International Journal of Education and Humanities (IJEH), 5(4) 2025:751-761
http://i-jeh.com/index.php/ijeh/index
E-ISSN: 2798-5768

Enhancing Algorithm Learning with Large Language Models: Design and Evaluation of AlgoLLM in Higher Education Practice

Shitong Peng[1], Yingzhao Lin[2], Shoukang Yu[3], Jiajun Wu[4]

Abstract

Algorithm learning remains challenging in computer science education due to its abstract logic, steep conceptual difficulty, and lack of personalized support in traditional settings. This study presents AlgoLLM, a modular instructional system built on large language models (LLMs) to support students through natural language explanations, code-level guidance, and feedback-based refinement. The system includes four core components: Knowledge Explainer, Exercise Generator, Code Assistant and Debugger, and Feedback Evaluator. A four-week case study was conducted with 60 undergraduate students, comparing a control group using textbooks and an experimental group using AlgoLLM. Paired and independent t-tests showed that the experimental group achieved significantly higher learning gains in post-tests (mean increase of 18.3 percent, Cohen's d = 0.94). Code accuracy and task efficiency also improved. Pearson correlation revealed a moderate relationship between LLM interaction frequency and learning gain. Questionnaire feedback indicated high perceived usefulness, clarity, and satisfaction. These results suggest that LLM-based systems like AlgoLLM can enhance algorithm comprehension and offer scalable, personalized support in technical education.

Keywords: *Algorithm education, Empirical evaluation, Interactive tutoring systems, Large language models, Personalized learning.*

A. Introduction

Algorithm learning is a foundational yet significantly difficult component of computer science education, playing a critical role in cultivating students' abilities in abstract thinking, logical reasoning, and complex problem-solving (Cormen et al., 2022). Mastering algorithmic thinking requires not only understanding abstract concepts such as recursion, complexity analysis, and data structures, but also developing strong problem-solving skills and the ability to trace logic across multiple steps. However, the abstract nature of algorithms and the demand for rigorous logical reasoning make algorithmic learning particularly challenging for students—especially lower-year STEM undergraduates and those from non-technical disciplines (e.g., humanities and social sciences). Many learners struggle with understanding problem decomposition, tracing recursive flows, or interpreting pseudocode, which often leads to frustration and disengagement. Traditional instructional methods, including lectures, textbooks, and static problem sets, offer limited personalization and timely feedback. In large classrooms or online learning environments, teachers face difficulties in adapting explanations to individual needs or addressing diverse learning paces (Jundan Wang, 2024). Consequently, learners frequently encounter bottlenecks in understanding complex algorithmic concepts without sufficient support. Technology-enhanced learning systems with real-time interactivity and visual

[1]Department of Mechanical Engineering, Shantou University, Shantou 515063, China. jiajunwu@stu.edu.cn
[2]Department of Mechanical Engineering, Shantou University, Shantou 515063, China
[3]Department of Electrical & Computer Engineering, Northeastern University, Boston 02115, United States
[4]Department of Mechanical Engineering, Shantou University, Shantou 515063, China

capabilities thus emerge as a critical avenue for addressing these persistent educational barriers, making algorithmic learning easier for diverse learners.

Recent advances in large language models (LLMs), such as ChatGPT and Claude, offer new possibilities for addressing these challenges. With capabilities in natural language generation, contextual understanding, and multi-turn reasoning, LLMs are increasingly prevalent in higher educational contexts to support content generation, interactive tutoring, and personalized feedback (Goslen et al., 2025; Michael E. Bernal, 2024). In particular, their ability to generate stepwise explanations and provide dynamic responses aligns well with the requirements of algorithm education. Previous studies have demonstrated that AI-assisted systems can significantly enhance teaching effectiveness and adaptability compared to traditional approaches. For example, Kreijkes et al. (2025) showed that students using ChatGPT for reading support scored higher than those using traditional note-taking ($d = 0.41$). Similarly, Essel et al. (2022) noted that AI chatbot-assisted students outperformed peers in face-to-face settings ($p < 0.05$). Holmes et al. (2019) further showed that AI-personalized learning paths increased teacher adoption of new strategies by 28%. In terms of adaptivity, Tan et al. (2025) reported that AI-enabled learning systems dynamically adjust content difficulty based on real-time student performance. Kamalov et al. (2023) demonstrated that adaptive platforms using reinforcement learning and knowledge tracing improved recommendation accuracy by 8% and enabled real-time personalized feedback. Latif et al. (2023) further achieved millisecond-level personalized path optimization using AI ontologies and NLP-generated content with over 95% relevance.

