International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 12 Issue 2, February, 2023, Impact Factor: 7.751

Explore the Benefits and Limitation of GitHub Copilot

Roy Gill[1], Wilda Hardy[2], Xiang Chen[3], Bill Zhang[4]

[1,4]Amazon Web Service, Amazon
[2,3]Department of Computer Science, Boston University

ABSTRACT

GitHub has launched Copilot since 2021. It is an AI assistant to help developers to write code. It has become very popular since its launch. GitHub has announced that a million software engineers have begun to use it during their daily work and 20,000 organizations have adopted it in their work process[3]. According to a research from GitHub, there will be an increase in productivity from using Copilot and it is expected to add $1.5 trillion to global GDP by 2030[11]. Although GitHub Copilot has brought us a lot of benefits and improvements in the software development process, many developers are aware of its shortage and limitations. This paper discusses Copilot's impact on the software industry and illustrates what are the pros and cons of this AI tool. The study is based on many online resources and my daily work experience in the company. This paper points out the negative and positive impacts of Copilot and demonstrates corresponding solutions to avoid those shortcomings. In the end, it aims to come up with better approaches to using Copilot to enhance the software development work.

INTRODUCTION

GitHub Copilot is a code generator based on the GPT-3 model from Open AI. It is a cloud-based artificial intelligence tool that can turn natural language prompts into programming suggestions. It helps developers generate code snippets and gives useful coding suggestions for multiple programming languages including Python, Java, C++, and so on. Copilot helps software engineers save plenty of time and improve their productivity greatly. It aims to help developers focus on business logic over boilerplate code. However, Copilot is not perfect and it does have many limitations and challenges. This paper highlights its risks and corresponding approaches to overcome those risks in order to utilize this tool better in the software development.

BACKGROUND AND MOTIVATION

GitHub Copilot is powered by Codex, which is a modified version of GPT-3 (Generative Pre- trained Transformer 3)[10]. GPT-3 is a neural network machine learning model trained using internet data to generate human-like texts. It is a decoder-only transformer model (see Fig.1), which uses attention in place of previous recurrence and convolution-based architectures. The attention mechanisms allow the model to selectively focus on segments of input text it predicts to be the most relevant[10]. GPT-3 has been used to generate different types of text structures, such as articles, new reports, and dialogue, it can also generate programming code.

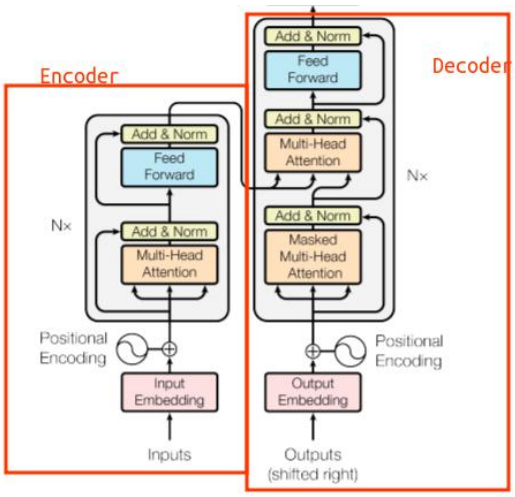


Fig.1. Polosukhin et al. "Attention Is All You Need". arXiv:1706.03762

Codex is a special version of GPT-3 since it is trained on large sets of source code from different programming languages on GitHub. Codex has the power to understand natural languages, but it generates code instead of text, meaning you can issue commands in English to any piece of software with an API. Codex is a general-purpose model and people are able to use it for different programming tasks, such as code refactoring, code explanation, and code transpilation (For example, transpilers in JavaScript are source-to-source compilers that transform source code in non-JavaScript languages such as CoffeeScript and TypeScript to equivalent JavaScript source code)[4]. In this way, GitHub Copilot works as the AI pair programmer to help developers to solve the coding problems.

