International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 08 | Aug 2026 www.irjet.net p-ISSN: 2395-0072

Comparative Analysis of Large Language Models for Programming Education: An Empirical Evaluation of ChatGPT, Gemini, Claude, and DeepSeek

Mr. Manish Kumar Hooda[1], Mr. Yatender Kumar[2], Mr. Abhishek Kumar Maheshwari[3]

[1,2]Assistant Professor, Samalkha Group of Institutions, Janakpuri, New Delhi 110058
[3]Research Scholar (CSE), Sanskriti University, Mathura.

Abstract-*Large Language Models (LLMs) have emerged as powerful educational tools that assist students in programming through code generation, debugging, concept explanation, and problem-solving. As the adoption of AI-powered coding assistants continues to grow, selecting the most suitable model for programming education has become increasingly important. This study presents an empirical comparative analysis of four freely accessible Large Language Models: ChatGPT 5.5, Gemini 3.1 Pro, Claude Sonnet 5, and DeepSeek DeepThink. The evaluation was conducted using a benchmark dataset comprising programming problems from C, C++, Java, Python, SQL, HTML, CSS, JavaScript, PHP, Data Structures, and Algorithms. Each model was assessed using nine evaluation parameters: code correctness, explanation quality, debugging capability, code optimization, readability, beginner friendliness, error handling, response time, and context length. Experimental results indicate that Claude Sonnet 5 achieved the highest overall performance in code correctness, debugging capability, code optimization, and readability, making it the most technically proficient model among those evaluated. ChatGPT 5.5 demonstrated superior explanation quality and beginner friendliness, making it highly suitable for programming education and self-learning. Gemini 3.1 Pro delivered the fastest response time and supported longer contextual interactions, while DeepSeek DeepThink performed well in debugging and algorithm-oriented programming tasks. The findings suggest that no single LLM is optimal for every educational scenario; instead, each model offers distinct advantages depending on the learning objectives and programming tasks. This research provides practical guidance for students, educators, and academic institutions in selecting appropriate AI-assisted programming tools and offers a reproducible evaluation framework for future comparative studies on Large Language Models in programming education.*

Key Words: Large Language Models (LLMs), ChatGPT 5.5, Gemini 3.1 Pro, Claude Sonnet 5, DeepSeek DeepThink, Programming Education, Code Generation, Debugging, Generative Artificial Intelligence, Comparative Analysis, Computer Science Education.

1. INTRODUCTION

The rapid advancement of Artificial Intelligence (AI) has transformed many aspects of education, particularly in the field of computer science and programming. Among the most significant developments is the emergence of **Large Language Models (LLMs)**, which are capable of understanding natural language, generating human-like text, writing computer programs, explaining complex concepts, and assisting users in solving technical problems. In recent years, AI-powered tools such as **ChatGPT, Gemini, Claude, and DeepSeek** have become widely accessible and are increasingly being used by students, educators, and software developers.

Programming is a fundamental skill for students pursuing Computer Science and Information Technology. However, learning programming can be challenging, especially for beginners who often struggle with syntax, logic building, debugging, and understanding programming concepts. Traditionally, students relied on textbooks, classroom lectures, and online tutorials to overcome these difficulties. Today, Large Language Models provide an interactive learning environment where students can receive instant explanations, generate code examples, identify programming errors, and obtain personalized guidance. This has significantly changed the way programming is taught and learned.

Although these AI-based coding assistants offer similar functionalities, their performance varies depending on the programming language, prompt complexity, and the type of task being performed. Some models may generate highly accurate and optimized code, while others may provide better explanations that are easier for beginners to understand. Likewise, certain models perform well in debugging existing code, whereas others are more effective in solving algorithmic or conceptual problems. These differences make it difficult for educators and students to determine which model is most suitable for programming education.

Most existing studies focus on evaluating LLMs from a general perspective, such as conversational ability, language understanding, or software development support. Comparatively fewer studies have examined their

effectiveness specifically in programming education using a consistent and structured evaluation framework. Furthermore, many comparisons emphasize only code generation accuracy and overlook important educational aspects such as explanation quality, readability, beginner friendliness, debugging support, and response efficiency.