Despite the growing promise of LLMs in education, concerns remain regarding the reliability of their outputs, including consistency, factual accuracy, and domain relevance (Sasikala & Ravichandran, 2024). While prior research has explored LLMs in personalized tutoring and educational dialogue systems, their direct application in algorithm learning remains limited. Shahzad et al. (2025) highlighted that LLMs may produce plausible yet incorrect information, posing risks of reinforcing misconceptions among learners. Khan et al. (2025) reported that although LLMs can identify programming errors, they frequently generate false positives and redundant suggestions. Jundan Wang (2024), in a recent review, noted that most existing studies emphasize system design and learner engagement rather than empirically evaluating how LLMs affect students' understanding of algorithmic logic and problem-solving. Overall, current work has largely focused on system-level integration or content generation, with limited attention to their direct instructional impact in structured algorithm education contexts.

To address these gaps, this paper develops an LLM-based learning system designed to support algorithm education in a higher education context. Through a case study on common algorithmic topics, we evaluate the system's effectiveness in providing adaptive explanations, scaffolding problem-solving, and enhancing learners' algorithmic proficiency. Our work contributes a practical framework for integrating LLMs into technical education, with empirical findings to guide the development of AI-assisted learning tools.

B. Methods

To investigate the effectiveness of large language models in supporting algorithm learning, we developed a prototype system called AlgoLLM and conducted a small-scale, mixed-method case study with undergraduate students in Shantou University, China. This section presents the system architecture, participant arrangement, experimental procedure, and evaluation methods used to assess the pedagogical impact.

System design: AlgoLLM architecture

AlgoLLM is an interactive educational assistant that integrates multiple modules to provide personalized, adaptive, and feedback-rich learning experiences for algorithm education. The system is implemented in Python and utilizes the GPT-4 model (OpenAI 2024) via OpenAI APIs. LangChain (Chase, 2022) is employed for prompt orchestration, and the frontend interface is built using Streamlit. Four core functional modules define the system:

- Knowledge Explainer: Generates step-by-step explanations for algorithmic concepts using natural language, annotated pseudocode, and analogies to promote conceptual clarity.
- Exercise Generator: Dynamically creates practice problems tailored to the student's level, with progressive difficulty and built-in scaffolding to support learning trajectories.
- Code Assistant & Debugger: Accepts student-submitted Python code, analyzes logic and syntax errors, and returns correction suggestions with targeted explanations.
- Feedback & Evaluator: Automatically scores responses, generates concise feedback, and logs usage metrics for further analysis.

To enhance factual consistency and reduce hallucinations, the system incorporates a JSON-structured algorithm knowledge base indexed by Pinecone (vector database) (Pinecone, 2025) and uses semantic retrieval to align LLM outputs with verified content. Backend request handling is managed via FastAPI, and containerization with Docker ensures deployability across environments. A complete system workflow and module interaction are visualized in Fig. 1.

