IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

AI-Driven Software Development: Impact of Coding Assistants (Copilot, Gemini) on Productivity & Code Quality

Mrs. Swati Kale[1], Mr. Aditya Sanap[2], Mr. Prem Nhavi[3], Mr. Roshan Thakare[4], Mr. Pritam Thakare[5]
Prof., Students Computer Engineering Department, Adsul's Technical Campus, Ahilyanagar, India [1,2,3]
Students, Information & Technology Engineering Department, Adsul's Technical Campus, Ahilyanagar, India[4,5]

Abstract: *Artificial intelligence (AI) has rapidly increased the pace of software development, and there are now indispensable tools like AI powered code completion tools that developers rely on. In short, machine learning, deep learning and natural language processing (NLP) are leveraged to make coding more efficient, reduce development time and increase code quality with these tools. This review paper analyzes AI based code completion tools, their underlying technologies and its influence on developer productivity. We review the most popular AI led solutions, OpenAI Codex (GitHub Copilot), Tabnine, and Amazon CodeWhisperer to discuss their features, pros and cons. Additionally, we address the problems faced by AI coding including accuracy, ethics and dependency on such tools from developers. Moreover, the paper also covers the upcoming trends that involve personalized code recommendations and AI augmented pair programming for intelligent software development in the future. This review synthesized current research and industry progress to provide an inventory into the advancement of the use of AI based code completion, the potential and challenges, for future innovations in this area.*

Keywords: AI-based code completion, software development, developer productivity, machine learning, deep learning, natural language processing, GitHub.

I. INTRODUCTION

Over the recent few year, the Software development field transformed due to the rising need for complex, scalable and high quality applications. However, as software systems continue to increase in complexity, this has become a difficult task especially when it comes to writing large volumes of code, ensuring accuracy, and maintaining their efficiency. While traditional coding methods like manual typing and rule based auto completion go a fair way, they lack in resolving challenges of today and more. Addressing these challenges, the revolutionary solution developed are AI based code completion tools. These tools use machine learning techniques, such as deep learning and natural language processing along with these techniques provide assistance to developers by predicting and suggesting relevant code snippet in real time. Figure-1: AI tools. AI driven code completion reduces cognitive load of programmers and errors which further increases the productivity and efficiency. Given the growing amount of AI tools being adopted in software development, it is important to understand the evolution, effectiveness, and possible of such tools.

The software development industry is experiencing a profound and rapid transformation, driven by the integration of generative artificial intelligence (AI). What began as a simple code completion feature has evolved into a sophisticated collaborative tool known as an "AI pair programmer". This new paradigm challenges the traditional model of solo development, replacing it with a symbiotic relationship where a human developer acts as the "driver" and the AI acts as a "navigator," offering real-time suggestions, writing boilerplate code, and assisting with complex tasks like refactoring and documentation. Major tech companies, including Google and Amazon, have launched their own advanced AI coding assistants, such as Gemini Code Assist and Amazon Q Developer, respectively, to compete with pioneers like GitHub Copilot.

DOI: 10.48175/568

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

Fig.1: AI Tools

These tools are designed to streamline the software development lifecycle by automating repetitive and time-consuming tasks, enabling developers to focus on higher-level design and innovation. This shift is not merely a technological upgrade but a strategic move, with many organizations mandating the use of these tools to gain a competitive edge. This growing adoption necessitates a comprehensive and critical evaluation of the true impact of these technologies on the industry.

Scope & Structure of paper:

The key challenges of AI code completion, including accuracy limitation, ethical concern, as well as dependence on developers, are further discussed in Section 5. Section 6 discusses some emerging trends and research directions in software development enabled with AI. Section 7 concludes the key findings and comments on the future of AI driven coding.

Objective of Project:

This review aims to:

• Analyze the development and advancements in AIbased code completion tools.
• Explore the key technologies enabling intelligent code suggestions, such as deep learning models and NLP techniques.
• Compare and evaluate existing AI-driven code completion tools in terms of accuracy, usability, and performance.
• Assess the impact of AI-assisted coding on developer productivity and efficiency.
• Identify challenges and limitations of AI-based code completion, including ethical concerns and technical constraints.
• Discuss future trends and research directions in AIdriven software development.

II. LITERATURE SURVEY

A review of the existing body of literature reveals a complex and often contradictory picture of the effectiveness of AI pair programmers.