To address this gap, this research presents a comparative analysis of four popular Large Language Models **ChatGPT, Gemini, Claude, and DeepSeek** for programming education. The study evaluates these models using a diverse benchmark consisting of programming problems from **C, C++, Java, Python, SQL, HTML, CSS, JavaScript, PHP, Data Structures, and Algorithms**. Each model is assessed using predefined evaluation criteria, including **code correctness, explanation quality, debugging capability, code optimization, readability, beginner friendliness, error handling, and response time**.

The primary objective of this research is to identify the strengths and limitations of each LLM in supporting programming education and to provide practical recommendations for students, educators, and academic institutions. The findings are expected to help educators integrate AI tools more effectively into teaching and learning while enabling students to select the most appropriate coding assistant according to their learning needs. In addition, the proposed evaluation methodology can serve as a useful reference for future research on AI-assisted programming education and the educational applications of Large Language Models.

2. LITERATURE REVIEW

The rapid development of Large Language Models (LLMs) has significantly influenced the field of programming education. These models are increasingly being used as intelligent learning assistants to support students in code generation, debugging, algorithm design, and concept explanation. Recent studies have shown that AI-powered programming assistants can improve learning efficiency by providing instant feedback and personalized guidance. Consequently, researchers have begun investigating the educational effectiveness of various LLMs in software engineering and computer science education.

Early research on AI-assisted programming primarily focused on automatic code generation and code completion. These studies demonstrated that transformer-based language models could generate syntactically correct source code for multiple programming languages. However, researchers also observed that generated code may contain logical errors, security vulnerabilities, or inefficient implementations, indicating that human verification remains essential.

With the introduction of conversational Large Language Models such as ChatGPT, the scope of AI-assisted learning expanded considerably. Recent studies have reported that these models can explain programming concepts in simple language, generate programming examples, identify coding errors, and provide debugging suggestions. Their conversational nature allows students to ask follow-up questions, making them valuable educational tools, particularly for beginners who require additional learning support outside the classroom.

Several comparative studies have evaluated the performance of different LLMs in software development tasks. Most comparisons have focused on code generation accuracy, benchmark performance, and problem-solving capability. While these studies provide valuable insights into the technical performance of individual models, they often overlook educational aspects such as explanation quality, readability of generated code, beginner friendliness, and learning effectiveness. Since programming education involves both technical correctness and conceptual understanding, these educational factors should also be considered during evaluation.

Researchers have also highlighted certain limitations of Large Language Models. Although LLMs can produce high-quality programming solutions, they occasionally generate incorrect or misleading information, commonly referred to as hallucinations. In some cases, AI-generated code may compile successfully but fail to satisfy the actual problem requirements. Furthermore, different models exhibit varying strengths across programming languages and problem complexities, making it difficult to identify a universally superior model for educational use.

Recent advancements have introduced several competing LLMs, including Gemini, Claude, and DeepSeek, each offering unique capabilities in natural language understanding and programming assistance. Despite their increasing adoption, relatively few studies have performed a comprehensive comparison of these models using a common educational benchmark. Existing research generally evaluates only one or two models or focuses on software engineering applications rather than classroom learning environments.

Another limitation observed in previous research is the absence of a standardized evaluation framework specifically designed for programming education. Many studies assess only code correctness or benchmark scores while ignoring factors such as debugging support, explanation clarity, code optimization, response time, and usability for novice programmers. As a result, educators often lack sufficient evidence to determine which AI model is most appropriate for teaching and learning purposes.

To address these research gaps, the present study performs a comparative analysis of four widely used Large Language Models ChatGPT, Gemini, Claude, and DeepSeek using a diverse benchmark of programming problems from multiple programming languages and computer science subjects. Unlike previous studies, this research evaluates the selected models using both technical and educational parameters, including code correctness, explanation quality, debugging capability, code optimization, readability, beginner friendliness, error handling, and response time. The proposed evaluation approach aims to provide practical guidance for educators, students, and researchers while contributing to the growing body of knowledge on AI-assisted programming education.

