AI's Effects on Software Engineering Jobs

A Research Paper

In STS 4600

Presented to

The Faculty of the

School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment of the Requirements for the Degrees

Bachelor of Science in Computer Engineering

Bachelor of Science in Computer Science

By

Ethan Ermovick

April 16, 2025

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Joshua Earle, Department of Engineering and Society

Gang Tao, Department of Electrical and Computer Engineering

Miaomiao Zhang, Department of Computer Science

Introduction

AI is a relatively recent technology that is reshaping the lives of many people in the US and across the world. AI is able to simulate the work of humans in new ways (University of Illinois Chicago, 2024). One such field where AI has the potential to do the work of people is in Software Engineering. In this paper, I explore AI and software engineering to answer the question, "How is AI affecting jobs in software engineering?". More specifically, we will examine whether AI is replacing or supplementing human software engineers, and how the trends today might answer this question tomorrow.

What is AI?

As described by Amazon, one producer of AI models, AI is a technology able to emulate the problem solving capabilities of people. The work of AI looks like it was done by people, and it is able to write and make predictions based on data (AWS, 2024). AI is a large umbrella term that encompasses many different ways that a computer can be intelligent. One of the most prominent types of AI, and the one that this paper will focus on, is called a Large Language Model, or LLM for short. A Large Language model is a type of AI that, as the name suggests, works using language (Cloudflare, 2022). A typical Large Language Model is able to read an input text or "prompt" supplied to it, then generate an output text attempting to reply to the prompt.

There are many different LLM models that exist today. A model is a particular creation of an LLM, and the responses for a given input can differ between different models. Popular models include OpenAI's ChatGPT (OpenAI, 2022), Google's Gemini (Pichai & Hassabis, 2023), and more recently Deepseek (DeepSeek, n.d.). Responses can also differ with the same input between

trials using the same model, as models often incorporate a degree of randomness into their output responses (IBM, 2024).

Large Language Models work by operating on tokens. Tokens are parts of words that language models use internally, rather than operating directly on words (OpenAI, 2024b). Tokens are the smallest unit of language that AI language models can understand, and have various semantic meanings to the model (IBM Watsonx as a Service, 2024). A text often has more tokens than it does words. For example, this sentence has nine tokens. The token count of the previous sentence was found using a tokenizer by OpenAI (OpenAI API, n.d.). These tokens are then fed into the AI model, which effectively predicts the next token until it has generated a sufficient amount of text (Fitch, 2024). This can be done in multiple ways, but one popular way is the transformer model, which was introduced in a paper titled "Attention Is All You Need" by Vaswani et. al. in 2017 (Vaswani et al., 2017). The transformer model uses many matrix multiplication operations to both train and use the AI model, which are efficiently parallelizable using graphics processing units, and libraries such as cuBLAS, which stands for the CUDA Basic Linear Algebra Subroutine library (CuBLAS, n.d.). The ability to efficiently train AI models on GPUs, as well as the ability to further parallelize the training workload to multiple GPUs, greatly speeds up AI advancements and makes AI more feasible to develop and use (Parallel Training Methods for AI Models: Unlocking Efficiency and Performance, 2024). According to an article on medium, AI without GPUs would be possible, but not feasible, as it would be very slow and energy inefficient (Dhanush Kandhan, 2024).

Limitations of AI

As it exists right now, AI is not a perfect technology. Current AI models work using a system of parameters that define how the input should be processed into an output. These parameters are numbers in the range of billions for the top AI models, GPT-4, an LLM by OpenAI, has over one trillion parameters (Bastian, 2023), that operate on the input, in this case text, to generate an output text. These parameters are "learned" during a process called training. Training is when the AI model is fed information, and based on this information its parameters are updated, effectively meaning that it has "learned" something from this training data (LLM Training: How It Works and 4 Key Considerations, n.d.).