Productivity and Efficiency: Multiple studies confirm that AI-driven code generation significantly boosts developer productivity. A randomized controlled trial with 96 Google engineers found that using AI features reduced the time developers spent on a complex task by 21%. Similarly, a study by GitHub and Accenture reported that AI pair programming helped developers code up to 55% faster, leading to an 8.69% increase in pull requests and an 84% increase in successful builds. A separate enterprise field study by Faros AI noted a 55% decrease in "lead time to

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

production," a critical metric for development velocity. However, not all findings are uniformly positive. Some anecdotal and observational data suggest that experienced programmers can be up to 19% slower with AI tools, largely due to the new cognitive demands of writing clear prompts and meticulously reviewing the AI's output. The effectiveness of these tools appears to be task-dependent, with the largest efficiency gains seen in routine, standalone tasks and diminishing returns on more complex, multi-step projects.

Code Quality and Security: The impact of AI on code quality presents a paradox. On one hand, AI assistants are highly effective at reducing common errors; a report by Apiiro found that AIassisted code saw a 76% reduction in syntax errors and a 60% reduction in logic bugs. On the other hand, several studies have linked the use of these tools to a surge in complex and dangerous security vulnerabilities that often evade automated security tools and human review. A comprehensive Veracode report, which tested over 100 LLMs, found that models generated insecure code in 45% of cases when no specific security instructions were provided. The security failure rate was particularly high for Java (over 70%) and for common flaws like cross-site scripting (86%) and log injection (88%). Furthermore, research from Apiiro revealed a surge in complex architectural flaws, including a 322% increase in privilege escalation issues and a 153% increase in design problems. AI-assisted developers were also found to expose cloud credentials nearly twice as often as their non-AI counterparts. This is further compounded by a novel threat vector known as "slopsquatting," where AI can hallucinate non-existent software packages that attackers can then register with malicious code.

Developer Experience and Skill Evolution: Developers' subjective experience with these tools is largely positive. A GitHub study found that 95% of developers enjoyed coding more when using an AI assistant. This is largely attributed to a reduction in "cognitive grunt work," such as writing repetitive code and searching for information, allowing developers to maintain a "flow state" and focus on strategic, higher-level problem-solving. However, this convenience has sparked a debate on "deskilling," as AI automates tasks traditionally performed by junior developers, such as writing boilerplate code. Some researchers argue that this could undermine the development of foundational coding skills, as developers may become overly reliant on AI for "quick fixes" rather than learning the underlying principles. Conversely, other studies view AI as an "always-on mentor" that accelerates learning by providing instant feedback and explaining complex concepts.

Research Questions and Hypotheses

Based on this review, this study aims to address the following core research questions:

1. RQ1: What is the quantifiable impact of AI pair programmers on developer productivity, and what are the mediating factors of this impact?
2. RQ2: How do AI pair programmers influence code quality and security, and what are the primary trade-offs?
3. RQ3: What are the long-term implications of AI collaboration for a developer's skill set and career trajectory, and how can organizations mitigate potential negative outcomes?

Significance of the Study

This research is significant because it provides a consolidated and data-driven perspective on a rapidly evolving technology. With 76% of developers already using or planning to use AI coding tools and major corporations mandating their use , a comprehensive evaluation of the benefits and risks is essential. This report provides a crucial framework for organizations to develop effective AI governance policies and for developers to proactively adapt their skills to thrive in a new era of human-AI collaboration.

III. FUNDAMENTALS OF COMPLETION TOOLS

One of the stellar features that helps developers write code faster and with less mistakes is code completion tools, and it is now compulsory for modern software development. They have these tools, which are intelligent suggestions based on the context; these do the work for repetitive coding task and in addition it improves overall productivity. While artificial intelligence (AI) has come a long way from traditional code completion techniques, AIdriven solutions for

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

code completion have evolved into solutions with increasingly more sophisticated accuracy and adaptability. In this section, it defines a code completion tool, discusses types of code completion tools, and lists out the key technologies involved when one uses code completion tools.

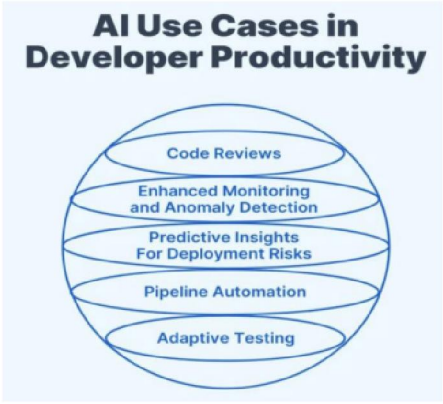

Fig.2: Developer Productivity

Key Technologies in AI-Based Code Completion

A. Natural Language Processing (NLP):

Natural Language Processing (NLP) is an Artificial Intelligence (AI) filed that allows the machines to understand and create the human like text. In terms of code completion, NLP is employed in AI models such that they can analyze code syntax, structure, intent just like they do for natural languages. AI tools that leverage NLP can suggest based on past code patterns and context. This gap is bridged by the capacity for the deterministic program to transform such a model into human readable code, one facet of the ability that also improves code completion accuracy.

