Research Article

From Augmentation to Delegation: AI Coding Assistants and the Redistribution of Cognitive Labor in Software Development

Vikas Jain[1], **Ashish Kumar Jain**[1]

1. Institute of Engineering & Technology Devi Ahilya Vishwavidyalaya, India

AI coding assistants are revolutionizing the way in which software is created. This is because these AI assistants are changing the nature of the coding process from an exclusively mental activity to one in which humans and computers work together in tandem. While much has been written on the productivity gains that these AI coding assistants offer, little attention has been given to the way in which these tools affect the way in which the mind is engaged, professionalism is practiced, and skills are developed. This paper provides a framework to understand the way in which the cognitive engagement with AI coding assistants varies across the autonomy continuum from low autonomy assistants to high autonomy assistants. It examines the four levels of low autonomy assistants, conversational coding assistants, AI-native development environments, and high autonomy coding assistants in relation to the way in which the mind is engaged, professionalism is practiced, and skills are developed. This paper is guided by the theories of automation, cognitive offloading, and socio-technical change and argues that the way in which the mind is engaged with AI coding assistants from low to high autonomy is from the builder to the evaluator and supervisor of the coding process, while professionalism is practiced and transformed in the process, albeit in an evolved manner.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

1. Introduction

It is obvious that AI is becoming an integral part of the digital workspace. Take coding helpers, for example. These help in coding, explaining what the code is doing currently, explaining and doing system

architecture, and even creating large amounts of code on their own. The interesting thing is that they're doing jobs that require human thought, so they're not just any ordinary coding utilities.This whole thing with AI coding tools is really part of something bigger that's happening, where we're splitting up the thinking work between people and computers. It's not merely a question of efficiency; it's a question of the distribution of cognitive work between humans and machines [1].

Everyone in tech keeps going on about how productive and efficient AI coding tools are, but we actually don't know much about the underlying cognitive and socio-technical consequences of AI coding assistants. Programming has been traditionally benefited from social dynamics of joint discussion, reasoning, and continuous problem-solving efforts. However, if AI assistants alter the locus of decision-making and reduce the mental requirements for 'cognitive crunching,' they disrupt the established patterns of the way we work.

This study creates a detailed conceptual framework that places AI coding helpers on a spectrum of augmentation and delegation. Rather than exploring common questions of efficiency or performance, it investigates how the level of autonomy afforded to AI coding assistants shapes the professional identity of software developers and their ways of thinking and social interactions. By placing AI coding assistants within broader debates about automation and trust in digital work, it helps answer questions about what becomes of human expertise in a world of pervasive algorithms.

2. The Emergence of AI Coding Assistants

The development and use of large language models (LLMs) and their training on big text and programming data is closely related to the development of AI coding assistants. Modern coding assistants use probabilistic pattern recognition to provide code suggestions that are both syntactically and contextually correct, moving beyond the rule-based code development tools used in the past [2]. By predicting the next code input, these AI assistants treat code as if it is a language.

There is also the general trend in AI development to see foundation models that can reason across multiple domains and now appear in the realm of generative AI for software development [3]. So, AI coding helpers are not fixed tools but ones that can now converse and collaborate with the developer.

Various research has also shown that the developers do not simply use these AI tools to speed up the process but also to aid in the process of brainstorming, thinking and exploring the possibilities [4]. So, the

process is not simply linear but iterative and dialogic in nature. What this also brings up is the question of agency and authorship in the realm of coding.

There has also been the evolution of development environments to move from simple text editors to environments with version control systems and debugging tools. The latest evolution is the AI coding helper and the integration of generative AI [5]. So, the evolution here is deterministic assistance to probabilistic and collaborative assistance.

There is also the research done on the realm of workplace automation and the fact that the improvement in automation does not simply replace the human but changes the allocation of the task. So, in the realm of software development, the AI tool is now assisting the developer in code generation and memory recall [6], and the developer is now evaluating the results provided by the tool. So, this is the augmentation continuum of automation.

3. Taxonomy of AI Coding Tools

Coding Assistants offered by AI are generally grouped together as a single category in the world of technology. However, there are a lot of variations in the amount of autonomy that coding assistants offer and they do, the way in which we are interacting with them, and the amount of cognitive load that they handle. If we treat them as a single, interchangeable category, we miss the important role that they play in the coding process.