GitHub Copilot has been developed for years to evolve from a simple plugin to a well-known subscription service. The rough design idea was early. The very beginning of it is 'Bing Code Search' plugin for Visual Studio 2013, which was a Microsoft Research project released in February 2014. This plugin integrated with various sources, including MSDN and StackOverflow, to provide high-quality contextually relevant code snippets in response to natural language queries. Then GitHub Copilot was first launched in the Visual Studio code development environment in Jun 2021[3]. Then it was released as a plugin in JetBrains in Oct 2021. After that, GitHub released the GitHub Copilot Neovim plugin as a public repository. In Jun 2022, Copilot was officially available as a subscription service for all developers (GitHub Copilot is generally available to all developers). GitHub Copilot is most capable in Python, but it is also proficient in over a dozen languages including JavaScript, Go, Perl, PHP, Ruby, Swift and TypeScript, and even Shell[12].

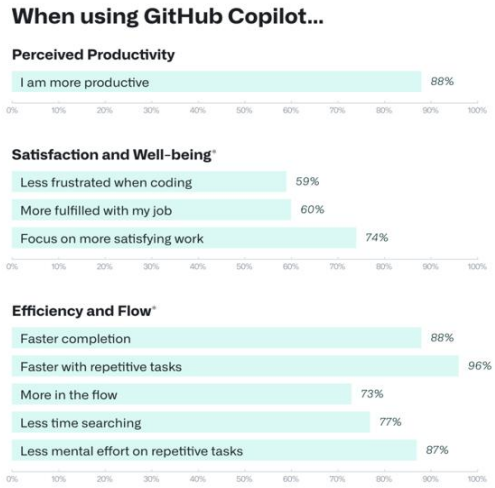
GitHub Copilot has become very popular among software developers rapidly. According to one survey from GitHub, there are more than 1.2 million developers using it in 2022, 72% of all GitHub developers are willing to use it next year, and 75% of GitHub developers have a desire for future adoption[8]. Actually, the survey also shows that there are 46% of the code on GitHub was generated by Copilot in 2022, and 61% of the code is Java. Meanwhile, many companies and organizations have started to use GitHub Copilot[5]. For example, Microsoft has encouraged all its developers to embrace GitHub Copilot in their development. Some companies like Citi and Samsung have shown strong interest in leveraging GitHub Copilot in their software-building process.

GitHub Copilot has significantly improved the productivity of developers. In fact, GitHub has conducted one survey to fully understand the user feedback. In the research, the GitHub Next team surveyed more than 2000 developers to learn about their experience using GitHub Copilot and conducted multiple rounds of research including qualitative (perceptual) and quantitative (observed) data to assemble the full picture. They collect metrics that cover all dimensions of the SPACE framework[7]. SPACE is the acronym for a multidimensional framework that describes several aspects of developer productivity: (1) Satisfaction and well-being; (2) Performance; (3) Activity; (4) Communication and collaboration; and (5) Efficiency and flow. We can clearly find the positive results of using GitHub Copilot in Figure 2. Those metrics in Figure 2 are tracked over time for some specific groups of GitHub users so that we can clearly see the “before and after” effect. According to this research from GitHub, they have found that using Copilot can support faster completion times, conserve developers’ mental energy, help them focus on more satisfying work, and ultimately find more fun in the coding they do.
Fig.2. Kalliamvakou, Eirini. “Survey responses measuring dimensions of developer productivity when using GitHub Copilot”. GitHub Blog. 7 Sep 2022

IMPROVEMENTS AND BENEFITS

GitHub Copilot is branded as an "AI pair programmer" that uses artificial intelligence to auto- generate code in your editor. However, GitHub Copilot is more than just an autocomplete solution. Based on the prompt and context you provide, it offers many different features to help developers solve different programming tasks.