B. Deep Learning and Transformer Models :

Transformer based models have to single handily disrupted the field of AI code suggestion. More key transformer models include the GPT (Generative Pre trained Transformer) which is capable of learning from massive open source code datasets and prediction of human like text /code. OpenAI Codex is an example of GPT in action — it's the technology powering GitHub Copilot.

C. Reinforcement Learning

Another technique used to improve code completion tools is Reinforcement Learning (RL). RL based approaches train the AI model with a bunch of trial and error to the past completions. Based on developer's acceptance or rejection of suggestion, it gives feedback and model keeps improving its predictions by reinforcing successful completions. It helps with the AI-powered code completion tools to adjust the code on the individual style of the coder, and the predictions over time will become more personalized and relevant

IV. IMPACT ON DEVELOPER PRODUCTIVITY & EFFICIENCY

The software development process has been significantly changed by the AI based code completion tools that are improving the productivity and efficiency. They cut down on the amount of time needed to write code, enhance the quality of code, and decrease debugging time, among others; and are designed to run on multiple programming languages. Furthermore, developer feedback and usability studies indicate that AI driven code completion enhances the workflow and speeds up the software development. The impacts of these have been examined in detail.

A. Reduction in Code writing time

The primary benefit of the use of AI powered code completion tools is they save us code writing time. These tools suggest real time code which requires only minimal manual typing from the developers. Automated code generation is

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

one of the key contributions of AI driven tools like GitHub Copilot and Amazon CodeWhisperer can generate the functions, loops and the full class with minimal input. They also give autocomplete for boilerplate code that generates repetitive code structures such as getters, setters, and database queries, saving lots of time. Additionally, context aware predictions are used by AI driven tools to understand the wider context of the code, make better suggestions as compared to traditional autocomplete.

B. Improvement in Code Quality:

AI driven code completion tools not only allow for faster coding but also help in creating better quality code. However, these tools do suggest best practices, prohibit most common mistakes, and improve maintainability. It would include standardized code practices (where by AI models that are trained on the most effective practices would recommend that the code is standardized and clean), and error prevention (where AI powered tools detect potential errors like incorrect variable use, missing parameters or improper syntax). Indeed, these tools facilitate code readability by proposing optimized and structured code, which makes them a better means to maintain the code.

C. Reduction in Debugging and Error Rates:

Another time-consuming aspect of software development is debugging, but AI-driven code completion tools help to reduce the error rate by reducing the chance of mistakes by preventing them through completion, and assisting with debugging. Among the key contributions are error prediction and prevention, as AI tools not only detect where an error might occur (based on context) but also suggest the error-free code that avoids syntax and logical errors; and automated error fixes where some tools also provide automatic suggestions for solving common issues e.g. missing imports or incorrect function calls. Moreover, many of these AI inspired IDEs also support linters and debuggers as the corrections are carried out on the fly.

D. Support for Multiple Programming Languages:

A vast variety of programming languages are supported by AI-based code completion tools, which makes them useful to developers working in a variety of technology stacks. Other key contributions is being multi language compatible which GitHub Copilot, Tabnine, etc to write code for any languages like Python, JavaScript, Java, C++, Go, and other languages (they directly target the editor too). In addition to most supported platforms (Windows, Mac, etc), these tools also support cross framework and library, offering suggestions across various frameworks such as React, Django, Flask, and so on. Besides, AI tools evolve with developer preference with identifying user behaviour and enhancing the suggestions based on the coding style of users.

E. Developer Feedback and Usability Studies

AI code completion tools have been widely adopted by many, and researchers have learned much of their utilisation, how they are being used, and their strengths and limitations from developers. An analysis of developer feedback yields insights that AI-assisted tools facilitate higher productivity by allowing developers to complete tasks faster and less frustratingly when they are repetitive coding tasks. In addition to being such useful tools, these also help the learning experience as real time coding assistant to junior developers to understand how to code by best practice. The customization and adaptability of the AI model to learn the developer's coding style and offer personalized suggestions is quite appreciated by many developers.

