Comparing Code Generation Capabilities of ChatGPT-4o and DeepSeek V3 in Solving TypeScript Programming Problems

Filip Stamenković
University of Belgrade
Faculty of Organizational Sciences
Belgrade, Serbia *filip.stamenkovic@fon.bg.ac.rs*

Jelica Stanojević
University of Belgrade
Faculty of Organizational Sciences
Belgrade, Serbia *jelica.stanojevic@fon.bg.ac.rs*

Dejan Simić
University of Belgrade
Faculty of Organizational Sciences
Belgrade, Serbia *dejan.simic@fon.bg.ac.rs*

Abstract

The rapid development of large language models significantly impacts software development, particularly in code generation. This paper focuses on the analysis of the performance and features of ChatGPT and DeepSeek chatbots, based on their GPT-4o and V3 models, respectively, with an emphasis on code generation. Particular attention is given to the architecture of the models, multimodality, open-source status, and token limits. Through experimental evaluation of 60 TypeScript LeetCode problems across different difficulty levels, we evaluated accuracy, debugging ability, and the number of attempts needed for correct solutions. The results show that DeepSeek achieved an accuracy of 68.3%, while ChatGPT achieved 61.7%. The paper highlights the advantages of DeepSeek as an open-source option and points to the potential to improve generated code, contributing to the understanding of the application of large language models in programming.

Keywords: ChatGPT-4o, DeepSeek V3, large language model, chatbot, programming.

1. Introduction

Advances in chatbots and large language models (LLMs) have played a major role in advancing artificial intelligence (AI), transitioning from simple rule-based systems to modern transformer-based models [26] that generate human-like text using neural networks [6]. Since the introduction of GPT-1 in 2018 [21], models such as GPT-3 have scaled up to billions of parameters, improving text understanding and generation [3]. These advances, driven by unsupervised learning [13], have enabled LLMs to adapt to various tasks, powering virtual assistants, customer support, and conversational AI like ChatGPT [27]. Their versatility comes from training on diverse large datasets, making them applicable across industries [13]. In software development, LLMs have the potential to automate coding, assist in debugging, and accelerate workflows [12].

This paper presents an experimental comparison of two advanced chatbots, ChatGPT, which is based on the GPT-4o model (hereafter referred to as ChatGPT-4o) and DeepSeek, which utilizes the V3 model, to analyze their code generation capabilities, performance, and basic features. The comparison is made by solving the same TypeScript programming problems from the LeetCode platform. The LeetCode platform is useful for assessing the ability to solve programming problems, and large language models like

ChatGPT can solve most LeetCode problems [2]. The focus of the research is on the accuracy of the generated solutions, the number of attempts needed for a correct solution, and the ability of the models to correct their own mistakes.

2. Related Work

Previous research has introduced benchmark frameworks to evaluate code generation capabilities of LLMs. The HumanEval benchmark [4] evaluates the performance of LLMs in code generation tasks. The study [4] showed that Codex solved 28.8% of Python problems with single sampling, and up to 70.2% with 100 samples, demonstrating how multiple sampling increases accuracy. Similarly, the MBPP benchmark and dataset [1], with 974 basic Python problems, shows performance scaling with model parameters, and improvement with few-shot prompting, highlighting how prompt design can impact code quality.

Beyond controlled benchmarks, recent work has explored how AI coding assistants like GitHub Copilot are being adopted in practice. A study on the use of GitHub Copilot [25] indicates that while the tool helps developers by providing an initial starting point, the users often struggle to understand or modify the generated code. Moreover, the study found no substantial improvement in overall task completion time or success rate in practical settings.

While most existing research has been conducted on ChatGPT [2], [14], [24], which has brought revolutionary changes in the field of large language models, particularly in the subfield of code generation using artificial intelligence [15], comparatively less attention has been given to DeepSeek, resulting in a research gap.