First of all, GitHub Copilot can help convert code comments into runnable code. With GitHub Copilot, developers can generate code much faster than they would be able to write it from scratch. This can be especially helpful when working on large projects or when facing tight deadlines. At the beginning of your code, you just need to type instructions as comments, then Copilot will suggest a solution of code snippet that you can accept or decline. (See Fig.3). GitHub Copilot can help you convert your business logic written in plain text into the



corresponding code. This feature is pretty useful when you write some functions or methods to achieve some straightforward goals[5]. After you have leveraged GitHub Copilot to complete writing many functions or methods, you can easily build a complex class based on them.

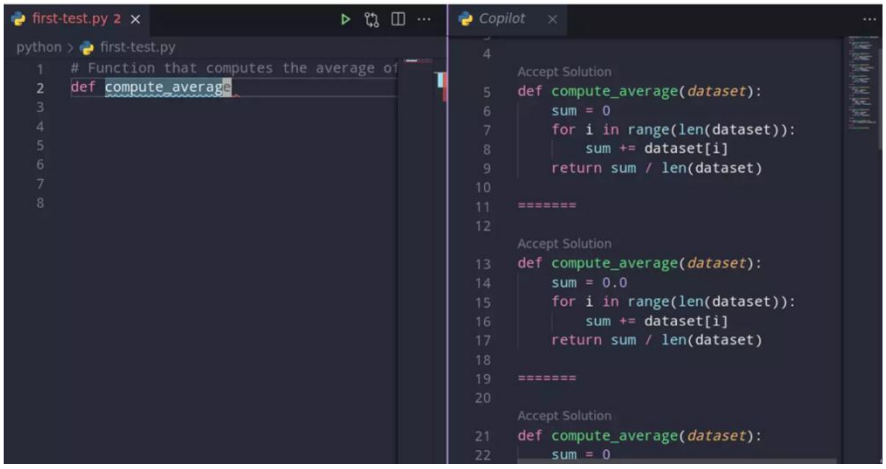


Fig.3. Daniel Diaz. "What is GitHub Copilot? An AI Pair Programmer for Everyone". SitePoint. 9 Aug 2021

Besides, GitHub Copilot is also helpful for developers to understand what a block of code is doing, especially for a new coding framework. GitHub Copilot can help make programming more accessible to people without much coding experience, as it can guide them through the process and provide feedback and explanations. GitHub Copilot helps reduce the barriers to entry for new developers, which can be especially important in a field that is often criticized for being too insular or elitist[1]. Developers can use Copilot to explain the code block that is not familiar or lacks context. (See Fig.4) When we review others' code, usually it costs a lot of time because we need more time to understand the context and

purpose of each method. By leveraging GitHub Copilot, we can easily ask and interact with this AI assistant to quickly understand the code.

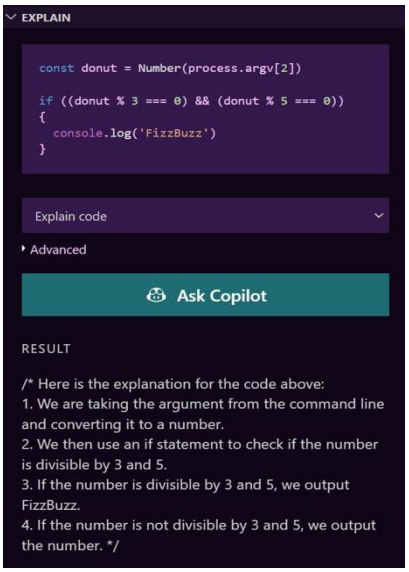


Fig.4. Daniel Diaz. "What is GitHub Copilot? An AI Pair Programmer for Everyone". SitePoint. 9 Aug 2021

Furthermore, GitHub Copilot is very useful for developers to translate their code between different languages or frameworks. For example, developers can use it to convert Node.JS code into Python with simple prompts (See Figure 5). This kind of conversion work is needed when we would like to do some migration to change the programming language for the back-end or front-end. Such migration used to be a very time-consuming task. Now with the help of GitHub Copilot, the migration and conversion work is pretty easy and fast. Developers just need to provide simple prompts and ask the AI to do the work[6]. The generated code is usually of very good quality that can be simply verified using unit and integration tests.

