EduGenAI: A Framework for Generative AI in Computational Science Education

Vijayalakshmi Saravanan[1,2] Tamil Lakshman[2]
[1]Department of Electrical and Computer Engineering, University of Texas at Tyler, Tyler, Texas, USA
[2]Department of Electrical and Computer Engineering, University of Texas at Dallas, Dallas, Texas, USA
vsaravanan@uttyler.edu,(vijayalakshmi.saravanan,laxman)@utdallas.edu

Abstract

This paper proposes a systematic three-phase pedagogical approach for integrating generative AI tools into computational science education, encompassing Parallel and Distributed Computing (PDC), Big Data analytics, Data Science, and AI/ML domains. Based on preliminary trial experiences at the University of Texas at Tyler in courses on Parallel and Distributed Computing (PDC), Distributed Systems and Cloud Computing (DSC), and Advanced Neural Networks and AI/ML (ANN), involving 23 students with 18 maintaining consistent learning journals, we present early observations from pilot implementations across these specialized courses.

Our preliminary findings suggest potential benefits in student engagement with complex computational concepts spanning distributed systems, data analytics pipelines, and machine learning workflows, though rigorous evaluation remains ongoing. We propose a structured implementation approach that addresses key challenges, including assessment design for computational competency, faculty development needs, and maintaining academic integrity while leveraging AI assistance for learning MPI, Spark, and TensorFlow.

This work contributes a research agenda and practical guidelines for institutions considering AI integration in computational science curricula, identifying critical research questions that require systematic investigation to validate the effectiveness of AI-assisted learning.

CCS Concepts: • **Applied computing** → **Education**; • **Computer systems organization** → *Parallel architectures*; • **Computing methodologies** → *Machine learning*.

EduHiPC, Hyderabad, India

ACM ISBN 978-1-4503-XXXX-X/2025/11
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Vijayalakshmi Saravanan[1,2] Tamil Lakshman[2] . 2025. EduGenAI: A Framework for Generative AI in Computational Science Education. In *Proceedings of In conjunction with the 32nd IEEE International Conference on High-Performance Computing, Data, Analytics (HiPC 2025) (EduHiPC).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Computational science education in neural networks, big data, data science, and cloud computing faces a fundamental challenge. Students increasingly enter these specialized courses already using generative AI tools, yet most curricula have not adapted to address this reality. At the University of Texas at Tyler, we observed that 19 out of 23 students across our Computer Engineering courses on Parallel and Distributed Computing (PDC), Distributed Systems and Cloud Computing (DSC), and Advanced Neural Networks and AI/ML (ANN), had experimented with ChatGPT or similar tools for coding assistance, often with mixed or counterproductive results for domain-specific computational tasks.

This observation motivated us to develop a systematic approach rather than simply prohibiting the use of AI. The rapid evolution of AI capabilities in code generation, data analysis, and system optimization presents both opportunities and significant challenges for computational science education, particularly in teaching complex concepts like distributed computing architectures, big data processing pipelines, and machine learning model optimization [1].

Our institution has access to TACC's comprehensive computational infrastructure, including leadership-class HPC systems with MPI implementations, cloud platforms for big data processing, and extensive GPU resources for machine learning workloads. This multi-domain computational environment, while representative of many teaching institutions, creates diverse educational challenges that motivated exploring AI-assisted learning approaches.

Preliminary informal observations during Fall 2024 in our courses at UT Tyler suggested that students who received structured guidance on AI tool usage for computational science tasks showed better engagement, though we lack rigorous data to support these initial impressions. Building on these early experiences and the increasing research on

AI's role in computer science and engineering education [2], we developed the comprehensive approach presented in this paper, grounded in established pedagogical theories.

2 Related Work and Background

Recent work has explored generative AI applications in various computing disciplines [3], with some studies showing promise for programming education [4, 12, 22]. However, the specific challenges of computational science education [18], which include the complexity of debugging distributed systems, understanding big data processing pipelines, and optimizing machine learning models, have received limited attention in the AI-assisted learning literature. The few existing studies on AI in systems education focus primarily on sequential programming or web development [5]. The unique pedagogical challenges of computational science education, including the need for students to reason about parallel execution, data flow architectures, and model performance optimization, suggest that specialized approaches may be necessary.

Our teaching experience in computational science courses at UT Tyler has highlighted several recurring challenges that affect student success [17]. Students struggle to transition from traditional programming to distributed thinking, data pipeline design, and ML model architecture [8]. In our courses, approximately 70% of students initially attempt to solve domain-specific problems using conventional approaches, even after explicit instruction on distributed computing, cloud-based data processing frameworks, and neural network architectures.