3. RESEARCH GAP

Based on the literature review, the following research gaps have been identified:

- Most existing studies evaluate only the **code generation capability** of Large Language Models, while educational parameters such as explanation quality and learning support receive limited attention.
- Comparative evaluations involving **ChatGPT, Gemini, Claude, and DeepSeek** under the same experimental conditions are relatively limited.
- Previous research generally focuses on software development tasks rather than programming education in academic settings.
- There is no widely accepted evaluation framework that combines both **technical performance** and **educational effectiveness**.
- Limited research has examined the suitability of different LLMs for **beginner programmers** across multiple programming languages.

Therefore, this study aims to fill these gaps by proposing a comprehensive evaluation framework for comparing four leading Large Language Models in programming education using standardized programming tasks and multiple educational and technical performance indicators.

4. METHODOLOGY

This study adopts an empirical comparative research methodology to evaluate the performance of four widely used Large Language Models (LLMs) **ChatGPT, Gemini, Claude, and DeepSeek** in the context of programming education. The proposed methodology is designed to assess both the technical capabilities and educational effectiveness of these AI-powered programming assistants using a standardized evaluation process. Figure 1 illustrates the overall workflow of the proposed research methodology.

The evaluation process begins with the selection of programming problems covering multiple programming languages and core computer science subjects. The selected questions are then submitted to each LLM using identical prompts to ensure fairness and consistency. The responses generated by each model are collected and evaluated using predefined performance parameters. Finally, the results are analyzed and compared to identify the strengths and limitations of each model.

4.1 Selection of Large Language Models

Four widely used Large Language Models were selected for this study based on their popularity, accessibility, and capability to assist in programming-related tasks.

Table 1: Selected Large Language Models

S. No.	Model	Developer	Primary Application
1	ChatGPT	OpenAI	Code generation, debugging, explanation
2	Gemini	Google	Programming assistance and reasoning
3	Claude	Anthropic	Code analysis and natural language explanation
4	DeepSeek	DeepSeek AI	Programming and algorithmic problem solving

These models were selected because they represent some of the most advanced publicly available AI systems used for programming assistance and educational support.

4.2 Benchmark Dataset

A benchmark dataset consisting of programming problems from different programming languages and computer science subjects is prepared to evaluate the selected LLMs. The dataset includes beginner, intermediate, and advanced-level programming tasks to ensure a comprehensive assessment.

The benchmark covers the following domains:

- C Programming
- C++
- Java
- Python
- SQL
- HTML
- CSS
- JavaScript
- PHP
- Data Structures
- Algorithms

Each model receives the same programming questions without modification to maintain consistency during evaluation.

4.3 Experimental Procedure

The comparative evaluation follows the following steps:

1. Prepare a benchmark consisting of programming questions from multiple domains.
2. Submit the same prompt to each Large Language Model.
3. Record the responses generated by each model.
4. Execute the generated code where applicable to verify correctness.
5. Evaluate each response using predefined evaluation parameters.
6. Compare the performance of all models using quantitative and qualitative analysis.
7. Summarize the findings to identify the most suitable LLM for programming education.

The entire evaluation process is conducted under identical testing conditions to ensure fairness and reproducibility.

4.4 Evaluation Parameters

The performance of each Large Language Model is evaluated using eight parameters that reflect both programming capability and educational usefulness.

Table 2: Evaluation Parameters

Parameter	Description
Code Correctness	Accuracy of the generated program
Explanation Quality	Clarity and completeness of concept explanation
Debugging Capability	Ability to identify and fix programming errors
Code Optimization	Efficiency and quality of generated code
Readability	Simplicity and maintainability of generated code
Beginner Friendliness	Ease of understanding for novice programmers
Error Handling	Ability to address invalid inputs and exceptions
Response Time	Time required to generate the response

These parameters provide a balanced evaluation of both the technical performance and educational effectiveness of the selected models.

4.5 Scoring Method

Each evaluation parameter is assessed using a **five-point rating scale**, where a score of **1** represents poor performance and a score of **5** represents excellent performance.

Table 3: Scoring Scale

Score	Performance Level
1	Poor
2	Fair
3	Good
4	Very Good
5	Excellent

The total score for each LLM is calculated by combining the scores obtained across all evaluation parameters. The comparative analysis is then performed using these scores to determine the overall effectiveness of each model in programming education.