IV. CHALLENGES & LIMITATION OF AI-BASED CODE COMPLETION

AI-based code completion tools have transformed software development, significantly improving productivity and efficiency. However, these tools also come with several challenges and limitations. Issues such as accuracy, ethical concerns, resource constraints, and the potential over-reliance of developers on AI-generated code need to be addressed to maximize their benefits. This section discusses these challenges in detail.

A. Accuracy and Context Awareness Issues

AI based code completion tools are one of the major limitation when it comes to the accuracy and understanding full context of a developer's code. Modern AI models, based on the latest machine learning techniques, still lack one or other area of code generation. The contextual misinterpretation is also a challnage for AI models, in which the AI

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

model sometimes generates the code looking correct but it doesn't follow the real logic of the program. Such AI tools can also predict inconsistently, by offering different suggestions for similar inputs, which makes it difficult for the developers to rely on them.

B. Ethical and Security Concerns

AI-based code completion tools pose several ethical and security risks, including code plagiarism, intellectual property violations, and biases in AI-generated code.

• Code Plagiarism and Intellectual Property Issues

Mixing in Tabnine or GitHub Copilot, which are developed using AI models trained on huge quantities of openly accessible code, has caused anxiety that the code that's generated by the two companies may not be of their own creation.

• Biases in AI-Generated Code

Since AI models are trained on datasets, and that datasets have biases, the generated code has its own biases, which are born out of the dataset used. This leads to ethical and security concerns. Bias in gender and race is also a key issue with code suggestions from AI as algorithms often assume and take for granted things like suggestive gender names or functions, etc. Moreover, training data bias is a security risk for AI, as code generated by AI models trained upon obsolete or insecure coding patterns could result in incorporating unsafe code with vulnerabilities.

• Performance and Resource Constraints

That is why AI powered code completion tools require too much of the compute, which can cause performance issue for the developers using lower end hardware. High latency, for example, where AI generated code suggestions sometimes takes a while to show therefore disrupting workflow. Furthermore, it's an expensive computational overhead to run deep learning models for code completion on the CPU or GPU, making users wait until all the other development tasks are completed. Moreover, many AI based tools are dependent on cloud based inference, which means an internet connects is needed for its use and they can't be used offline.

• Developer Dependence on AI Tools

New and improved AI powered code completion tools will make developers rely too much on them, which in turn can make them lose their problem solving and coding skills. Biggest concerns are becoming less critical thinking in terms of new developers simply looking up to AI generated code without too deep understanding what's wrong and hence they may get it in wrong path, also coding fundamentals might start degrading even further, especially with junior developers who have hard time with problem solving if they just completely rely on AI suggestions. Moreover, it can also lead to the poor development of debugging skills as overuse of AI generated code will make developer struggle with debugging and troubleshooting.

V. CONCLUSION

This report has provided a comprehensive overview of the current landscape of AI-assisted code generation. It has demonstrated that while these tools offer significant, measurable increases in developer productivity and job satisfaction, they also introduce profound and complex challenges, particularly in the realm of code security. The analysis highlights a fundamental trade-off: the speed and convenience of AI can, without proper oversight, lead to a proliferation of deeper, more dangerous vulnerabilities. The automation of routine coding tasks is reshaping the role of the software engineer, moving the focus from low-level implementation to high-level architectural design, critical evaluation, and strategic management of AI-driven systems.

They have streamlined software development process through making it more productive, producing better quality code and saving debugging time. This review further discussed the evolution of these tools, comparing and contrasting the traditional and AI driven models, as well as the extent to the developer efficiency they impact, and finally touched upon concerns they have namely the accuracy, the ethical problems, and the computational limitations. While AI powered code assistants can be extremely transformative, they come with their set of drawbacks such as possible biases in

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

generated code, security risks and developers becoming over reliant on the code amenities. In the future, AI will continue to contribute significantly in software development by developing more comprehensive large language models, providing personalized code suggestions, evolving AI paired programming and guiding responsible AI practices in order to craft the next generation of tools. Further research should be done on enhancing contextual understanding, removing biases, and including ethical AI concepts within code generation to enhance their utilization. Moreover, developers must find a fine line between utilizing AI assistance and being incompetent in basic core skills of programming. With the emergence of these issues and the adoption of innovation, AI code completion tools will develop further and ultimately revamp how software is coded and rebuilt.