To highlight the differences that exist among coding assistants, a taxonomy will be proposed in the following sections based on the following axes:

Level of Autonomy: the amount of independence that the system can offer in the coding process.

Interaction Mode: the way in which the system assists the human in the coding process.

Cognitive Role: the way in which the system assists the human in the coding process.

This method of categorizing the system also relates to the overall results of automation. Multiple studies have shown that people react to varying levels of autonomy of machines and agents in different ways [7]. It also explores the ways people make judgments on computer systems, where readers make a distinction between technology that helps them and technology that replaces them [8].

3.1. AI Coding Assistants Categories

Inline autocomplete tools tightly hold the leash: they provide only minor suggestions—maybe a single word or a single line of code. The programmer still controls the structure and logic of the code, so the tool's assistance is useful but not very creative. Chat-based coding assistants have more freedom to roam, using dialogue to assist the programmer. You describe the purpose of the code in simple language, and the tool generates the code. You remain firmly in the driver's seat, but the AI may direct the way the problem is defined and the way the solution is developed, which may lead to trust or misinterpretation—the code may seem brilliant but not necessarily understood. AI-native code editors have a wider view of the project, seeing the entire project rather than focusing on a single file of code. The Fig. 1. below is for reference to understand the increasing autonomy with reference to greater cognitive delegation. They may suggest significant changes to the code on a grand scale, looking at many files and even suggesting improvements to the architecture of the project, which may lead to the adoption of the AI's design instincts. Autonomous coding agents go further than the others, with the highest independence: they may handle complex tasks, work on many files of code, test the code, and make changes to the code with minimal supervision from the programmer. Humans may be present to oversee the work but do not necessarily create the code themselves, which may lead to the erosion of the ability to problem-solve alone.

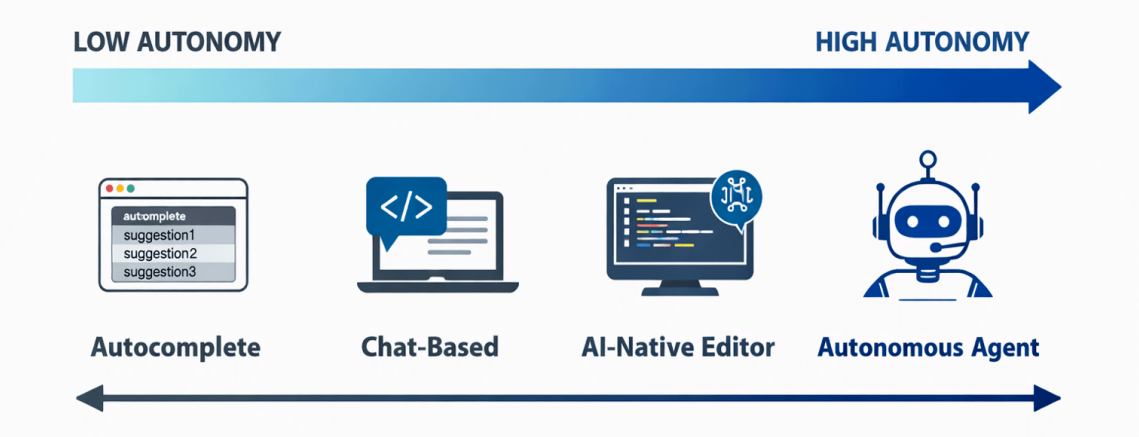

Figure 1. Continuum of AI autonomy across categories of coding assistants. Increasing autonomy corresponds to greater cognitive delegation.

doi.org/10.32388/EY1G5Q

3.2. Autonomy and Human Role Reconfiguration

As the autonomy increases, the role of the human changes from actually doing the code to mostly observing and overseeing the operation of the system. This is supported by studies on automation, which have shown that the more the system is left to operate on its own, the less the human does and the more the human observes what the system is doing [9]. While the tools with low autonomy require the programmer to be the actual problem solvers, tools with high autonomy may actually spend more time checking and reviewing results rather than actually creating everything from scratch. This may have an impact on the development of skills over time, which is particularly important for new developers. This taxonomy provides a simple way of understanding the cognitive and social implications of AI coding tools, which goes beyond the common assessment of the tools in relation to how much faster they allow humans to work.

