AI technology has made related ethical issues more and more prominent, such as whether over-reliance on AI will have a negative impact on students' independent learning ability and innovative thinking cultivation. It is urgent to form a complete ethical norm system and establish a regulatory mechanism to protect students' privacy rights and reasonably limit the scope of application of AI technology, which is the main issue that must be paid attention to at present.

The ethical risks of AI technology have shown the dual characteristics of 'dependency degradation' and 'algorithmic discrimination' in practice. A multinational experiment at the University of Pennsylvania showed that improper use of generative AI can significantly weaken students' ability to learn independently - although the basic AI tool improved grades by 48% in the short term, students' test scores dropped by 17% after the tool was withdrawn, and 32% of students experienced a blunt ability to think, confirming the existence of the 'AI crutch effect'. The figure below shows how generative AI causes students' "learning dependence" and knowledge grasp deterioration (international empirical research)

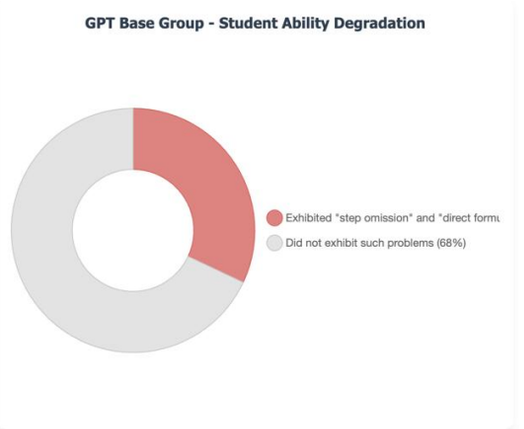

Figure 7 The ability deterioration of students in the GPT Base group

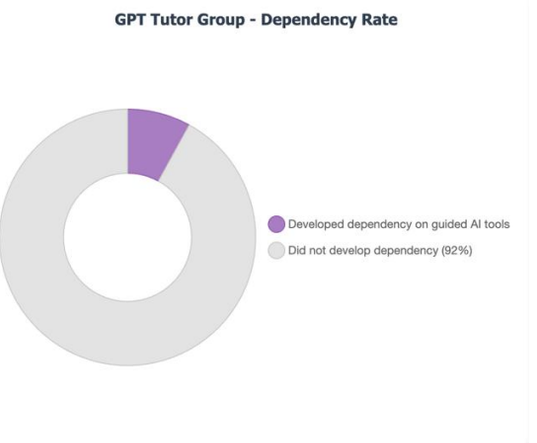

Figure 8 Dependence rate of students in the GPT Tutor group

Algorithmic bias directly impacts educational equity: due to the imbalance of training data, the admission recommendation system developed by the IIPSC in the United States has resulted in a 76% referral rate of students from low-income families to weak high schools, a gap of nearly 50 percentage points compared to students from wealthy families. Both types of cases show that AI ethical governance needs to take into account the dual dimensions of "usage guidance" and "data fairness". The high school admission recommendation system developed by the Institute for Selection and Innovation (IIPSC) in American public schools has been specially analyzed by the National Intelligent Social Governance

(Education) Characteristic Experimental Base of Peking University(Herold, 2013).

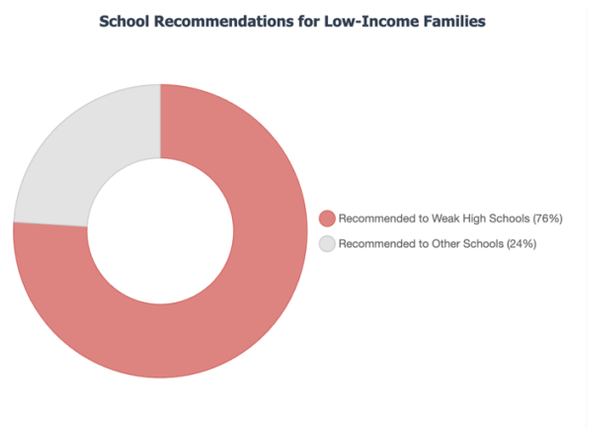

Figure 9 Distribution of recommended schools for students from low-income families