2.1 Theoretical Foundation

Our approach draws upon Bloom's Digital Taxonomy [19], which extends Bloom's traditional cognitive framework to incorporate digital-age learning skills. The three-phase structure aligns with the taxonomy's progression from lower-order thinking skills (remembering, understanding) in Phase 1, through application and analysis in Phase 2, to evaluation and creation in Phase 3. This progression supports the development of critical thinking and metacognitive skills essential for effective AI collaboration in computational science [20].

Additionally, our pedagogical approach incorporates principles from Vygotsky's Zone of Proximal Development (ZPD) [21], where AI tools serve as scaffolding to support learning at the edge of students' capabilities. The gradual reduction in AI dependency from Phase 1 to Phase 3 reflects the scaffolding-fading model, where support is systematically withdrawn as learners develop competency [7].

Computational science debugging presents unique challenges [7]. Race conditions in parallel code, data skew in Spark jobs, and overfitting in ML models are difficult for novices to identify and resolve. Students in PDC and ANN often spend disproportionate time on debugging rather than learning core concepts. Traditional programming assignments often provide delayed feedback, with students not discovering performance or correctness issues until after significant implementation effort, leading to frustration and misconceptions. Many institutions lack access to large-scale computational infrastructures. Our multi-domain setup, while valuable for education, cannot demonstrate the full-scale challenges students will encounter in production environments.

3 Proposed Three-Phase Pedagogical Approach

Based on our preliminary experiences implementing AI-assisted learning in PDC, DSC, and ANN courses at UT Tyler, and analysis of computational science education challenges, we propose a structured three-phase pedagogical approach for AI integration that addresses key areas: curriculum design, assessment modification, and faculty preparation. Figure 1 illustrates the major components and their relationships.

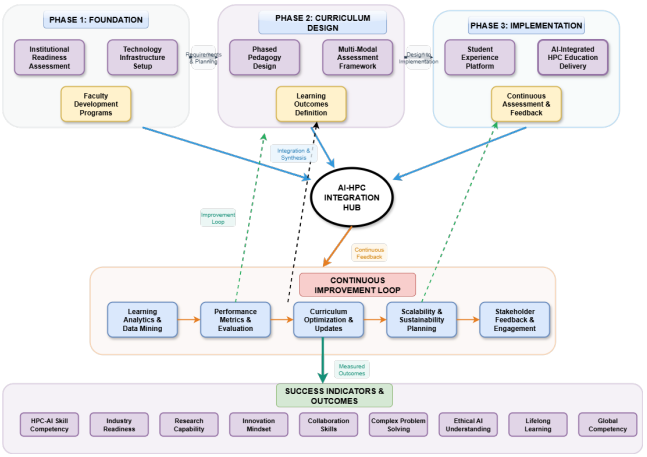

Figure 1. Proposed Three-Phase Pedagogical Approach for AI Integration in Computational Science Education. The approach progresses from foundation building without AI (Weeks 1-4), through guided AI integration with structured supervision (Weeks 5-8), to advanced AI collaboration where students work independently with AI as a partner (Weeks 9-12). Each phase builds upon the previous, developing increasingly sophisticated AI collaboration skills aligned with Bloom's Digital Taxonomy.

The approach emphasizes gradual integration rather than complete curriculum replacement, maintaining focus on fundamental computational science concepts while leveraging AI tools to address specific pedagogical challenges.

3.1 Curriculum Integration Strategy

Our curriculum integration follows a three-phase approach over a 12-week semester, with each phase building upon the previous to develop increasingly sophisticated AI collaboration skills.

Phase 1: Foundation Building (Weeks 1-4). In this initial phase, students build a strong foundation in core computational science concepts through traditional instruction without AI assistance. They begin by learning the fundamental principles of parallel programming paradigms and distributed computing architectures in PDC. The focus then shifts to big data processing frameworks such as Hadoop and Spark, along with cloud computing architectures covered in DSC. Students also explore data science methodologies that include statistical analysis and visualization techniques. In ANN, they study machine learning fundamentals such as supervised and unsupervised learning, neural network architectures, and model evaluation. Throughout this period, students engage in structured discussions on the capabilities and limitations of AI tools, preparing them for responsible and effective AI integration in the following phases.

During Phase 1, students are explicitly instructed not to use AI tools for assignments and exercises. This policy is reinforced through: (1) in-class supervised programming sessions where AI tool access is monitored, (2) emphasis on the importance of building foundational knowledge without AI dependency, and (3) assessment structures focused on demonstrating independent understanding. While we acknowledge that complete enforcement is challenging in take-home assignments, the combination of supervised sessions, clear communication of learning objectives, and honor code agreements helps establish Phase 1 as an AI-free foundation period. Students' pre-course survey responses (Section 4) indicate that most had limited computational science experience, making independent completion of Phase 1 exercises feasible without AI assistance.