4. Human–AI Collaboration and Cognitive Redistribution

This can be seen as a cognitive redistribution process, in the sense that it is a transfer of some cognitive activities from the person to the technology. Cognitive offloading is a research area that has shown that people mostly rely on external resources, such as notes or search engines, to reduce cognitive load [10]. It can help to speed up the process and reduce the load on the brain, but it can also have a long-term effect on the development of memory, reasoning, and problem-solving abilities [11]. However, in the case of AI coding assistants, it is more than that, as it does not merely store or retrieve information, but actually produces the solution. It is different from the older development tools that carried out the exact instructions given to them by humans. It uses a system of “probabilistic reasoning” to produce the solution. It can be seen as a form of “extended cognition,” in the sense that it includes the human and the machine as a cognitive system [12].

This sharing of cognition, however, doesn’t imply equality of roles between humans and AI systems. As the level of automation rises, humans have a tendency to move from directly addressing the problem to more of a supervisory position, approving the outcome of the work done by the machine. Research on human-automation interaction has demonstrated that as the level of automation rises, users of the system can become unaware of the situation as well as develop a lack of skills if they are less actively engaged with the machine [13]. In programming, this means a less effective understanding of the underlying logic of the program if the solutions have been almost entirely generated by the AI system

itself. Other research on human-AI interaction has shown that highly developed AI systems act as a partner to the human, as opposed to a tool, as seen in the case of human-automation interaction [14]. This has been demonstrated in the case of conversational coding assistants, where the human refines the prompt, directs the output, and influences the final code itself through a dialogue with the machine. This, however, isn't a form of teamwork, as the intentions of the machine aren't shared as they would be in the case of humans. We can understand this AI autonomy redistribution towards supervisory evaluation by the following Fig. 2 below.

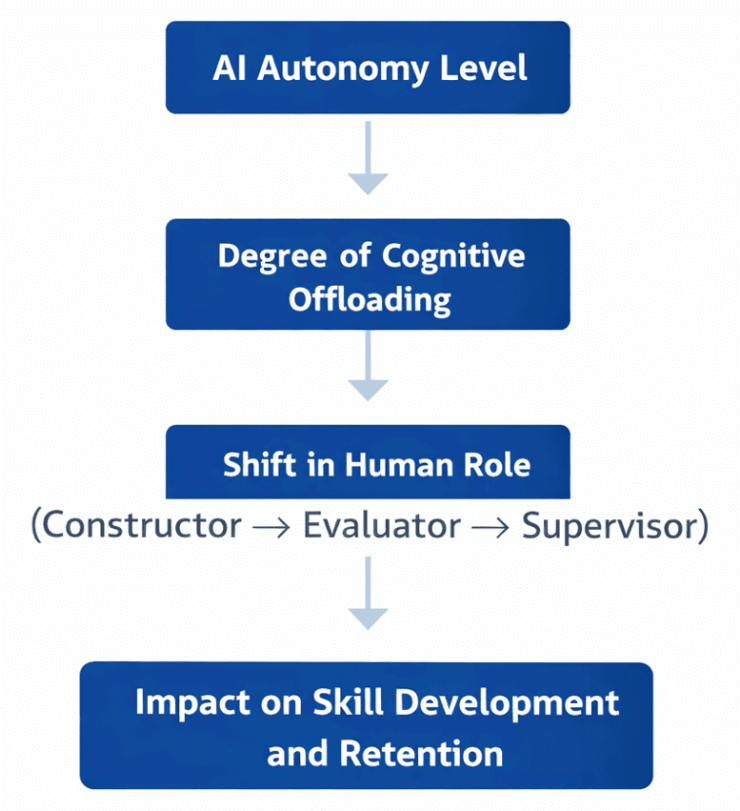

Figure 2. Increasing AI autonomy redistributes cognitive effort from active construction toward supervisory evaluation, influencing long-term skill formation.