Research conducted in [22] shows that ChatGPT, based on the GPT-4 model, solved a set of programming problems from the LeetCode platform with a success rate of 71.875%. The study also demonstrated a linear correlation between the acceptance rate of problems and ChatGPT's ability to solve them correctly, meaning that a high acceptance rate of a particular problem on the LeetCode platform predicts that ChatGPT will solve it successfully. It was concluded that ChatGPT has difficulties in debugging previously generated code, as well as in learning from the mistakes that were provided to it as feedback. Even after multiple attempts to correct the code based on the feedback, there was a decline in performance.

The paper [23] compares the performances of the ChatGPT model version o3-mini and the DeepSeek model version R1 in solving programming tasks from the Codeforces platform, using the C++ programming language. Both models demonstrated a high accuracy in solving easy tasks. When solving medium-difficulty problems, ChatGPT achieved a higher percentage of successfully solved problems, reaching 54.5%, while DeepSeek only solved 18.1% of the tasks given. Regarding hard tasks, ChatGPT solved 1 out of 9, while DeepSeek was not able to solve any of the tasks. Both models are seen as good at solving easier tasks, while greater differences in performance are noted when solving more complex tasks.

Additionally, the paper [16] evaluates the performance of the ChatGPT model version o1 and the DeepSeek model version R1 by using different metrics like readability, efficiency, and correctness. Comparing these models involved analyzing the performance and use of computational resources of generated codes as solutions to the programming problems in the Python programming language. The results show a slight advantage in DeepSeek in terms of code correctness, and the requirement of less attempts to generate codes that are acceptable as solutions, especially for problems involving algorithms. ChatGPT is more successful than its competitor when it comes to code quality, performance, and conciseness. The paper emphasizes the importance of continuous research in AI-assisted software development and provides useful insight to programmers in selecting the most suitable model as an assistant in software development by highlighting the advantages and disadvantages of the models.

Although there are papers that individually analyze these models, as well as those that compare earlier versions of ChatGPT and DeepSeek, a comparison of the ChatGPT-4o

and DeepSeek V3 models, in the context of solving TypeScript programming problems from the LeetCode platform, has not yet been conducted. This paper contributes to a better understanding of their capabilities and performance in practical tasks.

3. Overview of the Evaluated Chatbots

ChatGPT and DeepSeek are advanced AI chatbots that rely on large language models trained using the transformer architecture to generate human-like responses and assist users with various tasks [11], [26].

GPT-4o, developed by OpenAI, is a multimodal model capable of processing text, audio, images, and video inputs [19]. Its training data includes textual and audio data available up to October 2023 [19]. It is available to users through desktop, mobile, and web applications, and can be integrated into other software via the OpenAI API [18], [20]. The GPT-4o model supports an input context window of up to 128,000 tokens and can generate up to 16,384 output tokens per request [20]. Unlike the original transformer, which uses an encoder-decoder structure, the GPT models rely solely on the decoder component for unsupervised training on large text datasets [13], [21]. ChatGPT is available to its users through web, desktop, and mobile applications, while GPT models can also be used via the OpenAI API (Application Programming Interface) platform, allowing integration with other software solutions [18], [20].

DeepSeek V3, developed by DeepSeek, is an open-source model and a direct competitor to closed-source models [7] like GPT-4o. Using the DeepSeek chatbot, it is concluded that DeepSeek V3 is not a fully multimodal model. While users can submit images and documents to DeepSeek, the model does not directly analyze images but instead extracts embedded text, limiting its multimodal capability compared to ChatGPT. Like GPT models, DeepSeek V3 is based on the transformer architecture [26], and includes enhancements such as Multi-head Latent Attention (MLA) and a Mixture-of-Experts (MoE) mechanism [7]. These upgrades allow the model to activate specific experts for different tasks, improving efficiency [5]. The DeepSeek V3 model supports an input context window of up to 128,000 tokens and a maximum output of 8,000 tokens per request [7], [9]. Its knowledge cut-off date is not publicly available. The DeepSeek chatbot is available through web and mobile applications, and its models can also be accessed via the API platform [8, 9].