Phase 2: Guided AI Integration (Weeks 5-8). In this phase, students begin to incorporate AI tools into their learning through carefully structured and supervised tasks. They use AI to clarify complex concepts in distributed computing, cloud architectures, and neural network design. Students also engage in AI-assisted code reviews to identify potential issues such as race conditions, data pipeline inefficiencies, and model architecture limitations. In addition, they receive guided support for debugging through structured interactions with AI systems to troubleshoot parallel programs, Spark jobs, and machine learning models. This phase emphasizes responsible use of AI as a learning companion rather than a replacement for human problem-solving, helping students understand both the potential and the constraints of AI assistance.

Key to this phase is teaching students to formulate effective domain-specific queries. For example, rather than asking "Fix my Spark job," students in DSC learn to ask: "I'm seeing data skew in this Spark application where certain partitions take much longer to process. What partitioning strategies might resolve this performance issue?" Similarly, students in ANN learn to query about neural network convergence issues and overfitting problems with proper context.

Phase 3: Advanced Collaboration (Weeks 9-12). Students engage in complex, cross-domain projects where AI acts as a collaborative partner throughout the workflow. They conduct performance optimization studies with AI-assisted analysis for parallel applications in PDC and explore cloud-based big data processing in DSC. The work extends to large-scale computational projects that integrate parallel and distributed computing, data analytics, and machine learning model deployment in ANN. Students also carry out critical evaluations of AI-generated outputs, assessing the quality of parallel algorithms, data processing pipelines, and neural network architectures. The experience enhances their ability to collaborate effectively with AI systems while applying domain expertise and maintaining analytical rigor.

Throughout all phases, students maintain learning journals documenting their AI interactions, successful strategies, and encountered limitations specific to each computational area.

3.2 Assessment Framework Modifications

Traditional assessment methods become less effective when students have access to AI assistance. To address this challenge, we introduce a multi-modal assessment strategy designed to evaluate both conceptual understanding and responsible AI use. The approach begins with foundational assessments conducted through closed-book examinations that test core concepts in parallel and distributed computing, big data, data science, and machine learning. These assessments are performed without AI access to ensure students demonstrate independent mastery of fundamental computational science principles. The next component emphasizes process-focused evaluation, where students submit not only their code but also detailed documentation of their problem-solving process, the AI queries they employed, and their critical reflection on AI-generated suggestions. Additional evaluation methods include live coding sessions with in-class programming exercises that allow instructors to observe students' reasoning and interaction with AI tools. Finally, peer review activities encourage students to assess AI-generated solutions from anonymous submissions, helping them develop analytical skills for evaluating parallel programming strategies, cloud pipeline designs, and neural network architectures.

4 Preliminary Experiences

During Fall 2024, we conducted a limited pilot with 23 students enrolled across PDC, DSC, and ANN courses at UT Tyler and UT. These courses typically enroll 20-30 students total, primarily junior and senior computer science and computer engineering majors with varied computational backgrounds.

We collected multiple forms of data throughout the semester, including: (1) pre- and post-course surveys measuring student AI tool familiarity and attitudes (n=23 students), (2) weekly learning journals documenting AI interactions and effectiveness (n=18 students who maintained consistent journals throughout the semester, representing 78% of the cohort), (3) assignment performance metrics comparing AI-assisted vs. traditional problem sets (n=23), (4) mid-semester and final examination scores on foundational concepts with AI-free assessments (n=23), and (5) instructor observations during live coding sessions (n=23). The 18 students who consistently maintained weekly journals generated 312 total journal entries over the semester, providing rich qualitative data on AI interaction patterns, problem-solving strategies, and learning progression. The five students who did not maintain consistent journals still participated in all surveys, assignments, and examinations.

While our implementation was limited in scope and lacked proper control groups, we gathered preliminary quantitative and qualitative data that provide initial insights into AI-assisted learning effectiveness. The findings presented should be considered exploratory rather than definitive.

Methodological Considerations and Future IRB Approval. This preliminary study represents trial observations from the Fall 2024 course implementation conducted as part of ongoing pedagogical development. As this was implemented as a trial pedagogical approach within standard coursework, formal IRB approval was not obtained for this initial observation phase. However, we acknowledge that future systematic research will require proper IRB approval. We have planned to seek IRB approval for the prospective controlled study scheduled for Fall 2026, which will include formal consent procedures and systematic data collection protocols.