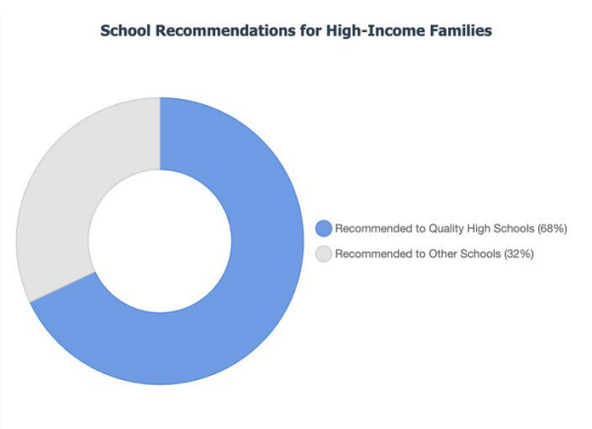

Figure 10Distribution of recommended students from high-income families

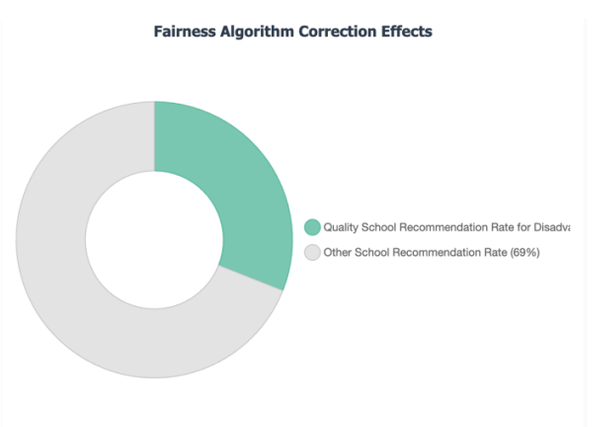

Figure 11 Correction effect of fairness algorithm

Fig. 11

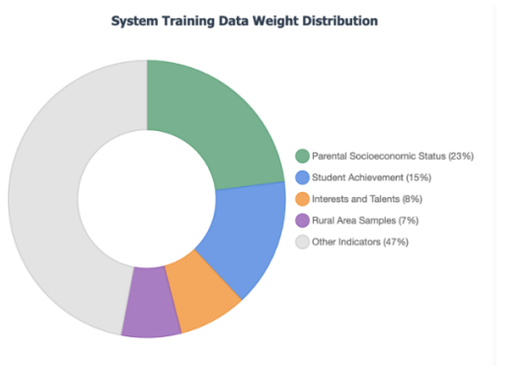

Figure 12 Weight distribution of system training data

In summary, the problem of algorithmic bias in the education system and the correction effect after the introduction of the fairness constraint algorithm are as follows: 76% of low-income families recommend weak schools, 68% of high-income families recommend high-quality schools, 23% of parents have a weight of socioeconomic status, and only 7% of the sample in rural areas, while the corrected recommendation rate of high-quality schools is 31%.

V. Systematic Development Countermeasures

5.1 Top-level design and policy support

5.1.1 Formulate regional development plans

Build a systematic top-level design framework to promote the coordinated development of education big data and artificial intelligence. According to the development plan and actual needs of Changsha's education industry, the "Guiding Opinions on the Integration and Application of Education Big Data and Artificial Intelligence in Changsha" is formulated to clarify the rights and responsibilities of the government, schools, enterprises and individuals in data collection, processing, storage, and sharing, and improve the whole life cycle management system of education data, covering key links such as data collection, cleaning, analysis, mining, and application(Gerlich, 2025). At the same time, it will promote universities, research institutes, and enterprises to strengthen cooperation, focusing on the scientific collection, in-depth analysis and innovation of machine learning algorithms of student learning behavior data, and developing intelligent tools such as "Doctor maths" to achieve accurate evaluation of students' academic performance and provide personalized learning plan suggestions.

5.1.2 Establish specialized industrial funds

Set up the "Education Big Data Industry Guidance Fund" to support leading education enterprises to establish industry-finance collaboration mechanisms(Chou et al., 2022). Use market-oriented means such as equity financing of smart education enterprises to integrate capital market resources,

promote the deep integration of the smart education industry chain, and build an internationally competitive core cluster of the domestic smart education industry.