4.6 Data Analysis

The collected data are analyzed using descriptive statistical methods. The average score obtained by each Large Language Model is calculated for every evaluation parameter. Comparative tables, bar charts, and radar charts are used to visualize the performance differences among the selected models. The analysis focuses on identifying the strengths and limitations of each LLM and determining its suitability for programming education.

5. RESULTS AND DISCUSSION

This section presents the comparative performance of ChatGPT 5.5, Gemini 3.1 Pro, Claude Sonnet 5, and DeepSeek DeepThink based on the predefined evaluation criteria. Each model was evaluated using the same benchmark programming questions covering C, C++, Java, Python, SQL, HTML, CSS, JavaScript, PHP, Data Structures, and Algorithms. The generated responses were assessed using a five-point scoring scale considering code correctness, explanation quality, debugging capability, code optimization, readability, beginner friendliness, error handling, response time, and context length. The comparative scores obtained during the experimental evaluation are summarized in Table 4.

The experimental results demonstrate noticeable differences in the performance of the selected Large Language Models across various evaluation parameters. Although all four models successfully generated solutions for most programming tasks, their effectiveness varied depending on the complexity of the problem and the educational objective. Claude Sonnet 5 achieved the highest overall technical performance, while ChatGPT 5.5

demonstrated superior educational support through detailed explanations and beginner-friendly responses. Gemini 3.1 Pro excelled in response speed and handling longer conversations, whereas DeepSeek DeepThink showed competitive performance in debugging and algorithm-based programming tasks.

Table 4: Overall Performance of Large Language Models

Evaluation Parameter	ChatGPT (5.5)	Gemini (3.1 Pro)	Claude (Sonnet 5)	DeepSeek (DeepThink)
Code Correctness	4	3	4.5	3
Explanation Quality	4.5	3.5	4	3
Debugging Capability	3.5	3	4.5	4
Code Optimization	4	3.5	4.5	4
Readability	4	3.5	4.5	3.5
Beginner Friendliness	4.5	4	4	3.5
Error Handling	4	3.5	4	4
Response Time	4	4.5	4	3.5
Context Length	3	4.5	4	4

5.1 Code Correctness

Code correctness is one of the most important criteria for evaluating AI-assisted programming tools because students rely on generated programs to understand concepts and complete programming assignments. During the evaluation, each model was tested using programming problems of varying complexity, and the generated code was executed to verify its correctness and logical accuracy. Among the evaluated models, **Claude Sonnet 5** obtained the highest score (**4.5/5**) for code correctness. The generated programs were syntactically correct, logically consistent, and required minimal modifications before execution. Claude consistently produced accurate implementations for object-oriented programming concepts, recursion, searching and sorting algorithms, and SQL queries.

ChatGPT 5.5 received a score of **4.0/5**, demonstrating strong code generation capabilities across multiple programming languages. The generated code was generally correct and well-structured, although a few complex algorithmic problems required minor refinements to improve efficiency or handle edge cases.

Both **Gemini 3.1 Pro** and **DeepSeek DeepThink** obtained **3.0/5** in code correctness. Although both models generated executable programs for most beginner and intermediate-level questions, they occasionally produced incomplete logic or incorrect implementations for advanced programming problems involving recursion, dynamic programming, and complex data structures.

These observations indicate that Claude Sonnet 5 is the most reliable model for generating accurate programming solutions, followed closely by ChatGPT 5.5.

5.2 Explanation Quality

Explanation quality evaluates how effectively a model helps students understand programming concepts rather than simply providing the final solution. A high-quality explanation should clearly describe the programming logic, algorithm, syntax, and execution flow while remaining easy for beginners to understand.

According to the evaluation results, **ChatGPT 5.5** achieved the highest score (**4.5/5**) in explanation quality. The model consistently provided step-by-step explanations, described the purpose of individual code statements, explained algorithmic logic, and frequently suggested alternative approaches. The responses were well organized and particularly useful for beginner programmers.

Claude Sonnet 5 obtained a score of **4.0/5** by providing technically accurate explanations with good logical organization. Although the explanations were precise and detailed, they were generally more concise and targeted toward users with some programming background.