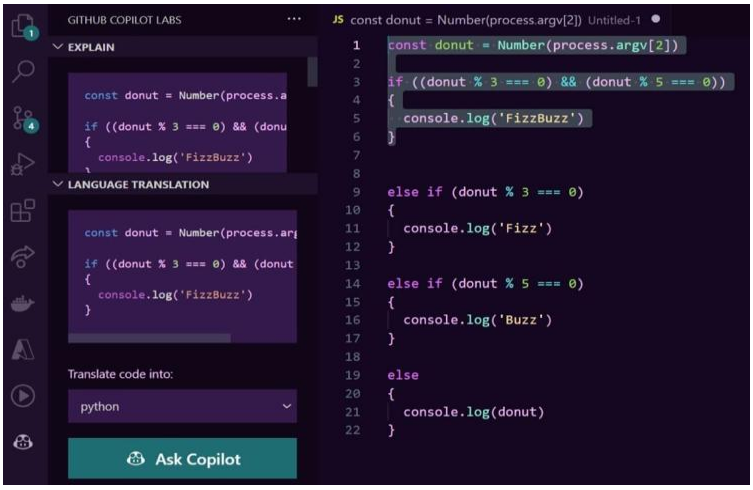


Fig 5. Michelle Mannering. "How to translate code into other languages using GitHub Copilot". DevTo. 16 May 2022

Last but not least, GitHub Copilot is able to improve code quality. Copilot is useful for generating test cases that cover primary and corner cases. This feature is pretty useful as it can save substantial amount of time for developers and increase code coverage at the same time. (See Fig.6) The auto-generated unit test cases will help the code coverage and detect any hidden bugs if necessary[7]. Many developers don't really like to write too many unit tests simply due to the perception that it consumes too much time. Now using GitHub Copilot can easily solve this problem. Those auto-generated unit tests work as very good guardrails and prevent us from checking in any bad code.

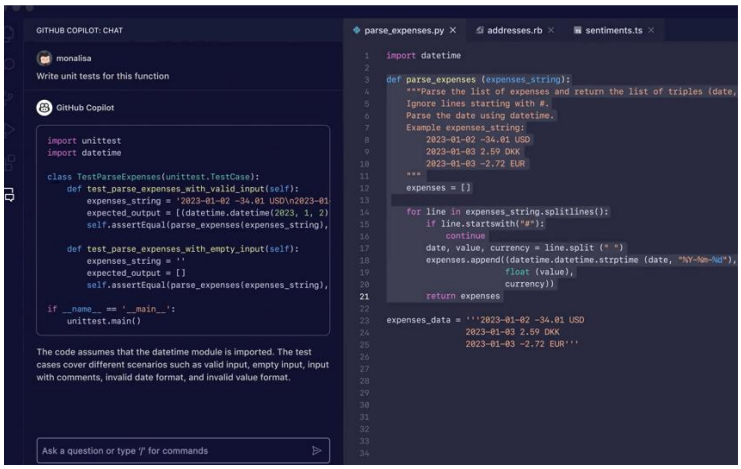


Fig 6. Daniel Diaz. "What is GitHub Copilot? An AI Pair Programmer for Everyone". SitePoint. 9 Aug 2021