We acknowledge several important methodological and ethical considerations. The trial nature of this implementation means our data collection was informal and embedded within normal course activities. Students were informed that their learning journals and feedback would help improve course design, but formal research consent was not obtained. This limits our ability to make strong research claims but allows us to share preliminary pedagogical observations that may benefit other educators. The absence of a control group limits causal inferences, as observed improvements may result from natural learning progression rather than AI integration specifically. Five students discontinued journal submissions, citing time constraints (three students) and preference for other reflection methods (two students); these students showed comparable performance on assignments and examinations (mean scores differed by less than 5 percentage points from journal-maintaining peers), suggesting that their exclusion from qualitative analysis does not substantially bias our findings.

4.1 Coding Methodology for Journal Entries

Journal entries were analyzed using a structured qualitative coding approach. Two researchers independently coded all 312 journal entries using a predetermined coding scheme developed through pilot analysis of 50 entries. The coding scheme categorized AI queries into three mutually exclusive types: (1) Complete Solution Requests: queries asking AI to generate entire solutions with minimal student input (e.g., "Write the full MPI code for matrix multiplication"), (2) Conceptual Questions: queries seeking understanding of concepts or design strategies (e.g., "Why does my Spark job have data skew and what partitioning approaches exist?"), and (3) Debugging Assistance: queries requesting help identifying or fixing specific errors with context provided (e.g., "I'm getting a deadlock in my MPI code at this barrier—what might cause this?").

Inter-rater reliability was assessed using Cohen's kappa, yielding =0.83, indicating substantial agreement. The seven ambiguous entries that did not clearly fit one category were resolved through discussion between coders. Each journal entry was assigned to a single category based on the dominant query type; if an entry contained multiple queries, the category was determined by the primary focus of the entry as indicated by word count and student reflection.

4.2 Quantitative and Qualitative Results

Table 1 summarizes key findings from our pre- and post-course surveys. Students showed increased confidence in their ability to debug complex computational problems across all domains, with the most notable improvement in parallel programming (PDC) where confidence increased from 2.3 to 3.8 on a 5-point scale. These results should be interpreted cautiously because there is no control group, and confidence increases are expected in any well-designed course regardless of AI integration. The value of these metrics lies not in demonstrating AI effectiveness, but in establishing baseline measurements for future controlled studies.

We compared performance on similar difficulty assignments before (Weeks 3-4) and after (Weeks 9-11) structured AI integration. Students showed improved completion rates, defined as submitting all required assignment components with at least minimal functionality, increasing from 73% to 91% ($p = 0.04$, Fisher's exact test).

Table 1. Pre- and Post-Course Survey Results (n=23). Without a control group, these confidence changes cannot be attributed specifically to AI integration, as they reflect the expected learning progression in computational science courses and provide baseline measurements for future controlled studies. No correction for multiple comparisons was applied, and all p-values should be considered exploratory given the small sample size.

Metric (1-5 scale)	Pre	Post
Confidence in parallel programming	2.3	3.8
Confidence in cloud/big data	2.7	4.1
Confidence in neural network design	2.5	3.9
Understanding when to use AI help	2.1	4.2
Ability to critically evaluate AI output	1.9	3.7

Without a control group, this improvement cannot be attributed to AI integration alone. It may reflect natural learning progression, increased student comfort with course material, or the scaffolding effect of accumulated knowledge. This metric captures submission completeness rather than correctness or depth of understanding; the increase could reflect either improved learning efficiency or greater surface-level completion through AI assistance without corresponding deep learning. Maintained examination performance (76% versus 78%) provides some reassurance that improvements in completion did not come at the expense of fundamental understanding, although more nuanced assessments may be needed to fully evaluate learning quality.

Analysis of 312 learning journal entries from 18 students (78% of the cohort) revealed that AI tool usage evolved over the semester. In Weeks 5-6 (early Phase 2), 68% of AI queries requested complete solutions. By Weeks 10-11 (Phase 3), this decreased to 31%, with students increasingly asking conceptual questions (42%) and debugging assistance (27%). Figure 2 illustrates this progression across the three course phases. This evolution in query patterns may reflect the pedagogical structure itself, rather than or in addition to genuine behavioral change. Phase 3 assignments were designed to require more conceptual understanding and complex debugging, which may have prompted different types of queries regardless of student development. Future research with control groups can disentangle assignment design effects from genuine learning progression.