Another limitation of current AI models is known as a "hallucination." Hallucinations are when an AI model presents false information as the truth, which can present issues for people looking for accurate information (The Beginner's Guide to Hallucinations in Large Language Models | Lakera – Protecting AI Teams That Disrupt the World., n.d.). One such case of AI hallucination was a lawyer who used AI in defense of a client, but the AI hallucinated prior cases that did not exist (Moran, 2023). There are, however, ways to mitigate hallucinations for certain scenarios. One such way is known as retrieval augmented generation, or RAG. RAG is when the model is able to query a database of known facts, and use this information when generating an output response (Databricks, 2023). This can increase model accuracy in regards to information in the RAG database. According to Amazon, RAG has multiple benefits (Amazon, n.d.). These include cost effective implementation, as the costs to use RAG on top of an existing model are low. The next benefit of RAG is that the information the AI model has access to can be kept up to date. The cost for updating the RAG database is low, so the information it has can always be current. The last major benefit of RAG is that it can provide sources for the information from the

RAG database, ensuring that the user knows the information is accurate, and preventing cases like the law hallucinations previously discussed (Moran, 2023).

Another way of injecting information into an AI model after its training is done is called fine tuning (OpenAI, 2024a). Fine tuning is when an AI model is partially re-trained on a specific set of data. For example, a model could be fine tuned on the code base of a specific company. According to a medium article by Refact.ai, fine tuning on a specific code base offers a few benefits, including the AI being knowledgeable about the APIs in the given codebase, or coding in a similar style to the rest of the codebase (Refact.ai, 2024).

Fine tuning and RAG can both be useful for introducing new information to an AI model after training, however according to an article on medium, RAG is typically more useful, as it is more accurate, and is easier to setup because the language model does not need to be retrained, which can be expensive (Ghosh, 2024).

What is Software Engineering

Software engineering is the engineering practice that encompasses the technical part of making software. This includes, but is not limited to, tasks such as programming, software testing, and maintenance (Michigan Tech, 2024). Modern programming is done via source code, which is most commonly mostly human readable english text, that follows certain syntax rules (Definition of Source Code, n.d.). However, there is more to software engineering than simply writing code. More than half of software engineers spend less than half the time of a workday writing code (Fum, 2022). Other tasks that software engineers work on include requirements gathering, where the engineers consult with stakeholders to determine what the software needs to

do. Software engineers also spend time in meetings with their team, or doing research into how to design their software systems.

Software jobs and AI

As mentioned above, LLMs operate using natural language, and programming, which is a part of software engineering, also is done using mostly-natural language. A next question would therefore be to ask, can LLMs program, and if so, how well can they do it? The answer to the first question is yes, LLM models can write code (LLMs for Code Generation: A Summary of the Research on Quality, n.d.). The next question is a bit more complicated to answer.

To gain an insight into how well they can program, we can look into various AI model benchmarks, which are created to test how well various AI models perform at certain tasks. Leetcode is a website with thousands of programming problems, with the goal of allowing programmers to practice their skills at coding (LeetCode, 2021). To examine the effectiveness of AI at leetcode problems, we look at a study by Coignion et. al. in 2024 (Coignion & Quinton, 2024). Right off the bat, the study finds that LLMs are typically better than people at solving Leetcode problems. We can also look at the pass rates of various models, specifically we examine the "pass@10" rate, meaning that the model is given 10 chances to generate a solution to pass the Leetcode problem, which is relevant given how the outputs of the same AI model can vary even with the same input. The study finds that the highest pass@10 rate is 20.6% by CodeLlama-13B-instruct, and that only two models have a pass@10 rate above 20%. The pass@1 rates are even worse, with the StarCoder model having a passing rate of only 9.5% on the first trial. This study also examined the performance of the LLM generated solutions compared to humans. On LeetCode, there is a ranking system, where each correct solution to a

problem is timed, and compared to all other correct solutions submitted to that problem. Coignion et. al. 's work found that the language model CodeGen-6B-mono had a mean ranking of 73%, meaning that on average the model's solution was faster than 73% of the other solutions. Using the paper's assumption that the other solutions were mostly made by humans rather than by other language models, LLMs are on average able to generate faster code to leetcode problems than people.

However, this benchmark is not perfect, as leetcode problems do not necessarily match actual problems faced in the daily lives of software engineers, but rather leetcode is primarily designed to help pass technical coding interviews (LeetCode, 2021). To get a better understanding of how AI can perform at typical software engineering use cases, we can look at SWE-bench, a benchmark of AI language models on actual software engineering problems from github by Jimenez et. al. in 2024 (Jimenez et al., 2024). Specifically, we look at their benchmark leaderboard of the full github dataset, with 2294 problems from actual github repositories. All experiments were done in pass@1, meaning the AI model only had one opportunity to generate a correct solution. The best performing model was "OpenHands + CodeAct v2.1 (claude-3-5-sonnet-20241022)", with a pass rate just shy of 30%. One interesting thing to note is that this variation of Claude 3.5 has almost 14x better results at this benchmark than one of its predecessors, Claude 2, of which the two models were only released less than a year apart from one another (Claude 2, 2023; Introducing Claude 3.5 Sonnet, 2024; Jimenez et al., 2024).