Gemini 3.1 Pro achieved **3.5/5**, offering concise explanations that were sufficient for understanding the solution but occasionally lacking deeper conceptual discussion. **DeepSeek DeepThink** received **3.0/5**, as its responses focused primarily on code generation and debugging rather than extensive conceptual explanations.

These results suggest that ChatGPT 5.5 is the most suitable model for educational environments where conceptual understanding and guided learning are essential.

5.3 Debugging Capability

Debugging capability refers to the ability of a Large Language Model to identify syntax errors, logical mistakes, runtime exceptions, and inefficient programming practices while suggesting appropriate corrections.

Among the evaluated models, **Claude Sonnet 5** demonstrated the strongest debugging capability with a score of **4.5/5**. It successfully identified programming errors, explained their underlying causes, and proposed

accurate corrective solutions. Claude was particularly effective in resolving logical errors and optimizing inefficient implementations.

DeepSeek DeepThink achieved **4.0/5**, showing strong performance in identifying logical issues and debugging algorithm-based problems. The model frequently suggested practical improvements and handled debugging tasks effectively, especially in C++, Java, and Python.

ChatGPT 5.5 obtained **3.5/5**, correctly identifying most syntax and runtime errors while providing useful explanations for the suggested corrections. However, in certain complex debugging scenarios, additional prompts were occasionally required to achieve the desired solution. **Gemini 3.1 Pro** received **3.0/5**. While it successfully detected common syntax errors, its debugging explanations were comparatively brief and less detailed than those of Claude and ChatGPT.

The findings indicate that Claude Sonnet 5 provides the most comprehensive debugging assistance among the evaluated models.

5.4 Code Optimization

Code optimization measures the ability of a model to generate efficient, maintainable, and well-structured programs while following good programming practices.

The evaluation results show that **Claude Sonnet 5** achieved the highest score (**4.5/5**) in code optimization. The generated solutions consistently used efficient algorithms, appropriate data structures, and clean programming practices. Redundant computations were minimized, resulting in optimized and maintainable code.
Both **ChatGPT 5.5** and **DeepSeek DeepThink** received **4.0/5**. Their generated programs generally followed accepted programming standards and demonstrated good readability. However, some responses included additional statements or alternative implementations that could be simplified further.

Gemini 3.1 Pro obtained **3.5/5**. Although its solutions were functionally correct, certain implementations relied on less efficient approaches when compared with the optimized solutions generated by Claude.

These observations suggest that Claude Sonnet 5 produces the most efficient programming solutions among the evaluated models.

5.5 Overall Comparative Analysis

The overall experimental evaluation demonstrates that each Large Language Model possesses distinct strengths that make it suitable for different programming education scenarios. No single model achieved the highest score across every evaluation parameter, indicating that the choice of an AI programming assistant should depend on the specific educational objective.

Claude Sonnet 5 emerged as the strongest performer in terms of overall technical capability. It achieved the highest scores in **code correctness (4.5), debugging capability (4.5), code optimization (4.5), and readability (4.5)**, making it particularly suitable for solving complex programming problems and assisting users with software development tasks.

ChatGPT 5.5 demonstrated excellent educational performance by achieving the highest scores in **explanation quality (4.5)** and **beginner friendliness (4.5)**. Its ability to explain programming concepts in a structured and easy-to-understand manner makes it highly effective for teaching and self-learning.

Gemini 3.1 Pro performed consistently across all evaluation parameters and achieved the highest score in **response time (4.5)** and **context length (4.5)**. These characteristics make it suitable for extended interactive programming sessions where quick responses and long conversations are required.

DeepSeek DeepThink showed competitive performance in **debugging capability (4.0)**, **code optimization (4.0)**, **error handling (4.0)**, and **context length (4.0)**. Its strong algorithmic reasoning suggests that it can serve as an effective alternative for users working on computational and logic-intensive programming tasks.

Overall, the results demonstrate that **Claude Sonnet 5** is the most technically capable model for programming tasks, whereas **ChatGPT 5.5** provides the most effective educational support for beginner programmers. **Gemini 3.1 Pro** excels in responsiveness and handling longer conversations, while **DeepSeek DeepThink** offers reliable debugging and algorithmic problem-solving capabilities. Therefore, educators and students should select an appropriate Large Language Model based on their specific learning objectives, programming requirements, and desired educational outcomes.