The 18 students who maintained consistent journals reported varying AI effectiveness (5-point scale averaged from weekly journal entries): parallel programming debugging (3.2/5), cloud architecture design (4.1/5), neural network hyperparameter tuning (3.8/5), and data pipeline optimization (3.9/5). The lower rating for parallel programming reflects AI tools' limitations with race

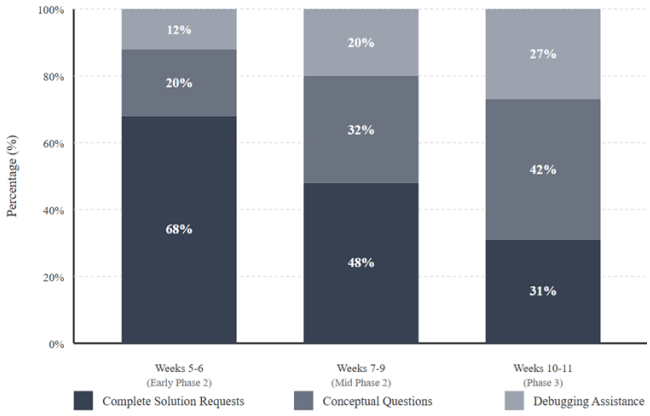

Figure 2. Progression of AI Query Types Across Semester (n=312 journal entries from 18 of 23 students). The graph shows the percentage distribution of query types: Complete Solution Requests (dark), Conceptual Questions (medium), and Debugging Assistance (light) across Weeks 5-6, 7-9, and 10-11. Journal entries were coded independently by two researchers using a predetermined coding scheme (Cohen's kappa=0.83, indicating substantial inter-rater agreement); the three categories were mutually exclusive and summed to 100% at each time point, with rare ambiguous entries (n=7) resolved through discussion. The observed shift from complete solution requests to more sophisticated query types may reflect both the pedagogical structure (Phase 3 assignments required more conceptual work) and potential student development in AI interaction skills.

conditions and deadlock detection. Students documented a total of 890 AI queries across all 312 journal entries, with an average of approximately 3 AI interactions per journal entry, though documentation completeness likely varied across students and some interactions may not have been recorded.

Analysis of learning journals, student interviews, and instructor observations revealed several important patterns. Students demonstrated measurably increased willingness to attempt challenging computational problems when AI assistance was available. In DSC, the average number of attempted optional advanced problems increased from 1.3 to 2.8 per student after Phase 2 introduction (n=23 students). In DSC and ANN, students reported feeling "less intimidated" by complex debugging scenarios (mentioned by 16 of the 18 consistent journal respondents).

Students progressed through distinct stages of AI interaction. Early attempts (Weeks 5-6) showed problematic patterns where students copied complete AI-generated solutions without understanding. One representative journal entry from DSC stated: "I asked ChatGPT to write the entire MPI broadcast code and it worked, but I couldn't explain why." By Weeks 10-11, students demonstrated more sophisticated queries. An ANN journal entry showed growth:

"Instead of asking for complete code, I explained my CNN architecture issue and asked what regularization techniques might help with the validation loss plateau I'm seeing."

We documented 47 instances where students recorded receiving flawed solutions (representing 5.3% of the 890 total AI queries documented in journals, though students may not have recorded all interactions). In DSC, the most common issue involved inappropriate MPI synchronization patterns that avoided deadlock but introduced performance bottlenecks (14 instances). AI tools frequently suggested inefficient Spark transformations that caused data skew (11 instances). In ANN, poorly regularized neural network architectures were common (13 instances), while suboptimal algorithm implementations masked underlying optimization issues (9 instances).

A key finding was students' growing ability to evaluate AI-generated code. In Week 6, only 4 of 23 students (17%) identified a deliberately flawed AI-suggested MPI solution during a classroom exercise. By Week 12, 17 of 23 students (74%) correctly identified and explained issues in a similar complexity AI-generated parallel algorithm ($p<0.001$, McNemar's test). While this 57 percentage point improvement is statistically significant, the absence of a control group prevents us from determining whether gains resulted from structured AI training specifically, general learning progression, or their combination. The finding demonstrates that students' critical evaluation abilities improved substantially over the semester, but controlled studies are necessary to isolate the role of AI integration.

Students reported mixed effectiveness for debugging support in post-course surveys. Syntax error identification was highly valued (mentioned positively by 21 of 23 students), with self-reported average debugging time for syntax issues decreasing from 45 minutes to 12 minutes. However, AI tools struggled with domain-specific issues. In PDC, only 3 of 23 students reported AI successfully helping with deadlock detection in their journal entries. In DSC, data skew diagnosis remained challenging (6 of 23 students reported successful AI assistance). In ANN, the overfitting diagnosis showed moderate success (13 of 23 students found AI interactions helpful for this task).

4.3 Case Study

To demonstrate the practical implications of structured AI integration, we present an anonymized case study from DSC. While Student S7 represents a successful example and may not be typical of all students (mean improvement in assignment correctness across all students was 15 percentage points compared to S7's 27 percentage points), this case illustrates observable patterns of AI usage development. Student S7, one of the 18 consistent journal respondents, was working on a parallel matrix multiplication assignment using MPI.