5.2 Data integration and ecological construction

5.2.1 Build a sharing platform to solve the problem of data silos

Breaking down data barriers and promoting the open, sharing and deep integration of educational data resources, Changsha became a national pilot city for education informatization reform in 2018(Wu et al., 2022). Taking this opportunity, Changsha City strives to promote the standardization and standardized development of education data, clarify data management responsibilities, build a cross-departmental and multi-level educational data resource sharing platform, and systematically sort out the data resources of various educational institutions at all levels to achieve comprehensive data optimization and accurate governance, so as to promote the efficient collaboration and comprehensive application of business data within the education system.

5.2.2 Establish industry-teaching-research alliances

Create the "China Big Data and Artificial Intelligence Education Industry Maker Alliance" to help its sustainable development(Davis, 2024). Government departments should work closely with the alliance to jointly establish the "Smart Education Industry Expert Advisory Committee" and the "Smart Education Technology Innovation Research Base" and formulate the "Smart Education Industry Development Report", "Smart Education Technology Specification Guidelines" and "Smart Education Innovation and Entrepreneurship Evaluation Standards", so as to improve the industrial planning and performance evaluation mechanism.

5.3 Teacher training and resource balance

5.3.1 Hierarchical training system

Establish a systematic AI education and training framework for teachers, and form a comprehensive curriculum plan according to the differences in rank and actual needs, and the relevant training content should cover the basic principles of AI, technical operation methods and application scenarios, and interdisciplinary integration applications. The online part provides digital data support and interactive communication functions(Cheshmehzangi & Tang, 2024)to promote self-learning and experience exchange, and the offline link deepens the understanding of professional knowledge through expert lectures, special seminars, practical training, etc., and integrates the results of artificial intelligence training into the teachers' work performance evaluation system, which is linked to job promotion, so as to enhance the enthusiasm to participate in training and drive the continuous improvement of their own skills.

5.3.2 Tilt support weak schools

The government should increase financial investment and support for artificial intelligence education, set up special funds to fund schools to purchase hardware equipment and software resources, so as to significantly reduce implementation costs, rely on incentives to promote in-depth cooperation between science and technology enterprises and basic education institutions, develop suitable products according to the characteristics of primary and secondary education, and provide services to target groups at reasonable prices to ensure that artificial intelligence technology achieves comprehensive coverage and regular application in primary and secondary schools.

5.3.3 International case reference

Singapore's "Smart Education Program" - AI-driven cross-regional balance of teachers and resources. In 2023, Singapore's Ministry of Education launched the "Smart Education Program" to build a national educational resource database through knowledge graph technology, supporting an AI teacher training system, and focusing on solving the problems of "urban-rural teacher gap" and "insufficient resource adaptation". The Education Statistics Summary (ESD) provides essential statistics on education in Singapore. It includes statistics on schools, enrollment rates, teachers, educational outcomes, employment outcomes, and finances.

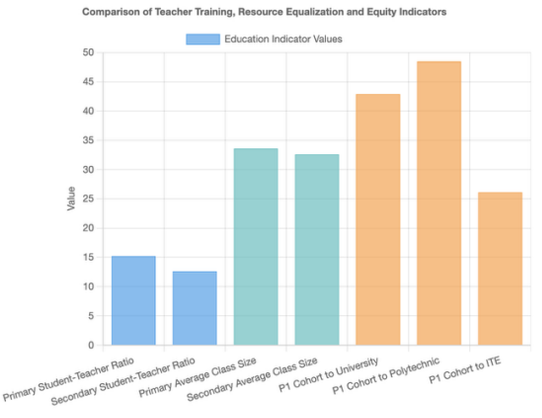

Figure 13 Comparison of teacher training, resource balance and fairness indicators in Singapore

International practice provides a replicable paradigm for AI-empowered teacher training and resource balance, and Singapore's "Technology Toolkit + Graded Training" model is a typical example - this model not only promotes the improvement of teacher capabilities through customized AI training toolkits (such as teaching adaptation modules for teachers of different subjects), but also relies on resource balancing AI algorithms to optimize the allocation of

educational resources, from the perspective of specific quantitative results, its primary school teacher-student ratio is 15.2, middle school teacher-student ratio is 12.6, and the average class size of primary school is controlled at 33.6 students The proportion of P1 cohorts admitted to universities, polytechnics, and ITE (Singapore Institute of Technology Education) is 42.9%, 48.5%, and 26.1% respectively, and the relevant charts reflecting the above-mentioned resource allocation and further education path selection can provide intuitive data support for education policy formulation and resource optimization and adjustment, which not only enriches the practical dimension of AI-enabled education balance, but also provides "hierarchical teacher training" and "resource tilt support" in the paper. and other countermeasures provide international empirical reference.