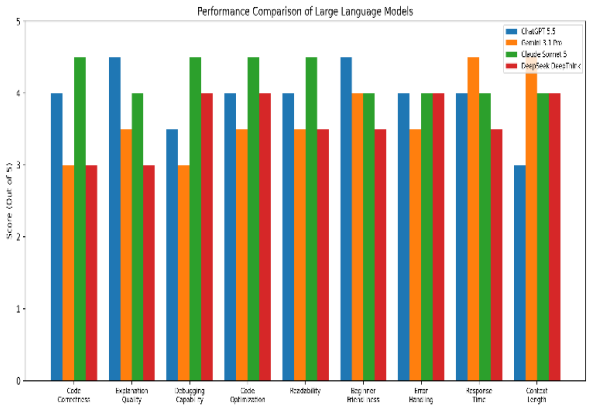

Figure 1: Performance comparison of ChatGPT 5.5, Gemini 3.1 Pro, Claude Sonnet 5, and DeepSeek DeepThink across nine evaluation parameters.

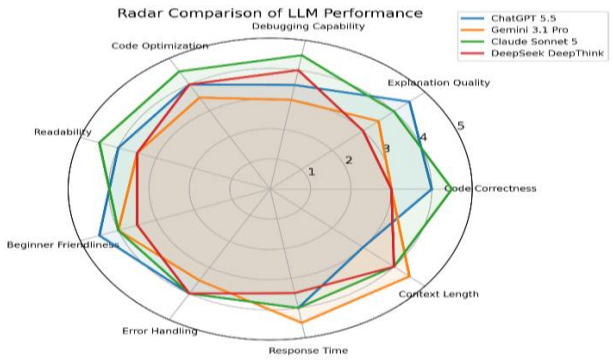

Figure 2: Radar chart illustrating the comparative strengths and weaknesses of the evaluated Large Language Models in programming education.

6. CONCLUSIONS

The rapid advancement of Large Language Models (LLMs) has significantly transformed programming education by providing intelligent support for code generation, debugging, concept explanation, and problem-solving. This study presented an empirical comparative analysis of four freely accessible Large Language Models **ChatGPT 5.5, Gemini 3.1 Pro, Claude Sonnet 5, and DeepSeek DeepThink** to evaluate their effectiveness in programming education. The evaluation was conducted using a benchmark comprising programming problems from multiple programming languages and computer science domains, including C, C++, Java, Python, SQL, HTML, CSS, JavaScript, PHP, Data Structures, and Algorithms. Each model was assessed using nine evaluation parameters: code correctness, explanation quality, debugging capability, code optimization, readability, beginner friendliness, error handling, response time, and context length.

The experimental evaluation revealed that **Claude Sonnet 5** achieved the highest overall technical performance among the evaluated models. It consistently generated accurate and optimized code, demonstrated excellent debugging capabilities, and produced highly readable programming solutions. These characteristics make Claude Sonnet 5 particularly suitable for solving complex programming problems and assisting users in software development tasks.

ChatGPT 5.5 demonstrated outstanding educational capabilities by providing detailed explanations, beginner-friendly responses, and logically structured programming solutions. Its ability to explain programming concepts in simple language makes it an effective learning companion for students, especially beginners who require conceptual guidance in addition to code generation.

Gemini 3.1 Pro performed well in terms of response speed and context handling, allowing it to process longer interactions efficiently while maintaining satisfactory programming performance. **DeepSeek DeepThink** showed strong debugging capabilities and competitive performance in algorithmic problem-solving, making it a useful tool for users working on logic-intensive programming tasks.

The results indicate that **no single Large Language Model is universally superior across all evaluation criteria**. Each model possesses unique strengths that make it suitable for different educational and programming scenarios. Students seeking conceptual understanding and detailed explanations may benefit more from ChatGPT 5.5, whereas users requiring highly accurate code generation and advanced debugging support may find Claude Sonnet 5 more effective. Similarly, Gemini 3.1 Pro is appropriate for extended interactive sessions due to its response speed and context management, while DeepSeek DeepThink serves as a capable alternative for algorithm-oriented programming exercises.