Table 1. provides a side-by-side comparison of the main technical characteristics and design choices of ChatGPT-4o and DeepSeek V3.

Table 1. Features of the ChatGPT-4o and DeepSeek V3 models

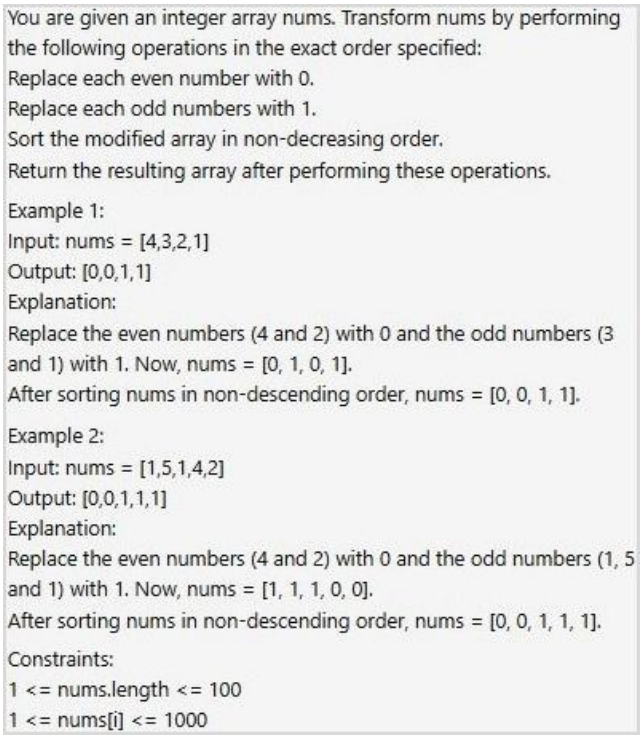
Feature	**ChatGPT-4o**	**DeepSeek V3**
Architecture	Transformer	Transformer with MLA and Mixture-of-Experts
Multimodality	Yes	No
Knowledge cut-off date	October 2023	Unknown
Open-source	No	Yes
Platforms	Desktop, mobile and web applications, API platform	Mobile and web applications, API platform
Input Context Window	128,000 tokens	128,000 tokens
Maximum Output Tokens	16,384 tokens	8,000 tokens

4. Methodology

This section of the paper presents an overview of the methodology used to compare the performance of the ChatGPT and DeepSeek chatbots and their GPT-4o and V3 models in solving problems from the LeetCode platform. Both models were used with default system parameters and temperature, which corresponds to a temperature of 1. Prompts were kept simple and consistent across tasks, following a format where the task description was passed as a plain string. The following are the key elements of the methodology.

4.1. Problems

To compare the performance of the models used by chatbots, 20 problems were selected from each of the three difficulty categories available on the LeetCode platform: easy, medium, and hard. Problems were randomly selected from the LeetCode platform using its "Pick one" option and built-in category filters. Testing across these three categories allowed for an assessment of how the models perform when solving problems of varying levels of complexity. The same problems were sent to the ChatGPT and DeepSeek chatbots and the implementation language was set to TypeScript. Each problem was sent with constraints, input and output examples, and in some cases, if provided by the LeetCode platform, an explanation. While the focus on accuracy, attempts, and error correction is similar to the work reported in [16], the key differences are the programming language (TypeScript), task set, platform (LeetCode), and models (ChatGPT-4o and DeepSeek V3). An example prompt of one of the easy problems that was sent to the ChatGPT and DeepSeek chatbots is shown in Fig. 1.


You are given an integer array nums. Transform nums by performing the following operations in the exact order specified:
Replace each even number with 0.
Replace each odd numbers with 1.
Sort the modified array in non-decreasing order.
Return the resulting array after performing these operations.