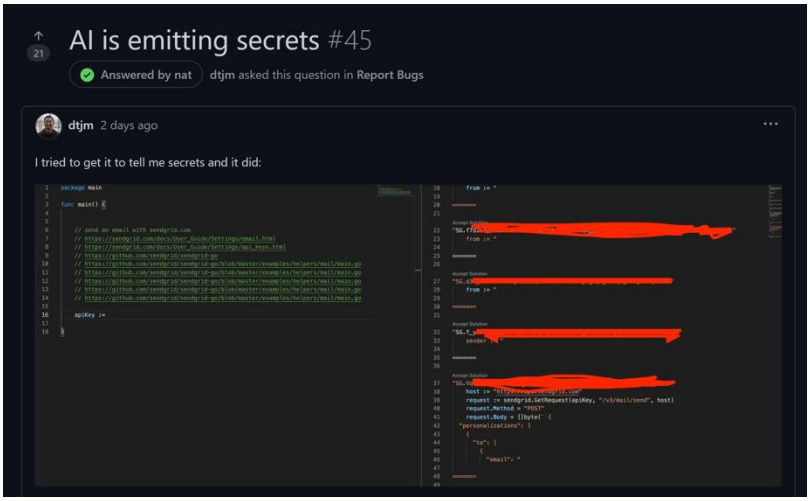
CHALLENGES AND LIMITATIONS

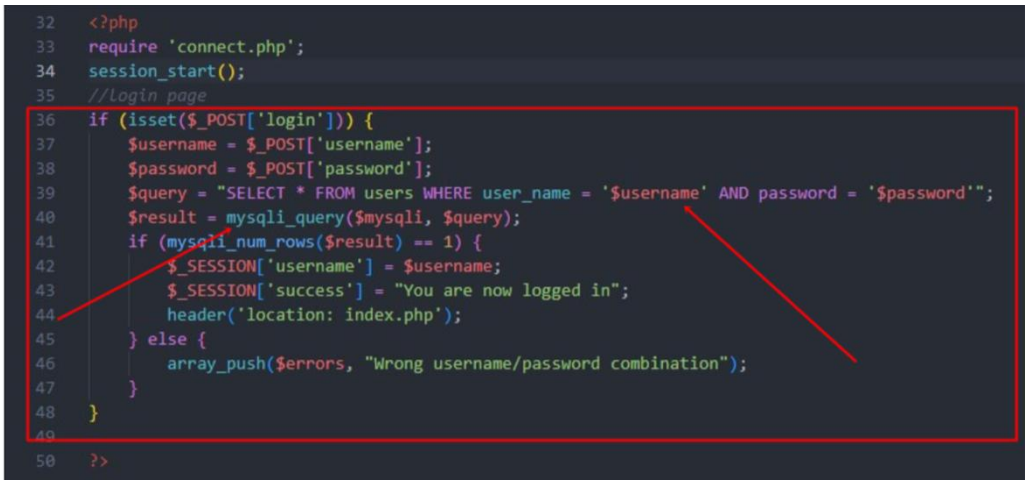
While the benefits and improvements of GitHub Copilot are compelling, there are also some potential drawbacks to consider. Nothing in the world is perfect. This paper highlights some developers' concerns on using GitHub Copilot. Some of those drawbacks shouldn't be easily overlooked as they may block the development of GitHub Copilot if no appropriate action item is taken.

First, despite its popularity among developers, many of them actually have concerns about the security issues of using GitHub Copilot[2]. The security concern does make sense because you basically put data, code, documents, and even images into GitHub Copilot to let it have a better understanding of the context of your questions. The more data you feed into GitHub Copilot, the more risks you have. You may leak those confidential data to public who are not meant to consume such information at all(See Fig 7). Any leaked sensitive data may cause a disaster for the company because other companies may use those data for unfair competition and customers may start a lawsuit against us.

Fig.7. Dotan Nahum. "The Dangers of AI/ML in Code & GitHub's CoPilot Leak". Security Boulevard. 11 July. 2021

Meanwhile, GitHub Copilot may not verify the code it generated. The code may not compile, run, or produce the desired result. More importantly, the code generated by GitHub Copilot may have hidden bugs or security concerns[8]. We know there is a chance that our code can be hacked easily by SQL injection if our SQL is not well written. The code from GitHub Copilot may overlook such security holes. For example, when we create a database and leverage GitHub Copilot to write code for the log-in functionality, it may produce bad code with security issues. As you can see in Fig.8, the SQL query in $query is built in a way that is vulnerable to SQL injection (values supplied by the user are directly used in the query).





```php
<?php
require 'connect.php';
session_start();
//login page
if (isset($_POST['login'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];
    $query = "SELECT * FROM users WHERE user_name = '$username' AND password = '$password'";
    $result = mysqli_query($mysqli, $query);
    if (mysqli_num_rows($result) == 1) {
        $_SESSION['username'] = $username;
        $_SESSION['success'] = "You are now logged in";
        header('location: index.php');
    } else {
        array_push($errors, "Wrong username/password combination");
    }
}

?>
```