In Week 5 (early Phase 2), S7's initial journal entry showed problematic AI dependency: "Asked ChatGPT to write the complete MPI matrix multiplication code. It compiled and ran but I don't understand the scatter/gather logic." The student submitted functional code but struggled to explain the communication patterns during an in-class discussion.

By Week 8 (late Phase 2), S7's approach had evolved: "Instead of asking for complete code, I wrote the basic structure myself and asked ChatGPT specific questions: 'How should I partition the matrix for optimal load balancing?' and 'What's the difference between MPI_Scatter and MPI_Scatterv for uneven distributions?'" The student's journal documented understanding these concepts and successfully implementing a load-balanced solution.

In Week 11 (Phase 3), S7 demonstrated sophisticated AI collaboration: "My initial implementation had poor scalability. I analyzed the performance profile myself, identified the communication bottleneck, then asked AI: 'For a block-distributed matrix multiplication with n processes, how do different communication patterns (ring, tree, all-to-all) affect latency hiding?' The AI suggestions helped me understand tradeoffs, and I redesigned using a pipelined approach." S7's final project showed both strong performance and clear documentation of design decisions.

This progression—from complete solution dependence to strategic AI consultation—was common among successful students, though not universal across the entire cohort.

4.4 Faculty Development Needs

Our preliminary implementation revealed several faculty development needs [10, 13, 14]. Faculty needed training on AI capabilities and limitations in computational science education. Many instructors were unfamiliar with how to guide students in effective AI usage across these diverse computational areas. Modifying existing assignments to account for AI assistance proved more complex than anticipated. Traditional programming assignments in DSC (parallel computing and cloud processing), ANN (neural networks), and related courses became trivial with AI help, requiring fundamental redesign. Distinguishing between appropriate AI collaboration and academic dishonesty required new policies and clear guidelines [6, 9, 11] that account for the different nature of problems in these domains.

5 Research Agenda and Implementation Guidelines

Our preliminary empirical findings have identified several critical research questions that require systematic investigation with larger sample sizes and rigorous experimental design. Our preliminary data showed maintained exam performance (76% vs. 78%) despite increased AI usage, but how do we more comprehensively

assess computational science competency when students use AI assistance? What validated metrics best capture deep understanding of distributed computing, cloud data processing, and neural network model design versus surface-level AI dependency? Our finding that students progressed from 68% complete solution requests to 31% suggests behavioral changes, but does this correlate with conceptual mastery?

What is the most effective balance between traditional instruction and AI-assisted learning? Our phased approach (4 weeks foundation, 4 weeks guided, 4 weeks advanced) showed promise, but would different timing produce better outcomes? Our data showed AI effectiveness varied by domain (3.2/5 for parallel programming vs. 4.1/5 for cloud architecture)—should integration strategies differ based on these domain-specific effectiveness patterns? Do students who learn computational science concepts with AI assistance retain knowledge and skills comparably to those who learn through traditional methods?

Our case study showed Student S7 maintained problem-solving approaches after AI integration, but systematic longitudinal studies are needed extending beyond individual courses to measure retention at 6-month and 1-year intervals. Our finding that students' ability to identify flawed AI solutions improved from 17% to 74% is encouraging, but what instructional approaches best develop these critical thinking skills? Can this transfer to evaluating non-AI solutions and professional code review contexts? While syntax error debugging time decreased dramatically (45 to 12 minutes), domain-specific debugging remained challenging (only 13% success for deadlock detection). How can we better support students in developing advanced debugging skills in AI-augmented environments, particularly for parallel and distributed systems?

Future research must address several methodological challenges. Ethical considerations may restrict the use of control groups that are denied AI access when such tools are widely available. There is also a need for validated instruments [15, 16] capable of measuring computational science competency in ways that remain effective regardless of AI tool availability. Additionally, researchers must recognize that studies conducted at a single institution may not generalize across diverse student populations, institutional resources, or varying levels of faculty expertise.

Before implementing AI integration in computational science curricula, institutions should evaluate their readiness across multiple dimensions. Technical infrastructure must be assessed, including reliable internet access for AI tools, sufficient computing resources to support multi-domain computational instruction, and technical support for diverse educational technologies. Faculty preparation is also critical, requiring instructors to be familiar with both AI tools and computational science pedagogy. Our experience suggests that 30 to 40 hours of dedicated training are necessary for effective implementation. A robust policy framework should be established with clear guidelines for academic integrity, appropriate AI usage, and assessment procedures developed collaboratively with faculty, students, and administration. Finally, student preparation must be considered, including the assessment of technological literacy and the establishment of baseline computational science competencies.