We can also look into the case study of Devin, "the first AI software engineer" (cognition.ai, 2024). Devin is an AI system that was marketed as if it could do much of the work of modern software engineers. In their article on the release of Devin, Cognition AI, the creator of Devin, made many bold claims about the capabilities of Devin. They advertised that Devin

could “build and deploy apps end to end”, or “contribute to mature production repositories” (cognition.ai, 2024). The demo also had a video of Devin completing software engineering tasks on Upwork, a contracting website (Upwork, 2024). While these claims might seem bad for human software engineers initially, it was later discovered that the demo of Devin on Upwork was exaggerated (Monge, 2024). According to an article in 2024, Devin did not successfully complete the jobs on Upwork, instead doing less complex, irrelevant jobs, which were “falsely represented as significant achievements” (Monge, 2024).

Data

Some companies are already beginning to implement the use of AI in their jobs to complete programming tasks. For example, Google, a major software company, has stated publicly that more than 25% of its code is written by AI (McKenna, 2024). According to the same Fortune article by McKenna Google also plans further use of AI in their code writing. Microsoft, another large tech company with hands in the AI field, has also started to use AI, specifically their copilot system, internally. According to a study by Microsoft, 70% of people in the survey said they were more productive at work when using copilot (Fleck, 2025).

I also consulted some people in the software field to discover how they have/have not been using AI in their jobs. One of the respondents said they use ChatGPT in their employment, and that they use the free version. They said that they mainly use the AI system for creating and refining ideas. This same person also said that they use ChatGPT more now than when it first came out, as they have gotten the chance to become more familiar with it. This person has also observed their coworkers using LLMs at their job, however all of the uses they have seen is to

generate ideas, rather than copying and pasting the output of the AI models. Finally, this respondent does not use generative AI outside of their job.

Another respondent said that they do use generative AI in their programming job, and that they specifically use ChatGPT-4o. This person also uses the same ChatGPT-4o model for writing outside of their job.

The final respondent worked at Google developing software. This person used Gemini, Google's AI model, internally while working at Google. They used Gemini for debugging code, as well as for help in writing repeated code. This person also answered that they have become more efficient at using generative AI in their job, as they have become better at writing prompts for it. Outside of their job, this interviewee uses both Gemini and ChatGPT for coding projects. They use the models for debugging, small pieces of code, and front end development.

Analysis

To gain an idea as to how all of these ideas come together, we will use Latour's Actor Network Theory to analyze this system (Nickerson, 2024). Actor Network Theory is the idea that systems exist as a set of different actors, with various relationships to each other, forming a network. In our scenario of AI and Software Engineering, there are a few different actors. The first group of actors are the companies that produce software. The next group of actors are individual software engineers. The third group of actors will be companies that create internal tooling for software developers. The final set of actors are the businesses and/or individuals that produce various AI models.

We can now examine the interesting relationships between each actor in this network and the other actors in this network. We will first take a look into the relationship between software

producing tech companies, and all of the other actors. As for the software producing companies and the individual software engineers, they currently need these software engineers to write their code. AI has not yet fully replaced programmers, as even big tech companies such as Google still have human programmers write almost 75% of their code (McKenna, 2024).

Now we shall look at the relationship between the software companies and the producers of AI. The first thing we can note is that in most/all cases, companies that produce AI are also general producers of software. Examples include Google, producer of Gemini (Google Deepmind, 2024), as well as other software, or Microsoft, creator of Copilot and additionally other software (Spataro, 2025). Despite the significant overlaps between these two actor categories, we will treat them as distinct groups for purposes of this Actor Network Theory analysis. For their relationship, software companies would ideally want the AI companies to build better, faster, cheaper AIs, so that they can use AI to develop programs more cost effectively than human programmers.