This Cognitive Redistribution Model assists in understanding the process in which the increase in the autonomy of AI causes a shift in the cognitive activities that humans are involved in, ranging from builder to collaborator to overseer. The coding help offered by AI assistants does not ends cognitive activities; it merely shifts them. If there is a low level of autonomy, the programmers will be responsible

for coding the program themselves. If there is a high level of autonomy, the programmers will merely oversee the results produced by the AI coding helper. This is similar to the trend in other algorithm-driven activities [15], in which humans are more likely to oversee the results produced by the AI system rather than actually working on the activity themselves. Thus, it can be seen that there is a shift in the skill that programmers require to be proficient in, as it is more related to the oversight and understanding of the results produced by the AI system, which requires a high level of reasoning [16]. However, it can also be seen that there is a shift in the activities that programmers can focus on as a result of the AI coding helper, in which they can focus more on the design and conceptual activities.

5. Programming Work in Transition

The emergence of AI coding assistants is revolutionizing the way in which software is being built. Programming is no longer considered to be an art that is based on meticulous thinking, strong logic, and constant practice—practicing and debugging to get familiar with the landscape. Programming skills are acquired over time through practical experience and the development of software systems. However, with the emergence of generative AI, the way in which this is done is changing, and the work of writing the code is slowly moving from the programmer to the AI assistant. As research on the nature of digital labor has been studied out, automation does not eliminate tasks but rather reorganizes them [17]. Workers move from the production side to the management side of the production process. Similarly, in the case of programming, the emergence of AI has led to the programmer moving from the production side to the management side of the production process, as the AI spits out the code and the programmer is left to review and merge the AI-generated parts of the code. We can see the various levels of AI Autonomy in Fig. 3 below.

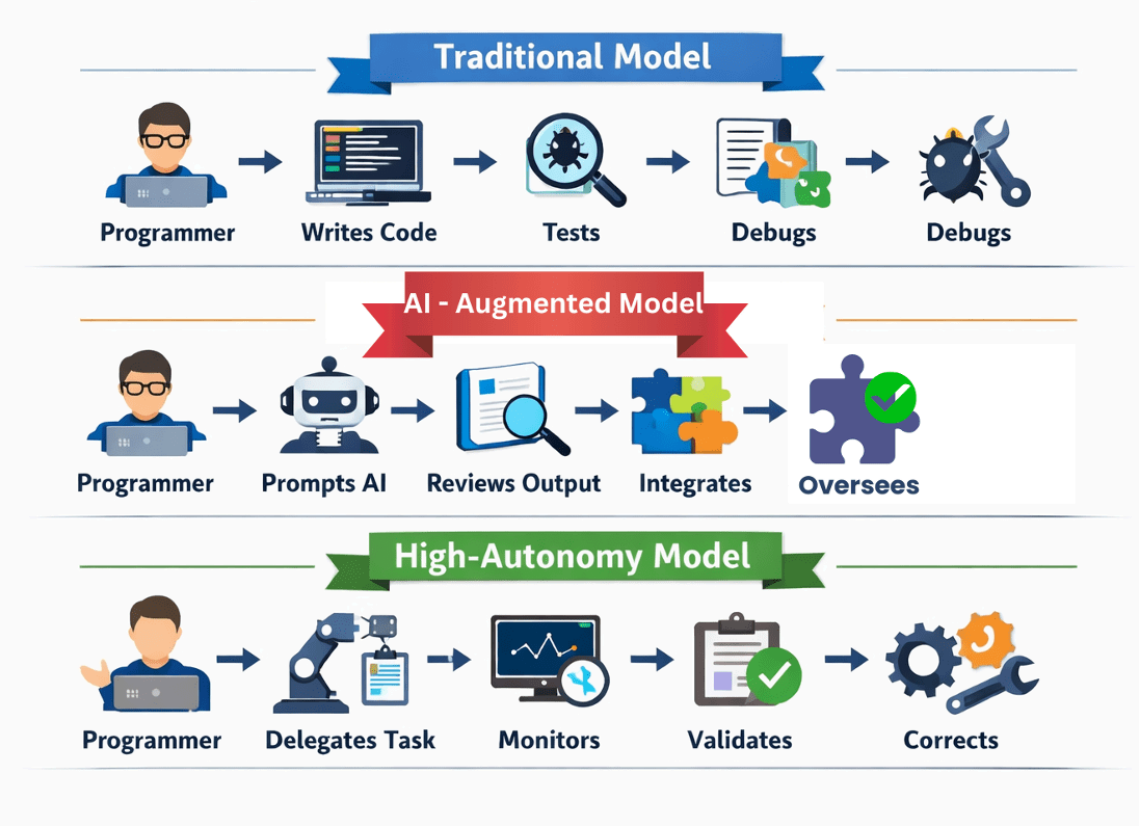

Figure 3. Evolution of programming roles across increasing levels of AI autonomy. Human activity shifts from direct construction toward supervision and validation.