Example 1:
Input: nums = [4,3,2,1]
Output: [0,0,1,1]
Explanation:
Replace the even numbers (4 and 2) with 0 and the odd numbers (3 and 1) with 1. Now, nums = [0, 1, 0, 1].
After sorting nums in non-descending order, nums = [0, 0, 1, 1].

Example 2:
Input: nums = [1,5,1,4,2]
Output: [0,0,1,1,1]
Explanation:
Replace the even numbers (4 and 2) with 0 and the odd numbers (1, 5 and 1) with 1. Now, nums = [1, 1, 1, 0, 0].
After sorting nums in non-descending order, nums = [0, 0, 1, 1, 1].

Constraints:
1 <= nums.length <= 100
1 <= nums[i] <= 1000

Fig. 1. Example prompt of one of the easy problems sent to the chatbots

4.2. Initial Prompt

Before the problems were sent to the chatbots, they received an introductory prompt to ensure they properly understood the task requirements. This prompt was the same for both chatbots to ensure they had the same starting conditions. The following prompts were specific problems that were presented to both chatbots, one by one.

The prompt that was sent was as follows: *"You are a coding assistant. You will be provided with programming challenges, and your task is to solve them in TypeScript programming language. Please solve the following problem to the best of your ability."*

4.3. Experimental Procedure

The experimental procedure, shown in Fig. 2, involved sending the generated code to the LeetCode platform for testing. The steps of the procedure were as follows:

- Sending the initial prompt;
- Submission of code: after the chatbots received the problem and generated their

solutions, the corresponding code was submitted to the LeetCode platform;
- Code evaluation: the platform issued a verdict for each solution, classifying it as accepted or not;
- Review: if the solution did not pass the initial try, the error received was forwarded to the chatbot to make additional corrections. Each chatbot was allowed a maximum of three attempts to correct the code. Responses were evaluated based on the final verdict.

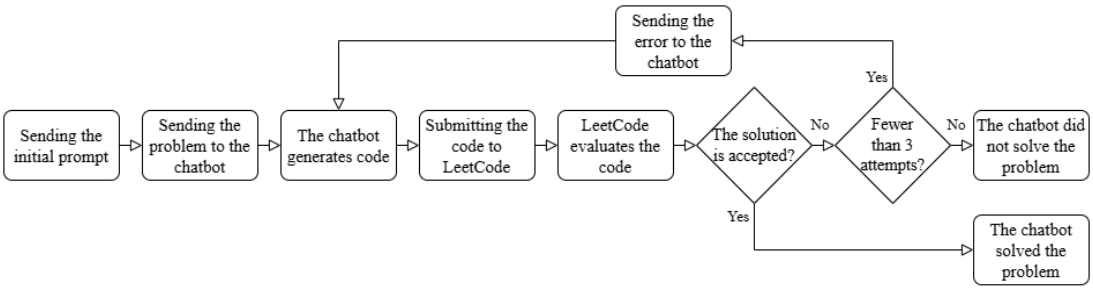


Fig. 2. Experimental procedure flowchart

4.4. Comparison of the Chatbots

After obtaining the results from the LeetCode platform, the chatbots' performance was compared, including their success in solving programming problems across different difficulties and topics, memory usage, and runtime of the generated code.

5. Results

This section presents the experimental results of testing ChatGPT and DeepSeek chatbots, based on their GPT-4o and V3 models, on 60 TypeScript programming problems from the LeetCode platform. The tasks were divided equally into three categories: easy, medium, and hard, with 20 tasks in each.

DeepSeek solved 41 out of 60 tasks (68.3%), while ChatGPT solved 37 tasks (61.7%). Fig. 3. shows the number of accepted solutions for each difficulty level. For easy problems, DeepSeek solved all 20, while ChatGPT solved 18. In the medium-difficulty group, DeepSeek solved 12 and ChatGPT 11 problems. Regarding hard problems, DeepSeek solved 9 and ChatGPT 8.