We recommend a gradual implementation approach: Semester 1 for faculty development and policy creation, Semester 2 for small-scale pilot with volunteer instructors, Semester 3 for expanded implementation with systematic data collection, and Semester 4+ for program-wide adoption with continuous refinement. This timeline allows for iterative improvement and addresses challenges as they arise rather than attempting comprehensive implementation immediately.

6 Limitations and Future Work

Our proposed approach has several acknowledged limitations. The approach is based on preliminary trial experiences with a small student cohort (n=23 students total, with 18 maintaining consistent learning journals) in PDC, DSC, and ANN courses at UT Tyler. While we collected quantitative data (surveys from all 23 students, performance metrics, and 312 journal entries from 18 students), we lacked a proper control group, limiting causal claims about AI integration effectiveness. The observed improvements (e.g., increased confidence, reduced debugging time) may be due to natural learning progression, instructor effects, or other confounding variables.

An important limitation is that this trial implementation did not have formal IRB approval, as it was conducted as part of ongoing pedagogical development within standard coursework. While students were informed their feedback would help improve the course, formal research consent procedures were not followed. This means our findings should be viewed as preliminary pedagogical observations rather than formal research results. Future systematic research will require proper IRB approval and formal consent procedures.

The absence of a control group represents our most significant limitation, constraining any causal interpretation of our findings. Ideally, we would have randomly assigned students to AI-integrated versus traditional instruction. However, several factors prevented this design: (1) students across all sections had already begun using AI tools independently before course start, making true control conditions impossible, (2) our small cohort size (n=23) would yield inadequate statistical power even with random assignment, (3) ethical concerns arose about potentially disadvantaging students by withholding access to tools

they were already using and that are becoming standard in professional practice, and (4) administrative constraints limited our ability to offer multiple isolated sections. Future research should attempt controlled comparisons, possibly through delayed-intervention designs where control groups receive AI integration in later course modules, or through multi-institutional collaborations enabling adequately powered studies.

Much of our data relied on student self-reporting (learning journals, debugging time estimates, AI effectiveness ratings). Self-reported metrics may be subject to recall bias, social desirability bias, or misestimation. Future work should incorporate more objective measures such as code repository analysis, automated debugging time tracking, and standardized performance assessments. Our experiences at UT Tyler with specific infrastructure (32-node HPC cluster, Hadoop/Spark clusters, GPU resources) and student populations may not generalize to institutions with different resources, class sizes, or student demographics. Systematic multi-site studies are needed.

The capabilities of AI tools are evolving rapidly, which may render specific recommendations obsolete. Our study focused on ChatGPT and similar tools available in Fall 2024, so the findings may not generalize to future, more advanced systems. Although AI-free examinations indicated maintained student performance (76% versus 78%), it remains unverified whether these assessments truly measure computational science competency in an AI-augmented environment. Traditional assessment methods may fail to capture emerging skills, including effective AI collaboration, critical evaluation of AI-generated outputs, and strategic use of AI tools.

We are developing a comprehensive research program to address these limitations. This program will include a controlled study comparing learning outcomes between traditional and AI-integrated computational science instruction, planned for Fall 2026 with formal IRB approval and rigorous experimental design including proper consent procedures. We also aim to establish partnerships with other universities to evaluate the generalizability of the approach across diverse contexts and student populations. Additionally, we will follow students beyond individual courses to assess long-term retention of computational science concepts and their professional preparation. Finally, the program will focus on creating validated assessment tools that measure computational science competency while accounting for AI tool availability.

7 Conclusions

The integration of generative AI into computational science education presents both significant opportunities and substantial challenges. Our preliminary experiences implementing AI-assisted learning in PDC, DSC, and ANN courses at UT Tyler suggest that structured AI integration may enhance student engagement and provide valuable learning support, but rigorous evaluation remains necessary to validate these initial observations.

The three-phase pedagogical approach presented in this paper provides a starting point for institutions considering AI integration in computational science curricula. However, we emphasize that this work represents early-stage thinking rather than validated best practices. The computational science education community needs systematic research to understand the most effective approaches for leveraging AI while maintaining educational rigor.

Key contributions of this work include the development of a structured three-phase pedagogical approach for integrating AI across computational science domains, grounded in established learning theories. We identify critical research questions that require systematic investigation. The work provides practical implementation guidelines informed by preliminary experience at our institution. Finally, it highlights the need for new assessment approaches that are suitable for AI-augmented learning environments.

We encourage the EduHPC community to engage with these questions and contribute to a systematic understanding of AI's role in computational science education. The decisions made in the next few years will significantly influence how future generations of students learn to design, implement, and optimize distributed computing systems, cloud-based big data processing pipelines, and neural network models.