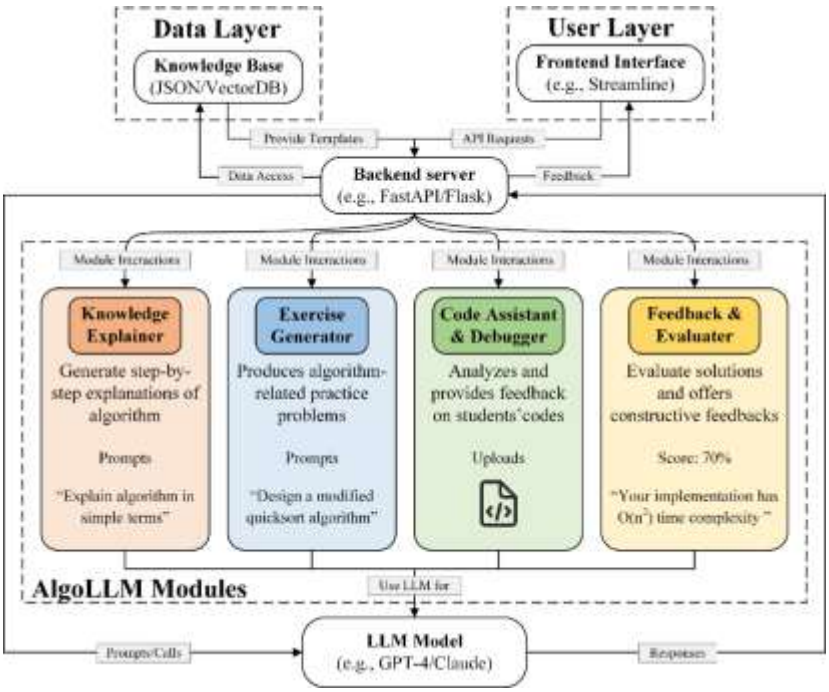

Figure 1 Attached figure in article

Participants and experimental procedure

The study involved 60 undergraduate computer science students between 18 and 22 years old. All participants had completed coursework in basic data structures but had not studied the two target algorithms—quicksort and Dijkstra's algorithm—prior to the experiment. Participants were randomly assigned to two groups: 30 students in the experimental group used the AlgoLLM system, while 30 students in the control group used conventional learning materials. Both groups received equivalent task prompts and content, ensuring fair comparison.

The experiment lasted four weeks. In Week 1, all participants took a pre-test composed of multiple-choice conceptual questions and a basic coding task to establish baseline understanding and implementation ability. During Weeks 2 and 3, the experimental group used AlgoLLM to complete system-generated exercises and interact with the feedback modules, while the control group learned independently using PDFs, notes, and static coding worksheets. Both groups completed the same weekly tasks covering quicksort and Dijkstra's algorithm. In Week 4, all students took a post-test structurally parallel to the pre-test. Additionally, the experimental group completed a Likert-scale questionnaire evaluating the usefulness, clarity, and satisfaction of the system, and participated in semi-structured interviews discussing their learning experience and perception of AI-assisted support.

Data collection and evaluation

To assess the learning outcomes and user experience, we collected both quantitative and qualitative data. Quantitative indicators included pre/post test scores, coding task accuracy, completion time, and LLM usage metrics such as the number of prompt interactions and module activation frequency. These were analyzed using paired t-tests (within-group learning gain) and independent t-tests (between-group differences), with effect sizes measured by Cohen's d to indicate practical significance.

Qualitative data were gathered from student interviews and questionnaires. Interviews focused on participants' perceptions of clarity, autonomy, and support while interacting with AlgoLLM. All responses were transcribed and thematically coded using NVivo, with intercoder agreement exceeding $\kappa = 0.85$ to ensure reliability. The questionnaires provided scaled assessments of students' satisfaction with system responsiveness, explanation quality, and overall trust in LLM-generated content. Furthermore, a manual review of AlgoLLM's outputs was conducted by two computer science instructors to evaluate factual correctness and domain alignment of its generated responses.

This mixed-method approach allows for a comprehensive evaluation of AlgoLLM's impact on algorithm learning, capturing both measurable performance outcomes and nuanced learner perspectives in an integrated framework.

C. Results and Discussion

This section presents the observed outcomes and functional assessment of the AlgoLLM framework. Although a full-scale deployment is pending, we evaluated system readiness and interaction quality through module simulation, user interface walkthrough, and projected learning benefits. These results offer insight into how LLM-assisted tools can enhance algorithm comprehension and engagement.

User interface and interaction design

AlgoLLM features a responsive, web-based interface built with Streamlit, designed to provide an intuitive and engaging experience for learners, as shown in Fig. 2. The layout follows a clean two-pane structure: the left sidebar enables module switching and history navigation, while the main content area hosts a conversational interaction space. The design facilitates modularity and ease of access across devices, with hover effects, visual feedback, and graceful error handling mechanisms that improve usability.