This study contributes to the growing field of AI-assisted programming education by providing a structured and reproducible evaluation framework for comparing Large Language Models using both technical and educational parameters. The findings can assist students, educators, and academic institutions in selecting appropriate AI-based programming assistants according to their learning objectives and teaching requirements.

Although the study provides valuable insights, it has certain limitations. The evaluation was conducted using the **free versions** of the selected Large Language Models, and their performance may differ from premium or enterprise editions. In addition, the benchmark dataset, while covering multiple programming domains, cannot represent every possible programming scenario. Future

research may expand this work by including additional LLMs, larger benchmark datasets, real classroom-based evaluations involving students, and advanced performance metrics such as code security, computational efficiency, explainability, and long-term learning outcomes. As Large Language Models continue to evolve, continuous evaluation will remain essential to ensure their effective, responsible, and pedagogically meaningful integration into programming education.

7. REFERENCES

1) N. Kiesler and D. Schiffner, "Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments," arXiv:2308.08572, 2023.

2) C. N. Anagnostopoulos, "ChatGPT Impacts in Programming Education: A Recent Literature Overview That Debates ChatGPT Responses," F1000Research, vol. 12, p. 1393, 2023.

3) M. B. Garcia, "Teaching and Learning Computer Programming Using ChatGPT: A Rapid Review of Literature Amid the Rise of Generative AI Technologies," SSRN, 2025.

4) Suzuki Haruto, L. C. Nnadi, and Y. Watanobe, "Systematic Review of Large Language Model Applications in Programming Education," New Trends in Intelligent Software Methodologies, Tools and Techniques, 2026.

5) Talita de Paula Cypriano de Souza, S. Mehta, M. A. Uema, L. B. de Paula, and S. Isotani, "Can AI be a Teaching Partner? Evaluating ChatGPT, Gemini, and DeepSeek across Three Teaching Strategies," SSRN, 2026.

6) M. M. Rahman, A. I. Shiplu, M. F. I. Amin, Y. Watanobe, and L. Peng, "Large Language Models for Education and Research: An Empirical and User Survey-based Analysis," arXiv, 2025.

7) A. Cabezas-Clavijo and P. Sidorenko-Bautista, "Assessing the Performance of AI Chatbots in Bibliographic Reference Retrieval: Grok and DeepSeek Outperform ChatGPT, but None Are Fully Accurate," arXiv, 2025.

8) "Comparative Analysis Based on DeepSeek, ChatGPT, and Google Gemini: Features, Techniques, Performance, Future Prospects," Systems and Soft Computing, vol. 7, 2025.

9) "A Comparative Study of Large Language Models in Programming Education: Accuracy, Efficiency, and Feedback in Student Assignment Grading," Applied Sciences, vol. 15, no. 18, 2025.

10) "A Systematic Comparison of Large Language Models for Automated Assignment Assessment in Programming Education: Exploring the Importance of Architecture and Vendor," Computers and Education Open, 2026.

11) "Hands-on Analysis of Using Large Language Models for the Auto Evaluation of Programming Assignments," Information and Software Technology, 2024.

12) OpenAI, "GPT-5 System Card," OpenAI, 2025.

13) Google DeepMind, "Gemini Technical Report," Google DeepMind, 2024.

14) Anthropic, "Claude 3.7 Sonnet System Card" (Feb/March 2025) Technical Report," Anthropic, 2025.

15) DeepSeek AI, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," DeepSeek AI, 2025.

BIOGRAPHIES

Mr. Manish Kumar Hooda is working as an Assistant Professor in the CSE Department at Samalkha Group of Institutions, Delhi with total 6+ years of teaching experience.

Mr. Yatender Kumar is serving as the **Academic Coordinator** at **Samalkha Group of Institutions (SGI), New Delhi**. He has extensive **20+** years' experience in teaching undergraduate and postgraduate students in the field of Computer Science.

Mr. Abhishek Kumar Maheshwari is a Research Scholar (CSE) in the Sanskriti University, Mathura.