Acknowledgments

This work is supported by the University of Texas System's RISING STARS Grant Program. We thank the limited student cohorts in PDC, DSC, and ANN courses at UT Tyler who participated in our preliminary observations and provided valuable feedback on AI tool usage in computational science education.

The authors acknowledge that this represents preliminary work with limited student populations, requiring substantial additional research to validate the proposed approaches and measure their effectiveness.

References

[1] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[2] Paul Denny, Sumit Gulwani, Neil T Heffernan, Tanja Käser, Steven Moore, Anna N Rafferty, and Adish Singla. Generative AI for education (GAIED): Advances, opportunities, and challenges. *arXiv preprint arXiv:2402.01580*, 2024.

[3] Ethan Mollick and Lilach Mollick. Assigning AI: Seven approaches for students, with prompts. *arXiv preprint arXiv:2306.10052*, 2023.

[4] Changyoon Lee, Junho Myung, Jieun Han, Jiho Jin, and Alice Oh. Learning from teaching assistants to program with subgoals: Exploring the potential for AI teaching assistants. *arXiv preprint arXiv:2309.10419*, 2023.

[5] Erfan Al-Hossami, Razvan Bunescu, Justin Smith, and Ryan Teehan. Can language models employ the Socratic method? Experiments with code debugging. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 53–59, 2024.

[6] Wayne Holmes and Kaśka Porayska-Pomsta. The Ethics of Artificial Intelligence in education: Practices, challenges, and debates. Taylor & Francis, 2022.

[7] Kurt VanLehn. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational psychologist*, 46(4):197–221, 2011.

[8] Kenneth R Koedinger, Julie L Booth, and David Klahr. Instructional complexity and the science to constrain it. *Science*, 342(6161):935–937, 2013.

[9] Iason Gabriel, Arianna Manzini, Geoff Keeling, Lisa Anne Hendricks, Verena Rieser, Hasan Iqbal, Nenad Tomašev, Ira Ktena, Zachary Kenton, Mikel Rodriguez, et al. The ethics of advanced AI assistants. *arXiv preprint arXiv:2404.16244*, 2024.

[10] Cecilia Ka Yuk Chan and Katherine KW Lee. The AI generation gap: Are Gen Z students more interested in adopting generative AI such as ChatGPT in teaching and learning than their Gen X and millennial generation teachers? *Smart Learning Environments*, 10(1):60, 2023.

[11] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, et al. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 214–229, 2022.

[12] Rania Abdelghani, Hélène Sauzéon, and Pierre-Yves Oudeyer. Generative AI in the classroom: Can students remain active learners? *arXiv preprint arXiv:2310.03192*, 2023.

[13] Tanya Nazaretsky, Mutlu Cukurova, and Giora Alexandron. An instrument for measuring teachers' trust in AI-based educational technology. In *LAK22: 12th international learning analytics and knowledge conference*, pages 56–66, 2022.

[14] René F Kizilcec. To advance ai use in education, focus on understanding educators. *International Journal of Artificial Intelligence in Education*, 34(1):12–19, 2024.

[15] Dina Foster, Caitlin McLemore, Brandon Olszewski, Ali Chaudhry, Ekaterina Cooper, Laurie Forcier, and Rose Luckin. EdTech quality frameworks and standards review. Technical report, 2023.

[16] Robert E Slavin. Evidence-based education policies: Transforming educational practice and research. *Educational researcher*, 31(7):15–21, 2002.

[17] Jacob Bryant, Felipe Child, Jose Espinosa, Emma Dorn, Stephen Hall, Dirk Schmautzer, Topsy Kola-Oyeneyin, Cheryl Lim, Frédéric Panier, Jimmy Sarakatsannis, Seckin Ungur, and Bart Woord. How COVID-19 caused a global learning crisis. Technical report, 2022.

[18] Vijayalakshmi Saravanan, Anpalagan Alagan, and Kshirasagar Naik. Computational Biology as a Compelling Pedagogical Tool in Computer Science Education. *The Journal of Computational Science Education*, 11(1):45–52, January 2020.

[19] Andrew Churches. Bloom's Digital Taxonomy. Educational Origami, 2008.

[20] Lorin W Anderson, David R Krathwohl, Peter W Airasian, Kathleen A Cruikshank, Richard E Mayer, Paul R Pintrich, James Raths, and Merlin C Wittrock. A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Longman, 2001.

[21] Lev S Vygotsky. Mind in society: The development of higher psychological processes. Harvard University Press, 1978.

[22] Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton, and David J Malan. Teaching CS50 with AI: Leveraging generative artificial intelligence in computer science education. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 750–756, 2024.