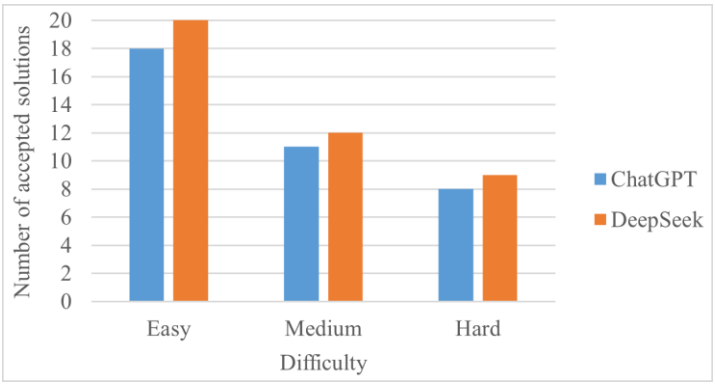


Fig. 3. The number of accepted solutions by difficulty categories for the ChatGPT-4o and DeepSeek V3 models

To determine whether this difference in overall performance is statistically significant, McNemar's test was applied to the paired outcomes. Of the 8 disagreements, DeepSeek was correct in 6 cases where ChatGPT was not, and ChatGPT was correct in 2 cases where DeepSeek was not, resulting in a p-value of 0.289, which is not statistically significant at the 0.05 level.

For each task, the LeetCode platform provided a verdict of Accepted or an error type. The main error types recorded were Wrong Answer, Time Limit Exceeded, and Compile Error. Table 2. shows the total number of errors by type for each chatbot. ChatGPT generated 19 Wrong Answers, 3 Time Limit Exceeded errors, and 1 Compile Error. DeepSeek generated 12 Wrong Answers and 7 Time Limit Exceeded errors, but no Compile Errors.

Table 2. Number of errors by type

Error type	ChatGPT-4o	DeepSeek V3
Wrong Answer	19	12
Time Limit Exceeded	3	7
Compile Error	1	0

Each chatbot was allowed up to three attempts per task. For easy tasks, most correct solutions were produced on the first attempt. For medium and hard tasks, both chatbots required multiple attempts more often. When an error occurred, LeetCode's feedback was used to resubmit the corrected code within the allowed attempts.

Across all tasks, runtimes and memory usage generally increased with problem complexity. For easy tasks, runtimes were a few milliseconds (ms) and memory usage ranged from 55 to 60 megabytes (MB). For medium tasks, runtimes in some cases reached tens of milliseconds, with memory usage exceeding 60 MB. Similar values were observed for hard tasks.

Regarding the tasks that were not solved, most had low acceptance rates on the LeetCode platform. Among the 19 problems that DeepSeek did not solve, 17 had an acceptance rate below 36%, with the remaining two at 43.1% and 69.2%. Regarding ChatGPT, 22 out of 23 unsolved problems had an acceptance rate below 46%.

Among the programming topics covered, arrays, strings, dynamic programming, and math problems were the most common. Tasks that required multiple concepts were more frequent in the medium and hard categories.

6. Discussion and Limitations

Both chatbots and their models performed well on easy tasks, providing quick and correct solutions. However, accuracy dropped for medium and hard tasks, with frequent issues such as type incompatibility and incorrect code generation. DeepSeek encountered more Time Limit Exceeded errors, while ChatGPT faced more compile/runtime errors, which were generally corrected within three attempts. The most common error was Wrong Answer, especially with dynamic programming and multi-topic problems, indicating challenges with edge cases and complex algorithms. DeepSeek often provided more detailed comments, making it useful for learning programming. Unsolved tasks were more complex, requiring higher computational power and longer runtimes. Both models excelled with arrays, strings, and simple math, but struggled with dynamic programming and combinatorics, emphasizing the need for improved prompt engineering for more complex tasks.