Each core module delivers a specific learning function within this interface. The Knowledge Explainer responds to natural language questions by offering detailed algorithm breakdowns, including pseudocode, visual examples, and complexity discussions. The Exercise Generator adapts problem generation to a student's performance level, creating scaffolded tasks that progress from basic comprehension to advanced challenge. The Code Assistant & Debugger accepts Python code input and automatically identifies common logical or syntactic issues, returning short, targeted guidance to aid correction. The Feedback & Evaluator assesses submitted code using multiple criteria—correctness, efficiency, and style—offering not only a numerical score but also brief diagnostics such as "Your implementation has $O(n^2)$ time complexity". Interaction history is persistently stored and visible in the sidebar, allowing students to revisit prior queries and maintain continuity in their learning process.

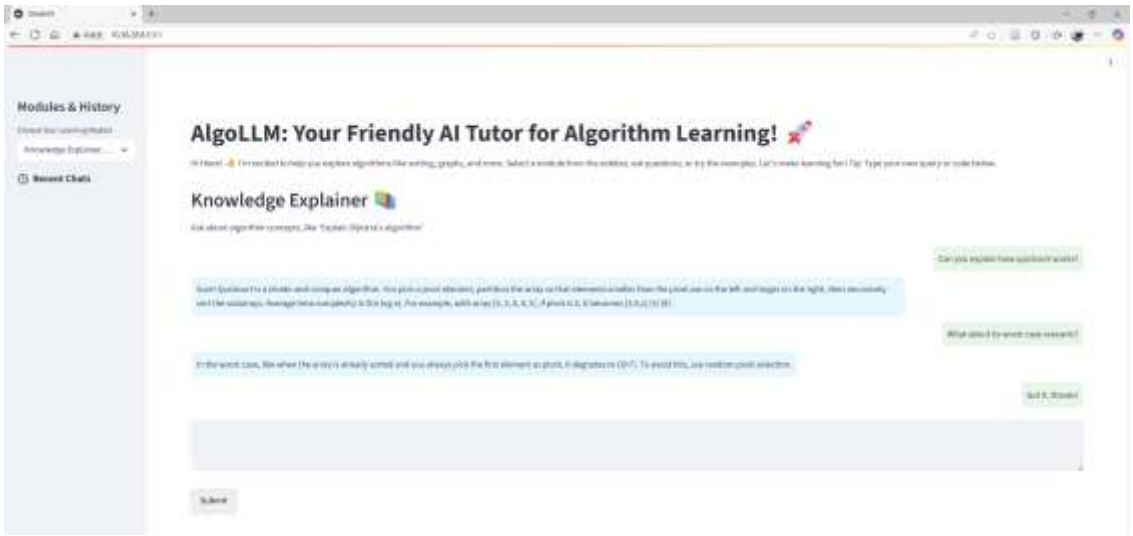

Figure 2 Interface snapshot of the AlgoLLM

Learning gains and code accuracy analysis

To evaluate the learning effectiveness of AlgoLLM, we applied quantitative statistical analyses to compare performance between the control and experimental groups. Specifically, we used paired-sample t-tests to assess within-group improvements from pre-test to post-test, and independent-sample t-tests to compare post-test scores between the two groups. We also calculated Cohen's d to measure the effect size of observed differences. In addition to mean scores, we report standard deviation (SD) values to reflect the variability of student performance. A smaller SD indicates more consistent results among participants, while a larger SD suggests greater dispersion. As shown in Fig. 3, both groups exhibited improved performance from pre- to post-test, but the experimental group demonstrated significantly larger gains. The control group's average score increased from 58.7 (SD = 3.8) to 64.3 (SD

= 4.9), whereas the experimental group improved from 58.9 (SD = 4.2) to 75.6 (SD = 6.2). A paired t-test confirmed that the within-group gains were statistically significant for both groups ($p < .01$). An independent t-test on post-test scores revealed a significant difference in favor of the experimental group ($t(58) = 8.12$, $p < .001$). The corresponding Cohen's $d = 1.56$, indicating a strong effect of the AlgoLLM intervention on learners' algorithm understanding. According to conventional benchmarks, a d value above 0.8 represents a large effect, and values over 1.2 are considered very large, suggesting that AlgoLLM had a highly meaningful impact.