The programmer is no longer the sole creator of the code and is now forced to validate and merge the AI-generated ideas and concepts in the code. This is not the only area in which this is happening, and generative AI tools are forcing workers in various fields to move from the production side to the management side of the production process [18]. This has also led to the raising of several important issues regarding ownership and accountability. Even though the programmer is still accountable for the final product, the question of accountability is complicated in the sense that it is not clear which parts of the logic have been generated by the programmer and which parts have been generated by the AI assistant. Software engineering has always been considered to be an art that is based on strong knowledge of syntax, logic, and system development [19]. However, as research on algorithmic management has pointed out, the development of intelligent machines in the management process is likely to delete the lines of skill and autonomy.

Yet this change will not affect all programmers equally; while veteran programmers may zoom ahead with the aid of AI tools, getting more done in less time [20], others may rely on these tools as crutches to learn the fundamentals of programming. This could have potentially change the way programming expertise is developed and challenge traditional approaches to becoming a programmer. Productivity tools like AI coding assistants are often marketed to companies, but the aforementioned research on automation reveals that increased productivity often brings new forms of dependency and shifts in the skills that programmers need to be effective. For instance, programmers may have more time to test and integrate the output of AI tools rather than simply typing away to produce code themselves [21].

Moreover, AI coding assistants do not replace the need for human thought, they simply shift and change where that mental effort is focused. For instance, while programming itself may be less of a mental task, the supervision of the output of AI tools becomes a prominent feature of programming work. This phenomenon is not unique to AI coding assistants; rather, it reflects a broader trend of the way that smart technology is changing the way that expertise is developed [22]. By looking at AI coding assistants through this lens, we move beyond the simple question of productivity to more interesting questions of what this phenomenon reveals about the nature of programming work and the role of the programmer themselves.

6. Autonomy, Trust, and Dependency

With the rise of autonomy in coding tools, the issue of trust and dependence becomes a larger concern. Software Developers might change their behavior and use case from actively reasoning through things to passively accepting whatever the tool suggests to them [23]. Research has shown that the level of trust varies, depending on the context, but the right amount of it facilitates better collaboration [24], whereas too much of it can result in automation bias, wherein the suggestions of the tool are taken at face value without proper verification. This is a concern since the suggestions of the tool, though fluent, are only probabilistic [25], which means that the fact that the suggestions are coherent does not imply that they are correct. The more autonomous tool, the higher the chance of dependence [26], which has been shown to result in a decline in skills if the tool is used over time, especially if programmer is new in field [27]. On the other hand, the tool can be a form of scaffolding, which allows higher-level reasoning to be lifted. Accountability is a larger issue, as the programmer remains accountable for the code [28], even if the

origins of the decision-making process are unclear, whereas the reputation of the institution can result in uncritical use of the tool if it has become popular [29].

7. Educational and Professional Implications

The exponential increasing use of AI coding assistants is transforming the way study, understanding and professional development in programming are done. The fact that true understanding and proficiency in programming can only be achieved through continuous reasoning and debugging means that the use of AI coding assistants could potentially alter the way education and professional development in programming are done. Research in computer education has shown that the key to effective and profound understanding and knowledge in programming is the level of engagement and the ability to break down problems into manageable parts [30]. However, the use of AI coding assistants could potentially alter this and lead to the loss of the productive struggle required to fully comprehend programming. Nevertheless, the use of AI coding assistants also has the advantage of making programming easier and increasing the level of confidence and entry points for new programmers. However, the key issue for educators and trainers to address is the balance to be struck between the use of AI coding assistants and the delegation of responsibility to the students [31]. AI coding assistants could potentially act as the much-needed scaffold to keep the students engaged and to increase their understanding and knowledge of programming. However, use of AI coding assistants could potentially change the way education and professional development in programming are done. In the context of professional development in programming, the use of AI coding assistants has altered the way programming is done and the skills required to do programming effectively [32]. The Fig. 4 is showing below the use of AI coding assistants across augmentation and delegation modes for better understandings. The key skills required in the context of AI coding assistants include the ability to design the prompt, the ability to evaluate AI code effectively, and the ability to supervise the architecture. These skills can also be seen to represent the much broader changes in the way work is done and the skills required to do work effectively in the current digital context [33].