Limitations of the study include the small sample size, focus on TypeScript, and lack of in-depth human assessment of code quality or efficiency. The three-attempt limit was chosen to simulate real AI coding assistant interactions, but may not reflect the models' full potential for complex tasks.

Compared to [22], which reported a 71.875% success rate for GPT-4 on LeetCode problems, ChatGPT-4o achieved a slightly lower rate of 61.7% in our TypeScript tasks. This difference may be attributed to the programming language used, the smaller sample size, and the limited number of retries.

Finally, while an earlier comparison [23] found DeepSeek to underperform ChatGPT on medium and hard problems in Python and C++, our results show DeepSeek V3 slightly outperforming ChatGPT-4o across all difficulty levels. However, McNemar's test suggests this difference is not statistically significant, highlighting the need for larger samples and more diverse problem sets. Despite this, our findings align with [16], which observed DeepSeek's slight advantage in code correctness and error correction.

7. Future Research Directions

Considering the limitations of this research, one of the key directions for future research is to expand the problem set of the experiment with the inclusion of a larger number of problems, as well as topics covered by them. In addition to expanding the set of problems, testing the models in different programming languages could contribute to a better understanding of their adaptability. Additionally, the inclusion of other chatbots that utilize large language models would allow for a more comprehensive comparison of the performance of ChatGPT and DeepSeek.

Another important research area is the optimization of generated code, with a focus on efficiency in terms of execution time and memory consumption. In this paper, particularly with DeepSeek, difficulties have been observed in solving tasks that require high time efficiency. Improvements in the ability of the models to recognize and correct their errors would increase their use in software development.

More research should be done on the importance of prompt engineering to increase the accuracy and efficiency of the answers given by LLMs. The formulation of prompts can greatly affect the model's output, particularly in more complex tasks [10].

8. Conclusion

This paper compares the capabilities of ChatGPT and DeepSeek chatbots, based on their GPT-4o and V3 models, in solving TypeScript programming problems. The chatbots were asked to solve 60 TypeScript programming problems of various difficulties from the LeetCode platform. The results show that DeepSeek achieved a success rate of 68.3%, and ChatGPT achieved a rate of 61.7%.

Both chatbots performed reliably on easy tasks, but struggled with medium and hard problems, which often cover multiple topics. Their models efficiently managed memory and runtimes for simpler tasks, while having an increase in both parameters when solving more complex problems.

While both chatbots demonstrated moderate success on the selected tasks, this study serves as an initial step in understanding their potential role in developer workflows. Broader validation of real-world tasks is needed to determine their practical applicability.

DeepSeek is an attractive option for users who want a cost-effective or an open-source solution while maintaining a high performance [17]. DeepSeek's ability to generate code with detailed explanations and comments is particularly beneficial when learning to code. The main disadvantage of DeepSeek is that it does not use a multimodal model like its competitor ChatGPT does. The multimodality of its model makes ChatGPT suitable for various uses alongside programming [11].

In conclusion, while the chatbots are not yet fully capable of solving difficult programming problems, they represent a significant step forward in the development of programming tools. Future research should focus on improving the quality of generated code, better understanding of the impact of prompt engineering, and improving the ability of models to correct themselves.

References

1. Austin, J. et al.: Program Synthesis with Large Language Models, http://arxiv.org/abs/2108.07732, (2021)
2. Billah, M.M., Roy, P.R., Codabux, Z., Roy, B.: Are Large Language Models a Threat to Programming Platforms? An Exploratory Study. In: Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 292–301. ACM, Barcelona Spain (2024)
3. Brown, T.B. et al.: Language Models are Few-Shot Learners, http://arxiv.org/abs/2005.14165, (2020)
4. Chen, M. et al.: Evaluating Large Language Models Trained on Code, http://arxiv.org/abs/2107.03374, (2021)
5. Dai, D. et al.: DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-