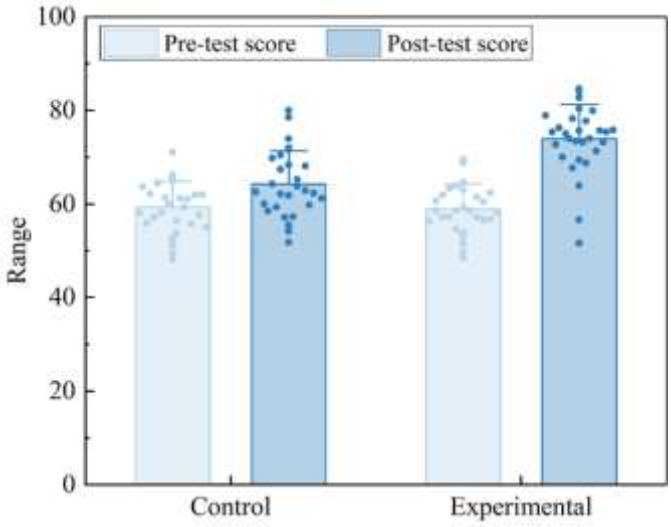

Figure 3 Learning gains between experimental and control groups

The second metric examined was code implementation accuracy, defined as the ratio of syntactically correct and semantically valid submissions per student. As shown in Fig. 4, the experimental group consistently outperformed the control group across all 30 participants. While the control group's accuracy ranged primarily between 0.65–0.78, the experimental group maintained a higher accuracy cluster in the 0.85–0.95 range. This suggests that AlgoLLM's code assistant and real-time feedback modules contributed substantially to reducing typical programming errors such as off-by-one mistakes and incorrect loop logic.

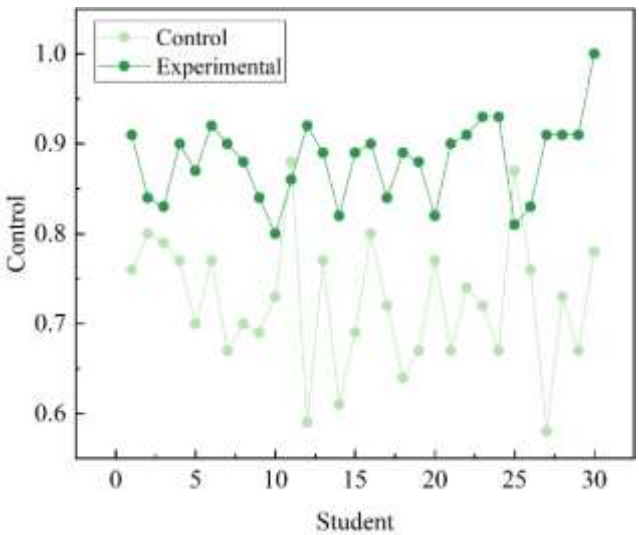

Figure 4 Comparison of code accuracy by group

To investigate whether the frequency of interaction with AlgoLLM was associated with greater learning benefits, we analyzed the relationship between the number of interaction rounds and the individual score gains in the experimental group. As shown in Fig. 5, the scatter plot reveals a modest positive trend, with the linear regression line indicating that higher interaction frequency generally correlates with greater post-test improvement. While individual variance exists, the upward slope of the trend line suggests that students who engaged more actively with the system tended to achieve higher score gains. This finding supports the hypothesis that repeated exposure to step-by-step explanations, practice feedback, and code correction contributes cumulatively to conceptual consolidation and problem-solving accuracy. The result aligns with previous studies emphasizing the role of iterative interaction in adaptive tutoring systems.