Additionally, the software engineers themselves are a major player in the network. The software engineers have a very strong relationship with the companies that produce software. The software engineers are employed by such companies, so they require them for their livelihood. If the software producing companies were to no longer need software engineers, be it due to AI or something else, this would be detrimental to the software engineers. They would likely need to find either another company, or another career entirely if all companies replace software engineers.

Next we look at the relationship between the software engineers and the creators of tooling for software engineers. The software engineers will want the creators of tooling to make tools that work just well enough to help them and to make their lives easier, but not well enough

to replace them. If the tooling companies make tools that enable the software engineers to do their job more efficiently, then it may make their lives easier. However, if it is too good, then it may replace them, or it may cause them to have to complete more work overall now that they are more efficient.

As for individual software engineers and the creators of AI models, the software engineers have a vested interest in the AI models not becoming too effective. This is because if the AI models become so proficient at coding that they can fulfill the job of the software engineers, then they have the potential to have their jobs replaced, which would not be good for them.

The next actor that we explore is the category of companies that create internal software development tooling. Specifically we are looking at companies that are not primarily AI focused, such as Jetbrains (JetBrains: Developer Tools for Professionals and Teams, n.d.). Companies such as jetbrains will want to sell their tooling software to producers of software. Jetbrains for example sells a business tier of their coding integrated development environments (IDEs) for up to $35 per month per session. Selling this tooling is a way that companies like this can profit, so if all software becomes done by fully autonomous AI, then they will no longer be able to sell tools aimed at helping developers.

We next look into the relationship between non-AI tooling companies and individual software engineers. Jetbrains also has software for individual developers (JetBrains: Essential Tools for Software Developers and Teams, n.d.), so they will be incentivized to create tooling that developers will want to use.

Examining the network between tooling companies and AI companies, we can see that there is a natural difference between the goals of these two actors. Non-AI tooling companies

will want most programming to still be done by human developers, so they can sell their tools for developers, which directly contradicts the AI companies' goal of selling more AI.

The creators of AI are a collective of actors that spans multiple large companies, including Google with Gemini (Google Deepmind, 2024), Microsoft with copilot (Spataro, 2025), or Anthropic with Claude (Introducing Claude 3.5 Sonnet, 2024; Spataro, 2025). This group of actors will want the software producing companies to buy their AI, so they can make a profit. ChatGPT, for example, offers a $200/mo "pro" subscription, which is likely targeted at businesses (Introducing ChatGPT Pro, 2024). It is likely that in their ideal world, their AIs will be used for all work, with the CEO of OpenAI, Sam Altman, even recently saying that we will reach this point, dubbed the "singularity", in 2025 (Eliot, 2025).

Next we look at the relationship between these creators of AI and individual software engineers. These AI creators want their AI models to reign supreme, and to be better than the software engineers. One thing to note is that these AI companies also need software engineers to build their AI models, which brings up an interesting question of if the engineers at these companies are working to put themselves out of jobs. However, that is a question that is beyond the scope of this paper.

Finally, we look at the relationship between the creators of AI and the creators of conventional software engineering tooling. The AI creators will want to replace convention software development, thus invalidating the business model of conventional tooling companies.

Discussion

As it stands today, AI has not fully replaced software engineers. However, there are many different companies with very talented people that are actively pushing the boundaries of what is

possible using Large Language Models in coding, working to create a world where AI can do the work of a software engineer. A scenario where AI fully replaces software engineers would likely be a technological singularity, where AI surpasses human capabilities and can do all the tasks that humans can do (Mucci, 2024). This AI would be known as an Artificial Super Intelligence, or ASI (IBM, 2023), which OpenAI CEO Sam Altman believes we are close to reaching (Barlow, 2025). However, we must be skeptical of Sam Altman's claims, as he has a direct motivation to upsell the capabilities of his AI systems in order to get more people to buy it. Another outspoken AI advocate and Google engineer Ray Kurzweil, believes that we will not reach an AI that surpasses human intelligence until the year 2045 (Kurzweil, 2022). As of right now, no one can really predict when or even if AI will ever surpass people, all we can conclude is that today, at the time of writing, it is not, and people are still needed for most programming roles.

Conclusion

There are many different possible ways that AI could end up influencing the field of software engineering and the world as a whole in the long term. One way would be for AI models to completely replace human software engineers. This way would be detrimental to the human software engineers that currently rely on their software engineering jobs as their livelihood, but would likely be profitable for both the companies that produce software, and especially so for the companies that produce said AI models. The seemingly most likely result, however, is that AI will work in tandem with human software developers, and this would be the best outcome for computer science majors such as myself.