Experts Language Models, http://arxiv.org/abs/2401.06066, (2024)
6. Dam, S.K., Hong, C.S., Qiao, Y., Zhang, C.: A Complete Survey on LLM-based AI Chatbots, http://arxiv.org/abs/2406.16937, (2024)
7. DeepSeek-AI et al.: DeepSeek-V3 Technical Report, http://arxiv.org/abs/2412.19437, (2025)
8. DeepSeek-AI: Introducing DeepSeek App | DeepSeek API Docs, https://api-docs.deepseek.com/news/news250115, Accessed: April 06, 2025
9. DeepSeek-AI: Models & Pricing | DeepSeek API Docs, https://api-docs.deepseek.com/quick_start/pricing, Accessed: April 06, 2025
10. Gao, A.K.: Prompt Engineering for Large Language Models, http://dx.doi.org/10.2139/ssrn.4504303, (2023)
11. Islam, R., Moushi, O.M.: GPT-4o: The Cutting-Edge Advancement in Multimodal LLM, https://www.techrxiv.org/users/771522/articles/1121145-gpt-4o-the-cutting-edge-advancement-in-multimodal-llm?commit=61ade21aa7b17723ee28564be40f083768c69df8, (2024)
12. Konda, R.: AI-Powered Code Generation Evaluating the Effectiveness of Large Language Models (LLMs) in Automated Software Development. J. Artif. Intell. Cloud Comput. 1–6 (2023)
13. Koubaa, A., Boulila, W., Ghouti, L., Alzahem, A., Latif, S.: Exploring ChatGPT Capabilities and Limitations: A Survey. IEEE Access. 11 118698–118721 (2023)
14. Liu, Y. et al.: Summary of ChatGPT-Related research and perspective towards the future of large language models. Meta-Radiol. 1 (2), 100017 (2023)
15. Liu, Z., Tang, Y., Luo, X., Zhou, Y., Zhang, L.F.: No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT, http://arxiv.org/abs/2308.04838, (2024)
16. Manik, M.H.: ChatGPT vs. DeepSeek: A Comparative Study on AI-Based Code Generation, https://doi.org/10.48550/arXiv.2502.18467, (2025)
17. Normile, D.: Chinese firm's large language model makes a splash. Science. 387(6731) 238 (2025)
18. OpenAI: Download ChatGPT | OpenAI, https://openai.com/chatgpt/download/, Accessed: April 06, 2025
19. OpenAI et al.: GPT-4o System Card, http://arxiv.org/abs/2410.21276, (2024)
20. OpenAI: Models - OpenAI API, https://platform.openai.com/docs/models/gpt-4o, Accessed: April 06, 2025
21. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving Language Understanding by Generative Pre-Training, https://api.semanticscholar.org/CorpusID:49313245, (2018)
22. Sakib, F.A., Khan, S.H., Karim, A.H.M.R.: Extending the Frontier of ChatGPT: Code Generation and Debugging, http://arxiv.org/abs/2307.08260, (2023)
23. Shakya, R., Vadiee, F., Khalil, M.: A Showdown of ChatGPT vs DeepSeek in Solving Programming Tasks. Presented at the 2025 International Conference on New Trends in Computing Sciences (ICTCS), (2025)
24. Sohail, S.S. et al.: Decoding ChatGPT: A Taxonomy of Existing Research, Current Challenges, and Possible Future Directions. J. King Saud Univ. - Comput. Inf. Sci. 35 (8), 101675 (2023)
25. Vaithilingam, P., Zhang, T., Glassman, E.L.: Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In: CHI Conference on Human Factors in Computing Systems Extended Abstracts. pp. 1–7. ACM, New Orleans LA USA (2022)
26. Vaswani, A. et al.: Attention Is All You Need, http://arxiv.org/abs/1706.03762, (2023)
27. Wu, T. et al.: A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development. IEEECAA J. Autom. Sin. 10 (5), 1122–1136 (2023)