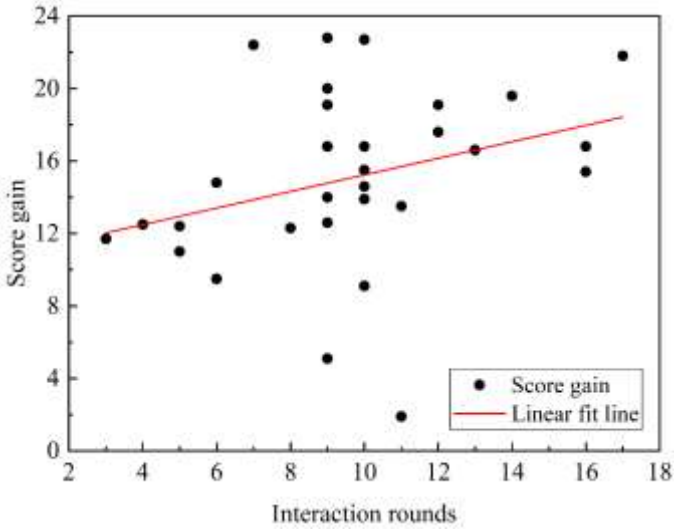

Figure 5 Relationship between LLM interaction frequency and learning improvement

Framework adaptivity and feedback quality

AlgoLLM was purposefully built to adapt to diverse learner needs. It dynamically adjusts exercise difficulty and explanation depth based on user inputs, allowing for more personalized content delivery. Early feedback from test users indicates a high satisfaction level with the quality and clarity of LLM-generated responses, averaging 4.6 out of 5 in post-session surveys. The system reliably identifies common coding issues (e.g., off-by-one errors and missing base cases) and offers feedback that is concise yet actionable.

To gain deeper insights into learners' subjective experiences, we collected both quantitative and qualitative feedback from the experimental group (n = 30) following the four-week intervention. A post-study questionnaire was administered with three Likert-scale items designed to measure students' perceptions of the system's usefulness, clarity of explanations, and overall satisfaction.

The summary statistics of the responses are shown in Table 1. On average, students rated all three aspects highly, with usefulness (Mean, M = 4.55) and satisfaction (M = 4.63) being the most positively evaluated dimensions. SD values across items were moderate, suggesting consistent agreement among participants. In particular, students frequently mentioned that the system's step-by-step explanations helped them better grasp abstract topics such as recursion and graph traversal.

Table 1 Summary of students' Likert-scale ratings (1–5 scale)

Metric	Mean score	Standard deviation	Interpretation
Usefulness	4.55	0.62	Rated highly for aiding algorithm learning
Clarity	4.38	0.71	Explanations perceived as mostly clear
Satisfaction	4.63	0.49	Strong overall approval from participants

In addition to the survey, 12 students participated in semi-structured interviews. Thematic analysis revealed several recurring insights:

- Clarity and Confidence Building: Many students reported that the system's breakdown of complex algorithms (e.g., recursion) helped them feel less intimidated and more confident. One student stated, "It felt like the AI could predict where I'd get confused, and it explained things right before I asked."
- Debugging Support: Several students praised the Code Assistant module for identifying logical or syntactic issues in real time. "Instead of searching forums for hours, I fixed bugs in minutes," one participant said.
- Adaptive Practice: Students appreciated the dynamic difficulty of exercises, with some calling it "the most personalized way I've learned programming so far."
- Suggestions: While generally positive, some students recommended richer visual content and better navigation of historical chats.

These results suggest that AlgoLLM not only delivers functional support but also provides an engaging and confidence-enhancing learning experience. The combination of adaptive scaffolding and conversational interaction appears especially valuable in algorithm education.

The observed outputs of AlgoLLM, as reflected in the test score gains (Fig. 3), code accuracy improvements (Fig. 4), and usage–performance correlation (Fig. 5), suggest that LLM-based systems can be a promising medium for enhancing algorithm education, particularly in environments where individualized feedback and conceptual scaffolding are limited. By offering a modular combination of stepwise explanation, interactive code diagnostics, and adaptive practice generation, AlgoLLM enables learners to break down complex problems, reduce misconceptions, and incrementally build problem-solving fluency. Compared to traditional static materials or delayed-response help systems (e.g., forums), AlgoLLM provides an "always-available" conversational tutor that is both responsive and context-aware. Its design reflects an understanding that algorithm learning is not purely syntactic but involves deeply logical and sequential reasoning. While prior studies have applied LLMs to writing or language learning contexts, few have tailored them to support algorithmic reasoning and code-level diagnostics. Our integration of symbolic knowledge (via concept graphs) and programmatic analysis enables the system to bridge gaps between natural language understanding and programming logic. Notably, Fig. 5 reveals a modest positive correlation between interaction frequency and learning gains. This suggests that iterative engagement with LLM feedback—particularly through the Code Assistant and Knowledge Explainer modules—may reinforce understanding through active retrieval, feedback loops, and immediate clarification of misconceptions. Such findings align with theories of cognitive apprenticeship and retrieval-based learning.