References

Amazon. (n.d.). *What is RAG? - Retrieval-Augmented Generation Explained - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/what-is/retrieval-augmented-generation/

AWS. (2024). *What is Artificial Intelligence? - Artificial Intelligence (AI) Explained - AWS*. Amazon Web Services, Inc. https://aws.amazon.com/what-is/artificial-intelligence/

Barlow, G. (2025, January 13). *Sam Altman predicts artificial superintelligence (AGI) will happen this year*. TechRadar. https://www.techradar.com/computing/artificial-intelligence/sam-altman-predicts-artificial-superintelligence-agi-will-happen-this-year

Bastian, M. (2023, March 25). *GPT-4 has a trillion parameters - Report*. THE DECODER. https://the-decoder.com/gpt-4-has-a-trillion-parameters/

Claude 2. (2023, July 11). Www.anthropic.com. https://www.anthropic.com/news/claude-2

Cloudflare. (2022). *What is a large language model (LLM)?* Cloudflare.com. https://www.cloudflare.com/learning/ai/what-is-large-language-model/

cognition.ai. (2024, March 12). *Cognition | Introducing Devin, the first AI software engineer*. Cognition.ai. https://www.cognition.ai/blog/introducing-devin

Coignion, T., & Quinton, C. (2024). *A Performance Study of LLM-Generated Code on Leetcode*. Arxiv.org. https://arxiv.org/html/2407.21579v1

cuBLAS. (n.d.). Docs.nvidia.com. https://docs.nvidia.com/cuda/cublas/

Databricks. (2023, October 18). *Retrieval Augmented Generation*. Databricks. https://www.databricks.com/glossary/retrieval-augmented-generation-rag

DeepSeek. (n.d.). Www.deepseek.com. https://www.deepseek.com/

Definition of source code. (n.d.). PCMAG. https://www.pcmag.com/encyclopedia/term/source-code

Dhanush Kandhan. (2024, November 19). *Do You Really Need a GPU for AI Models? The Truth, Hardware Needs, and Deployment Insights*. Medium. https://itzmedhanu.medium.com/do-you-really-need-a-gpu-for-ai-models-the-truth-hardware-needs-and-deployment-insights-37b650adfb91

Eliot, L. (2025, January 8). Sam Altman Stirs Mighty Waves With Tweets Of AI Singularity Staring Us In The Face. *Forbes*. https://www.forbes.com/sites/lanceeliot/2025/01/08/sam-altman-stirs-mighty-waves-with-tweets-of-ai-singularity-staring-us-in-the-face/

Fitch, S. (2024, March 8). *The Surprising Power of Next Word Prediction: Large Language Models Explained, Part 1*. Center for Security and Emerging Technology. https://cset.georgetown.edu/article/the-surprising-power-of-next-word-prediction-large-language-models-explained-part-1/

Fleck, A. (2025, February 6). *How we're tackling Microsoft 365 Copilot governance internally at Microsoft - Inside Track Blog*. Inside Track Blog. https://www.microsoft.com/insidetrack/blog/how-were-tackling-microsoft-365-copilot-governance-internally-at-microsoft/

Fum. (2022, February 9). *10 Essential Things Software Engineers Do Apart From Coding -*. Inspirezone.tech. https://inspirezone.tech/things-software-engineers-do-apart-from-coding/

Ghosh, B. (2024, February 25). *When to Apply RAG vs Fine-Tuning*. Medium. https://medium.com/@bijit211987/when-to-apply-rag-vs-fine-tuning-90a34e7d6d25

Google Deepmind. (2024). *Gemini - Google DeepMind*. Deepmind.google. https://deepmind.google/technologies/gemini/

IBM. (2023, December 14). *Artificial Superintelligence*. Ibm.com. https://www.ibm.com/think/topics/artificial-superintelligence

IBM. (2024, December 17). *LLM temperature*. Ibm.com. https://www.ibm.com/think/topics/llm-temperature

IBM watsonx as a Service. (2024, November 27). Ibm.com. https://www.ibm.com/docs/en/watsonx/saas?topic=solutions-tokens

Introducing ChatGPT Pro. (2024, December 5). Openai.com. https://openai.com/index/introducing-chatgpt-pro/

Introducing Claude 3.5 Sonnet. (2024, June 20). Www.anthropic.com. https://www.anthropic.com/news/claude-3-5-sonnet

JetBrains: Developer Tools for Professionals and Teams. (n.d.). JetBrains. https://www.jetbrains.com/

JetBrains: Essential tools for software developers and teams. (n.d.). JetBrains. https://www.jetbrains.com/#for-developers

Jimenez, C., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). *SWE-bench*. Www.swebench.com. https://www.swebench.com/

Kurzweil, R. (2022). The Singularity Is Nearer. Random House Large Print.