Fig.8. Kadir Arslan. "Insecure coding workshop: Analyzing GitHub Copilot suggestions". 6 Oct. 2022

Besides, another concern is related to the copyright of code generated by GitHub Copilot. GitHub Copilot is generating new code based on some existing code on GitHub. Existing code serves as the training data for the AI model[9]. If those existing code belongs to company A, then it is controversial to debate if the newly generated code is still owned by company A. Nowadays many individual developers are just using the newly generated code for free. Is it a good approach to protect the copyright of company A? If the copyright is not well protected, then more and more companies will stop sharing their code on GitHub, which eventually will decrease the inference ability of GitHub Copilot.

Furthermore, many developers are also complaining that GitHub Copilot is good at generating small pieces of code, but it is poor at handling the production code which may contain thousands of lines of code. The production code has some internal abstraction and complexity, which blocks the accurate inference of GitHub Copilot. There is a chance that GitHub Copilot may generate some code that is irrelevant or incorrect for the given task, which is inconsistent or incompatible with the existing code base[15]. As a result of that, the code actually reduces the functionality and reliability of the entire system. Developers cannot purely depend on the GitHub Copilot to write the whole production project. Many prompts and interactions are still heavily needed.

Another major concern from developers is the bad project integration. The generated code from GitHub Copilot usually works fine as itself. However, when it comes to integrating with other frameworks, components, and designed interfaces, many problems will occur. The poor integration is usually because GitHub Copilot does not have enough context of the whole proejct[17]. For example, the generated code may work fine to demonstrate your back-end business logic. However, due to the fact that it has little knowledge of your front-end framework and programming language, the back-end code may not be easily integrated with your front-end interface. Most of time, it will take developers additional time to modify the generated code to comply with the integration interface, which decreases the benefits of GitHub Copilot.

5. Overcoming The Limitations

GitHub Copilot is not a perfect tool, however, it doesn't mean we should stop using it. Actually, based on the research of many papers and the experience of our team in the company, we have figured out several approaches to overcome its limitations.

The most effective way is that we need to introduce code review and manual checks on code generated by Copilot. It is recommended not to fully trust AI-generated Code. We always need code review to check and understand every piece of code to ensure code quality. The majority of the generated code does make sense, however, it may contain some nonsense code in some small areas[20]. A manual check is always necessary. Code review does not only work as a safe guardrail to prevent us from checking in bad code but also gives us one opportunity to learn and understand the context of the generated code itself. Without a good understanding of the code, it is always a risk to merge the code no matter whether the code is auto-generated by AI or written by the developers.

Another good approach is to be aware of introducing sensitive data into GitHub Copilot. We should avoid including customer privacy data and sensitive production data. Instead of that, we can start using dummy data or testing data. This approach is similar to what we do during the mock test. Copilot can be used to help developers generate the structure of the code to demonstrate its core business logic. Using dummy data or mock data is sufficient enough. Developers just need to

pay attention to what feeds into GitHub Copilot and don't be so lazy to just copy and paste everything into Copilot. We need to keep in mind not feeding every piece of code or data into Copilot and always remember to filter out those sensitive data[18].