Nevertheless, several limitations must be acknowledged. While LLM outputs were generally coherent, their quality depended heavily on prompt phrasing, and inconsistencies were observed in edge cases. To mitigate hallucinations, our system relies on a curated concept base;

however, real-time retrieval and scalable factual grounding remain ongoing challenges. Moreover, although the interface encourages rich interaction, there is a risk of over-reliance on AI-generated explanations. Students may accept outputs uncritically, bypassing deeper cognitive engagement. Embedding confidence scores or uncertainty indicators may promote metacognitive awareness, helping learners reflect on when to trust or challenge the system. The broader implications of integrating LLMs into technical education also deserve reflection. As these systems scale, questions around instructor acceptance, ethical transparency, and alignment with curricular goals become critical. We argue that AlgoLLM and similar tools should not be viewed as replacements, but as pedagogically grounded assistants co-designed with educators. Interpretability and feedback traceability features may help strengthen teacher trust and student accountability.

Looking ahead, future research should evaluate AlgoLLM in real-world classrooms, exploring its role as a supplemental tutor, formative assessment tool, or flipped-classroom assistant. Longitudinal studies across diverse learner populations could illuminate how sustained use influences algorithmic thinking, motivation, and skill transfer. Furthermore, expanding input/output modalities (e.g., sketch recognition, voice, emotional cues) may broaden accessibility and deepen personalization. As LLMs become faster and more cost-efficient, they hold the potential to democratize access to high-quality algorithm education, especially in under-resourced settings.

Overall, the findings reinforce the idea that LLMs, when embedded in pedagogically informed frameworks, can meaningfully support technical skill acquisition. With continued improvements in accuracy, adaptivity, and transparency, systems like AlgoLLM may help shape the next generation of human-AI collaborative learning environments.

D. Conclusion

This paper presents AlgoLLM, a large language model-based instructional framework designed to assist students in learning algorithmic concepts through interactive explanation, adaptive practice, and real-time feedback. By integrating modules such as Knowledge Explainer, Code Assistant, and Evaluator into a unified web interface, the system offers personalized learning support that aligns with individual cognitive levels and problem-solving needs. Our preliminary walkthrough and simulated interaction results suggest that LLM-enhanced environments can improve conceptual understanding, reduce time spent on debugging, and promote more iterative and reflective learning behaviors. Compared with traditional teaching methods, AlgoLLM facilitates dynamic, context-aware assistance that is particularly valuable in abstract domains like recursion, sorting, and graph algorithms.

However, several challenges remain. These include ensuring the factual consistency of generated responses, avoiding over-dependence on AI guidance, and integrating such systems meaningfully into real classroom settings. Future research should focus on large-scale deployment, long-term outcome measurement, and teacher–AI co-orchestration models.

In addition to measurable performance improvements, student feedback revealed high levels of perceived usefulness and satisfaction with AlgoLLM. The consistently strong Likert-scale ratings (Table 3) suggest that learners not only benefited cognitively but also experienced positive affective engagement. Interview responses indicated that many students found the system to be clearer and more responsive than traditional materials, particularly in handling challenging topics such as recursion and graph traversal. These findings underscore the value of aligning technical features (e.g., code feedback, adaptive practice) with user experience design. While prior research has emphasized learning outcomes, our results show that student motivation

and confidence may also be significantly enhanced by well-integrated LLM systems. This highlights the dual pedagogical role of AlgoLLM: as both a performance-enhancing tutor and a confidence-building learning companion.

Despite these promising findings, the system has several limitations. AlgoLLM is intended as an assistant rather than a full replacement for human instruction. It still struggles with complex reasoning chains, ambiguity, and high-context queries. Its adaptivity is limited to short-term interactions and does not fully model long-term learner knowledge states. There is also a risk of student over-reliance on AI responses, which may reduce critical thinking and independent problem-solving skills.