LeetCode. (2021). *LeetCode - The World's Leading Online Programming Learning Platform*. Leetcode.com; LeetCode. https://leetcode.com/

LLM Training: How It Works and 4 Key Considerations. (n.d.). Www.run.ai. https://www.run.ai/guides/machine-learning-engineering/llm-training

LLMs for Code Generation: A summary of the research on quality. (n.d.). Www.sonarsource.com. https://www.sonarsource.com/learn/llm-code-generation/

McKenna, G. (2024, October 30). *Over 25% of Google's code is now written by AI—and CEO Sundar Pichai says it's just the start*. Fortune. https://fortune.com/2024/10/30/googles-code-ai-sundar-pichai/

Michigan Tech. (2024). *What is Software Engineering? | Computer Science | Michigan Tech*. Michigan Technological University. https://www.mtu.edu/cs/undergraduate/software/what/

Monge, J. (2024). *Devin's Demo As The "First AI Software Engineer" Was Faked*. Zeniteq.com. https://www.zeniteq.com/blog/devins-demo-as-the-first-ai-software-engineer-was-faked

Moran, L. (2023, May 30). *Lawyer cites fake cases generated by ChatGPT in legal brief*. Legal Dive. https://www.legaldive.com/news/chatgpt-fake-legal-cases-generative-ai-hallucinations/651557/

Mucci, T. (2024, June 5). *What is the Techological Singularity? | IBM*. Www.ibm.com. https://www.ibm.com/think/topics/technological-singularity

Nickerson, C. (2024, February 13). *Latour's Actor Network Theory*. Simply Psychology. https://www.simplypsychology.org/actor-network-theory.html

OpenAI. (2022, November 30). *ChatGPT*. Openai.com. https://openai.com/index/chatgpt/

OpenAI. (2024a). *OpenAI API*. Platform.openai.com. https://platform.openai.com/docs/guides/fine-tuning

OpenAI. (2024b). *What Are Tokens and How to Count them?* Help.openai.com. https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them

OpenAI API. (n.d.). Platform.openai.com. https://platform.openai.com/tokenizer

Parallel Training Methods for AI Models: Unlocking Efficiency and Performance. (2024). Sapien.io. https://www.sapien.io/blog/parallel-training-methods-for-ai-models-unlocking-efficiency-and-performance

Pichai, S., & Hassabis, D. (2023, December 6). *Introducing Gemini: our largest and most capable AI model*. Google. https://blog.google/technology/ai/google-gemini-ai/

Refact.ai. (2024, April 16). *LLM Fine-Tuning: Personalizing Your Code Suggestions*. Medium. https://medium.com/@refact_ai/lll-fine-tuning-personalizing-your-code-suggestions-d22d5d58f988

Spataro, J. (2025, January 15). *Copilot for all: Introducing Microsoft 365 Copilot Chat | Microsoft 365 Blog*. Microsoft 365 Blog. https://www.microsoft.com/en-us/microsoft-365/blog/2025/01/15/copilot-for-all-introducing-microsoft-365-copilot-chat/

The Beginner's Guide to Hallucinations in Large Language Models | Lakera – Protecting AI teams that disrupt the world. (n.d.). Www.lakera.ai. https://www.lakera.ai/blog/guide-to-hallucinations-in-large-language-models

University of Illinois Chicago. (2024, May 7). *What is (AI) Artificial Intelligence?* University of Illinois Chicago. https://meng.uic.edu/news-stories/ai-artificial-intelligence-what-is-the-definition-of-ai-and-how-does-ai-work/

Upwork. (2024). *Upwork | Hire Freelancers. Make things happen.* Upwork. https://www.upwork.com/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. ArXiv. https://arxiv.org/abs/1706.03762