5.4 Ethical supervision and standard construction

5.4.1 Data security regulations

Build a systematic legal system for data security and privacy protection, clearly define the boundaries of power and responsibility of educational institutions and enterprises for the collection and processing of student information, ensure the security of data storage and transmission, and improve the ethical review system of artificial intelligence education products(Dieterle et al., 2024).Implement strict algorithm technology evaluation of application software within the scope of basic education to ensure the fairness and interpretability of algorithm design, so as to effectively prevent the risk of bias, strengthen the supervision of intelligent education tools, establish an independent professional supervision organization, regularly carry out comprehensive inspections of the application status on campus, quickly deal with ethical compliance issues, increase investment in public data literacy education, and improve the data security awareness and privacy protection level of teachers, students and parents.

5.4.2 Algorithm review mechanism

Promote the close integration of education big data research and application, create a core industrial ecosystem around education big data, rely on the built big data industrial park, plan a separate education big data functional area, rely on policy encouragement, financial support and other means, promote more capital to join and gather senior talents, bring together the strength of education, management, computer science, statistics and other disciplines, carry out collaborative research and development on key issues and technical difficulties in the application of education big data, and accelerate the transformation and promotion of scientific research results. In accordance with the national education development master plan, establish a cross-regional data sharing system, strengthen basic theoretical research, improve decision-making support capabilities, and build a professional think tank with international influence on education big data.

VI. Future prospects

6.1 Affective computing and interdisciplinary integration

With the help of affective computing technology, the system can instantly detect students' mood fluctuations (anxiety, tiredness or excitement), adjust the teaching plan according to the student's emotional state, and once learning fatigue is detected, it will automatically give interesting content or arrange appropriate rest periods, thereby improving learning outcomes and satisfaction. The system will also combine students' learning process with knowledge mastery to formulate personalized teaching plans and learning materials to provide more accurate support for educators.

Artificial intelligence technology is booming, the achievement of personalized learning experience has been strongly supported, can create accurate educational resources and support systems for individual differences, rely on the continuous tracking of students' learning behavior data and in-depth analysis, intelligent algorithms can continue to improve the teaching plan, so that it meets the characteristics of students at different stages, in the actual teaching process, the system can adjust the course content and its difficulty level according to instant feedback, to ensure that students are always within the appropriate cognitive test range, With interdisciplinary integration, artificial intelligence helps students break through the constraints of a single discipline and achieve the integration of multi-faceted knowledge, so as to cultivate their comprehensive quality and sense of innovation.

6.2 Resource sinking and special needs support

Artificial intelligence technology may become the main driving factor in promoting educational equity, after the online education platform combined with the intelligent assisted teaching system, high-quality educational resources can break geographical limitations and benefit more teachable people, in remote areas and rural school environments, students have the opportunity to use AI technology to obtain the same curriculum resources and services as urban children, so as to significantly narrow the gap between urban and rural education; For some students with special needs, AI technology can also implement personalized assistance solutions, such as configuring learning devices with voice interaction functions for visually impaired students, tailoring personalized improvement plans for students in need, etc., to ensure that every child can receive equal and comprehensive development opportunities. With the help of intelligent teaching systems, personalized learning platforms, virtual reality technology and other ways, AI not only improves teaching efficiency, but also enhances students' interest in independent inquiry, enhances the enthusiasm of classroom interaction, and forms a solid foundation for cultivating innovative talents that meet the needs of future social

development. At present, the promotion of AI technology in primary and secondary education still faces many problems such as insufficient technical investment, lack of teacher capacity, ethics and privacy protection, and it is urgent for the government, schools, enterprises and all sectors of society to work together to explore scientific and effective ways to solve existing problems.

VII. Conclusion

With the continuous development of artificial intelligence, its application fields continue to expand, and will be more widely used in primary and secondary school teaching in the future, artificial intelligence will be more deeply integrated into the field of primary and secondary school teaching, and artificial intelligence will develop towards intelligence, personalization and fairness. As a tool for teachers to carry out teaching work, artificial intelligence can tailor personalized learning plans for students, and to a certain extent, it can also promote the balanced allocation of educational resources and contribute to the fair development of educational resources. In the face of the development of AI technology, the education sector should seize the opportunity to tap the potential of AI technology to achieve sustainable development of primary and secondary education.