In order to detect the hidden defect of generated code from GitHub Copilot, developers should always incorporate more integration tests and end-to-end tests to ensure integration. As we all know Copilot is poor at code integration, so integration tests and end to end tests are good opportunities to identify some hidden bugs. Some bugs are hard to find during the code review and will only happen during the execution. Adding more test cases, especially the end-to-end test is a very good approach to ensure the quality of our code. We may not have enough time to check every piece of the generated code. However, as long as the generated code can pass our end-to-end test and major functionality test, we will be pretty confident to ship those code. We need to add more test cases to cover some corner cases and edge cases. Those test cases can be divided into functionality tests and regression tests[15]. The former makes sure the generated code has implemented the business logic correctly, while the latter makes sure the newly introduced code won't break our existing system.

Lastly, one approach that may be overlooked is that we always need to have a detailed design doc and architecture design ready before using Copilot. We need to keep in mind that using GitHub Copilot is to reduce repeating manual work instead of any creative work. GitHub Copilot is a helper for us, not the driver of the project. Before using GitHub Copilot, we need to have a design doc discussion among our team and have a clear roadmap of what we would like to achieve. GitHub Copilot is just here to give better suggestions, not to dominate the entire design process. We need to have a good design of how different components interact, how the data flow works, and how functionalities are implemented. Having a well-designed architecture is the prerequisite to leveraging GitHub Copilot to generate code for our project.

CONCLUSION

The benefits of using Copilot are significant, however, there are also many challenges and limitations. Using Copilot is like a double-edged sword, which requires developers to carefully consider various aspects when deciding whether or not to use it. During the software development process, developers need to follow several approaches and pay attention to their risks, then decide on how to apply Copilot on different projects case by case. We still encourage developers to embrace GitHub Copilot given that it can significantly improve the productivity of programming. Meanwhile, we suggest developers to adopt many approaches like we discussed in this paper to overcome GitHub Copilot's shortcomings and not take everything from GitHub Copilot for granted. It is good to know that GitHub has realized several issues of Copilot and gradually launched and developed new features to address those concerns.

REFERENCES

[1]. Codemotion, "Is GitHub Copilot the Solution to Dev Struggles," Codemotion, 7 Nov 2022, 12 Sep 2023
[2]. Hammond Pearce, Baleegh Ahmad, "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions," IEEE Symposium on Security and Privacy, 2022
[3]. James Miller, "How GitHub's Copilot is Going to Impact Productivity," BairesDevBlog, n.d, 12 Sep 2023
[4]. Arghavan Moradi Dakhel, Vahid Majdinasab, "GitHub Copilot AI pair programmer: Asset or Liability," York University, 14 Apr 2023
[5]. Ben Dickson, "Study provides insights on GitHub Copilot's impact on developer productivity," Venture Beat, 7 Sep 2022, 12 Sep 2023
[6]. Qianou Ma, Tongshuang Wu, "Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI pAIr Programming," Human-Computer Interaction, 9 Jun 2023
[7]. Martin Heller, "Review: GitHub Copilot preview gives me hope, " InfoWorld, 8 Nov 2021, 12 Sep 2023
[8]. Dakota Wong, Austin Kothig, "Exploring the Verifiability of Code Generated by GitHub Copilot, " HATRA workshop at SPLASH, 27 Oct 2022
[9]. Mateusz Jaworski, Dariusz Piotrkowski , "Study of software developers' experience using the Github Copilot Tool in the software development process, " International Conference on Mining Software Repositories (MSR-2023), 12 Jan 2023
[10]. Polosukhin et al. "Attention Is All You Need". arXiv:1706.03762
[11]. Kalliamvakou, Eirini. "Survey responses measuring dimensions of developer productivity when using GitHub Copilot". GitHub Blog. 7 Sep 2022
[12]. Julien Delange. "GitHub Copilot and its impact on the future of coding". Codiga. 5 July
[13]. "Is GitHub Copilot the Solution to Dev Struggles?". Codemotion. 7 Nov 2022
[14]. Daniel Diaz. "What is GitHub Copilot? An AI Pair Programmer for Everyone". SitePoint. 9 Aug 2021