VI. FUTURE SCOPE

The future of software development is not a binary choice between humans and machines. It is, and will increasingly be, a collaborative endeavour where AI serves as a powerful new layer of abstraction. The challenge is not to resist this change but to adapt to it responsibly. The human developer's role is evolving from a coder to an orchestrator, a designer, and an ethical guardian of the systems they build.

The ultimate success of this new paradigm will depend on our ability to design intelligent systems that are not just efficient but also transparent, reliable, and secure by default.

A lot has happened in the last few years to the point that AI based code completion tools have already made leaps and bounds but the field is still moving fast. The next generation of AI driven development tools will be defined by future developments in the use of large language models for code generation, in context aware code generation, in the use of AI to augment pair programming, and in the ethical application of AI. In this section we examine some main research directions and novel trends in AI powered code auto completion.

VII. ACKNOWLEDGMENT

It gives us great pleasure in presenting the paper on “AI-Driven Software Development: Impact of Coding Assistant (Copilot, Gemini) on Productivity & Code Quality”. We would like to take this opportunity to thank our guide, Prof. Swati Kale, Professor, Computer Department, Adsul’s technical Campus, Ahlyanagar, for giving us all the help and guidance we needed. We are grateful to her for hers kind support, and valuable suggestions were very helpful.

REFERENCES

[1]. Apiumhub. (2025). Ethical Considerations in AI Development. Retrieved from https://apiumhub.com/tech-blog-barcelona/ethical-considerations-ai-development/
[2]. Adebayo, Y., Basha, S. S., & Ghaffar, A. (2023). AI-Driven Forecasting Models for Green Tech Stocks: Linking Carbon Capture Innovation to Stock Market Trends in 2025.
[3]. Armis Labs. (2025). Latest Armis Labs Report Uncovers New Security Risks of AI Coding. Retrieved from https://www.armis.com/blog/latest-armis-labs-report-uncoversnew-security-risks-of-ai-coding/
[4]. Bakal, G., Dasdan, A., Katz, Y., Kaufman, M., & Levin, G. (2025). Experience with GitHub Copilot for Developer Productivity at Zoominfo. arXiv. Retrieved from https://arxiv.org/abs/2501.13282
[5]. Adebayo, Yusuf & Basha, Syed & Ghaffar, Adnan. (2023). AI-Driven Forecasting Models for Green Tech Stocks: Linking Carbon Capture Innovation to Stock Market Trends in 2025.
[6]. Crowston, K., & Bolici, F. (2025). Deskilling and upskilling with AI systems. Information Research, 30(iConf).
[7]. Ghaffar, A., Gajiwala, C., Basha, S., & William, B. (2024). Integrating AI into Business Automation: Practical Frameworks for Streamlining Operation.
[8]. Economic Times. (2025). AI can fix typos but create alarming hidden threats, new study sounds warning for techies relying on chatbots for coding. Retrieved from https://m.economictimes.com/magazines/panache/ai-

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

can-fix-typos-but-create-alarminghidden-threats-new-study-sounds-warning-for-techies-relying-on-chatbots-forcoding/articleshow/123783245.cms

[9]. Ghaffar, Adnan & Gajiwala, Chirag & Basha, Syed & William, Bruce. (2024). Integrating AI into Business Automation: Practical Frameworks for Streamlining Operation.

[10]. Endor Labs. (2025). The most common security vulnerabilities in AI-generated code. Retrieved from https://www.endorlabs.com/learn/the-most-common-securityvulnerabilities-in-ai-generated-code

[11]. De Moor, A. van Deursen, and M. Izadi, "A Transformer-Based Approach for Smart Invocation of Automatic Code Completion," arXiv preprint arXiv:2405.14753, May 2024. [Online]. Available:

[12]. F. Liu, G. Li, Y. Zhao, and Z. Jin, "Multi-task Learning based Pre-trained Language Model for Code Completion," arXiv preprint arXiv:2012.14631, Dec. 2020. [Online]. Available:

[13]. Y. Li, Y. Peng, Y. Huo, and M. R. Lyu, "Enhancing LLMBased Coding Tools through Native Integration of IDE-Derived Static Context," arXiv preprint arXiv:2402.03630, Feb. 2024. [Online]. Available:

[14]. H. Vasconcelos, G. Bansal, A. Fourney, Q. V. Liao, and J. W. Vaughan, "Generation Probabilities Are Not Enough: Exploring the Effectiveness of Uncertainty Highlighting in AI-Powered Code Completions," arXiv preprint arXiv:2302.07248, Feb. 2023.