References

[1] Acevedo, K. (2025). Exploring the Impact of Generative AI to Mitigate Educator Burnout. *Electronic Theses and Dissertations*. https://digitalcommons.acu.edu/etd/925

[2] Afifah, R. N., Simanullang, T., & Madhakomala, R. (2022). VARIOUS ADVANTAGES IN EDUCATION. *International Journal of Business, Law, and Education*, *3*(2), 145–156. https://doi.org/10.56442/ijble.v3i3.65

[3] Ahmed, F. (2024). The Digital Divide and AI in Education: Addressing Equity and Accessibility. *AI EDIFY Journal*, *1*(2), 12–23.

[4] Almethen, a. (2024). Challenges in implementing artificial intelligence applications in secondary-level education: A teacher-centric perspective.), 0–0. https://doi.org/10.21608/mfes.2024.270936.1776

[5] Cheshmehzangi, A., & Tang, T. (2024). Changsha: The PuDong of Western China Through Regional Synergy and Technological Innovation. In A. Cheshmehzangi & T. Tang (Eds), *China Under Construction: Shaping Cities Through Recent Urban Transformation* (pp. 33–57). Springer Nature. https://doi.org/10.1007/978-981-97-9785-1_3

[6] Chou, C.-M., Shen, T.-C., Shen, T.-C., & Shen, C.-H. (2022). Influencing factors on students' learning effectiveness of AI-based technology application: Mediation variable of the human-computer interaction experience. *Education and Information Technologies*, *27*(6), 8723–8750. https://doi.org/10.1007/s10639-021-10866-9

[7] Davis, R. O. (2024). Korean in-Service Teachers' Perceptions of Implementing Artificial Intelligence (AI) Education for Teaching in Schools and Their AI Teacher Training Programs. *International Journal of Information and Education Technology*, *14*(2), 214–219. https://doi.org/10.18178/ijiet.2024.14.2.2042

[8] Dieterle, E., Dede, C., & Walker, M. (2024). The cyclical ethical effects of using artificial intelligence in education. *AI & SOCIETY*, *39*(2), 633–643. https://doi.org/10.1007/s00146-022-01497-w

[9] Gerlich, M. (2025). AI Tools in Society: Impacts on Cognitive Offloading and the Future of Critical Thinking. *Societies*, *15*(1), 6. https://doi.org/10.3390/soc15010006

[10] Herold, B. (2013, December 4). Custom Software Helps Cities Manage School Choice. *Education Week*. https://www.edweek.org/leadership/custom-software-helps-cities-manage-school-choice/2013/12

[11] Kong, L., Hu, C., Huang, L., Zhang, Y., Huang, W., & Huang, S. (2025). *The Double-Edged Effect of AI Use on Innovation Teaching Behavior among Primary and Secondary School Teachers in China: A Job Demands–Resources Perspective*. Research Square. https://doi.org/10.21203/rs.3.rs-6864947/v1

[12] Kong, Y. (2021). The Role of Experiential Learning on Students' Motivation and Classroom Engagement. *Frontiers in Psychology*, *12*. https://doi.org/10.3389/fpsyg.2021.771272

[13] Nemani, S. (2025). Evaluating the Impact of Artificial Intelligence on Reducing Administrative Burden and Enhancing Instructional Efficiency in Middle Schools. *Current Perspectives in Educational Research*, *8*(1), 1–16. https://doi.org/10.46303/cuper.2025.1

[14] Wei, C., & Liu, P. (2023). Artificial Intelligence Enabled Double Reduction Policy Path Analysis. *SHS Web of Conferences*, *178*, 03014. https://doi.org/10.1051/shsconf/202317803014

[15] Wu, M., Chen, R., Lv, Y., Wu, Y., & Qiu, Y. (2022). Driving forces in joint training of industry-education graduate students under regional innovation ecosystem: – A case study of Yibin. *Proceedings of the 5th International Conference on Big Data and Education*, 236–240. https://doi.org/10.1145/3524383.3524448

[16] Yang, S. J. H., Ogata, H., Matsui, T., & Chen, N.-S. (2021). Human-centered artificial intelligence in education: Seeing the invisible through the visible 见. *Computers and Education: Artificial Intelligence*, *2*, 100008. https://doi.org/10.1016/j.caeai.2021.100008