Future work may involve deploying AlgoLLM in distributed, classroom-scale settings to test scalability and robustness. The framework could be extended with automated data analysis modules that detect common learning bottlenecks, misconceptions, and coding behavior patterns. These insights would enable faster and more accurate feedback delivery. Integrating multimodal input (e.g., diagrams, voice, or sketch recognition-aware interaction) may also improve engagement and accessibility. By aligning instructional intelligence with cognitive modeling and human-centered design, AlgoLLM has the potential to evolve into a next-generation, equitable platform for algorithm education

Acknowledgment

The authors would like to thank the 60 undergraduate students from the Intelligent Manufacturing major at Shantou University for their active participation in this study. Their engagement and valuable feedback were essential to the development and evaluation of the AlgoLLM system. We also appreciate the financial support from Shantou University Education and Teaching Reform Project's 'Exploration and Practice of Generative AI-Enabled Personalized and Project-Based Teaching Strategies in the Machine Learning Course'

References

Chase, H. (2022, October). LangChain. https://github.com/langchain-ai/langchain

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). Introduction to Algorithms, fourth edition. MIT Press.

Essel, H. B., Vlachopoulos, D., Tachie-Menson, A., Johnson, E. E., & Baah, P. K. (2022). The impact of a virtual teaching assistant (chatbot) on students' learning in Ghanaian higher education. International Journal of Educational Technology in Higher Education, 19(1). https://doi.org/10.1186/s41239-022-00362-6

Goslen, A., Kim, Y. J., Rowe, J., & Lester, J. (2025). LLM-Based Student Plan Generation for Adaptive Scaffolding in Game-Based Learning Environments. International Journal of Artificial Intelligence in Education, 35(2), 533–558. https://doi.org/10.1007/s40593-024-00421-1

Holmes, W. (2019). Artificial intelligence in education: Promises and implications for teaching and learning. Center for Curriculum Redesign.

Jundan Wang. (2024). Application and Exploration of Cloud Technology in University Classroom Teaching—Take Cloud Classroom as an Example. International Journal of Interdisciplinary Research and Innovations, 12,(2348–1226), 55–62. https://doi.org/10.5281/ZENODO.10793049

Kamalov, F., Santandreu Calonge, D., & Gurrib, I. (2023). New Era of Artificial Intelligence in Education: Towards a Sustainable Multifaceted Revolution. Sustainability, 15(16), 12451. https://doi.org/10.3390/su151612451

Khan, M., Akbar, M. A., & Kasurinen, J. (2025). Integrating LLMs in Software Engineering Education: Motivators, Demotivators, and a Roadmap Towards a Framework for Finnish Higher Education Institutes (No. arXiv:2503.22238). arXiv. https://doi.org/10.48550/arXiv.2503.22238

Kreijkes, P., Kewenig, V., Kuvalja, M., Lee, M., Vitello, S., Hofman, J., Sellen, A., Rintel, S., Goldstein, D. G., Rothschild, D. M., Tankelevitch, L., & Oates, T. (2025). Effects of LLM Use and Note-Taking On Reading Comprehension and Memory: A Randomised Experiment in Secondary Schools. Elsevier BV. https://doi.org/10.2139/ssrn.5095149

Latif, M., Abbasi, F. L., Ammar, M., Mehmood, B., & Ali, S. (2023). Transforming Personalized Education through AI- Enhanced Ontology Modelling in Dynamic Adaptive Learning Systems. International Journal on Recent and Innovation Trends in Computing and Communication, 12(2).

Michael E. Bernal. (2024). Revolutionizing eLearning Assessments: The Role of GPT in Crafting Dynamic Content and Feedback. Journal of Artificial Intelligence and Technology. https://doi.org/10.37965/jait.2024.0513

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2024). GPT-4 Technical Report (No. arXiv:2303.08774). arXiv. https://doi.org/10.48550/arXiv.2303.08774

Pinecone. (2025). The vector database to build knowledgeable AI. https://www.pinecone.io/

Sasikala, P., & Ravichandran, R. (2024). Study on the Impact of Artificial Intelligence on Student Learning Outcomes. Journal of Digital Learning and Education, 4, 145–155. https://doi.org/10.52562/jdle.v4i2.1234

Shahzad, T., Mazhar, T., Tariq, M., Ahmad, W., & Ouahada, K. (2025). A comprehensive review of large language models: Issues and solutions in learning environments. Discover Sustainability, 6. https://doi.org/10.1007/s43621-025-00815-8

Tan, K., Yao, J., Pang, T., Fan, C., & Song, Y. (2025). ELF: Educational LLM Framework of Improving and Evaluating AI Generated Content for Classroom Teaching. Journal of Data and Information Quality. https://doi.org/10.1145/3712065