[15]. Kadir Arslan. “Insecure coding workshop: Analyzing GitHub Copilot suggestions”. 6 Oct. 2022
[16]. Hammond Pearce. “Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions”. Security and Privacy. arXiv:2108.09293
[17]. Beiqi Zhang, Peng Liang, “Practices and Challenges of Using GitHub Copilot: An Empirical Study,” The 35th International Conference on Software Engineering and Knowledge Engineering (SEKE), 27 Apr 2023
[18]. Dotan Nahum. “The Dangers of AI/ML in Code & GitHub’s CoPilot Leak”. Security Boulevard. 11 July. 2021
[19]. Michelle Mannering. “How to translate code into other languages using GitHub Copilot”. DevTo. 16 May 2022
[20]. Olimpiu Pop. “GitHub's Copilot Still a Long Way From Autopilot”. InfoQ. 11 Oct. 2021
[21]. Eirini Kalliamvakou. “Research: quantifying GitHub Copilot’s impact on developer productivity and happiness”. GitHub Blog. nd.
[22]. Mir, A. A. (2024). Sentiment Analysis of Social Media during Coronavirus and Its Correlation with Indian Stock Market Movements. Integrated Journal of Science and Technology, 1(8).
[23]. Khokha, S., & Reddy, K. R. (2016). Low Power-Area Design of Full Adder Using Self Resetting Logic With GDI Technique. International Journal of VLSI design & Communication Systems (VLSICS) Vol, 7.
[24]. Mir, A. A. (2024). Transparency in AI Supply Chains: Addressing Ethical Dilemmas in Data Collection and Usage. MZ Journal of Artificial Intelligence, 1(2).
[25]. Mir, A. A. (2024). Optimizing Mobile Cloud Computing Architectures for Real-Time Big Data Analytics in Healthcare Applications: Enhancing Patient Outcomes through Scalable and Efficient Processing Models. Integrated Journal of Science and Technology, 1(7).
[26]. Mir, A. A. (2024). Adaptive Fraud Detection Systems: Real-Time Learning from Credit Card Transaction Data. Advances in Computer Sciences, 7(1).
[27]. Chen, X. (2024). AI for Social Good: Leveraging Machine Learning for Addressing Global Challenges. Innovative Computer Sciences Journal, 10(1).
[28]. Chen, X. (2023). Real-Time Detection of Adversarial Attacks in Deep Learning Models. MZ Computing Journal, 4(2).
[29]. Chen, X. (2024). AI and Big Data: Leveraging Machine Learning for Advanced Data Analytics. Advances in Computer Sciences, 7(1).
[30]. Chen, X. (2023). Optimization Strategies for Reducing Energy Consumption in AI Model Training. Advances in Computer Sciences, 6(1).
[31]. [31]Wu, H., Bansal, P., Liu, Z., Wang, P., & Li, Y. (2024). Uncertainty quantification of mechanical behavior of corroded Al-Fe self-pierce riveting joints with statistical shape modeling. Journal of Manufacturing Processes, 124, 909-917.
[32]. Yu, H., Khan, M., Wu, H., Zhang, C., Du, X., Chen, R., ... & Sawchuk, A. P. (2022). Inlet and Outlet Boundary Conditions and Uncertainty Quantification in Volumetric Lattice Boltzmann Method for Image-Based Computational Hemodynamics. Fluids 2022, 7, 30.
[33]. Wu, H. (2022). Probabilistic Design and Reliability Analysis with Kriging and Envelope Methods (Doctoral dissertation, Purdue University).
[34]. Wu, H., Xu, Y., Liu, Z., & Wang, P. (2023, August). Mean Time to Failure Prediction for Complex Systems With Adaptive Surrogate Modeling. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 87301, p. V03AT03A051). American Society of Mechanical Engineers.
[35]. Wu, H., Khan, M., Du, X., Sawchuk, A. P., & Yu, H. W. (2019). Reliability analysis for image-based non-invasive pressure quantification in Aortorenal artery systems. Circulation Research, 125(Suppl_1), A122-A122.