Figure 4. Educational and professional integration of AI coding assistants across augmentation and delegation modes.

8. Ethical Considerations

The question of ethics comes into play with the advent of AI coding assistants, where the question of who to give credit to, who to blame, and the transparency of the entire issue comes into question. With AI programming, there is confusion between the role of the machine and the role of the programmer,

making it difficult to give credit to the right person or to assess the work of the programmer in the classroom or other settings. When the AI program makes mistakes, the blame still rests on the programmer; however, the more autonomy that AI possesses, the less verification is required, making it more important to have good documentation [34]. With the advent of AI programming, there is also the issue of transparency, where many AI programs are black boxes; however, having a program that is transparent helps to keep the issue of ethics alive [35]. With the advent of cloud computing, there is the issue of privacy and security of proprietary code [36], however, there is also the issue of biases that may be embedded into the programming, where certain coding methods may be favored over others [37], which may create more disparity between people who have access to the technology and those who do not [38].

9. Future Research Directions

The swift development of AI coding assistants offers a lot of avenues to be explored in the future, as the impact of these tools has yet to be completely understood. There's a need to examine the effect of extensive use on the development of skills, depending on the level of autonomy of the tool, such as conceptual, debugging, and architectural skills. There's a need to examine the social implications of the use of these tools, such as accountability, trust, and cooperation, among developers using these tools in day to day tasks. There is a need to study and examine the educational implications of the use of these tools, such as the best way to incorporate these tools without compromising the fundamental skills, such as restricted, controlled, and open use of the tools. There's a need to examine the technical implications of the use of these tools, such as the need to improve explainability, verification, and confidence, to avoid over-reliance on the tools, as well as the need to conduct interdisciplinary investigations that incorporate software engineering, cognitive science, sociology, and ethics of artificial intelligence to better comprehend the socio-technical and professional implications of the use of these tools.

10. Conclusion

There is changing the face of software development as we see AI coding assistants entering the world and use case of code creation, solution building, and our work execution processes. From a coder's perspective, we're no longer looking at a individual operation but a collaborative approach involving both human beings and AI assistants. The study outlines a framework for AI coding assistants as a spectrum of augmentation and delegation. This proves the idea that cognitive work is not ending but merely transformed from one entity to another. At one end of the spectrum, we still have systems requiring

human reasoning despite their low autonomy level, while on the other end, we have AI assistants requiring programmers to be in a supervisory role. These changes extend beyond programmers' efficiency but also influence their learning process, trust, accountability, and their professional identity as programmers.Organizations should integrate AI assistants into their systems in a thoughtful manner, ensuring they not only increase their efficiency but also provide a platform for mental engagement. At the end of it all, AI coding assistants do not replace programmers/ engineers/ developers but change the programming environment itself.

References

1. ^*Gillespie T (2014). "The Relevance of Algorithms." In: Media Technologies: Essays on Communication, Mat eriality, and Society. MIT Press.*
2. ^*Brown T et al. (2020). "Language Models Are Few-Shot Learners." Adv Neural Inf Process Syst. **33**:1877–19 01.*
3. ^*Bommasani R et al. (2021). "On the Opportunities and Risks of Foundation Models." Stanford Center for Re search on Foundation Models Report.*
4. ^*Barke S, James MB, Polikarpova N (2023). "Grounded Copilot: How Programmers Interact With Code-Gene rating Models." Proc ACM Program Lang. **7**.*
5. ^*Autor D (2015). "Why Are There Still So Many Jobs? The History and Future of Workplace Automation." J E con Perspect. **29**(3):3–30.*
6. ^*Sheridan TB (2016). "Human–Robot Interaction: Status and Challenges." Hum Factors. **58**(4):525–532.*
7. ^*Sheridan TB, Verplank WL (1978). "Human and Computer Control of Undersea Teleoperators." MIT Man-Machine Systems Lab Report.*
8. ^*Kellogg A, Valentine M, Christin A (2020). "Algorithms at Work: The New Contested Terrain of Control." Ac ad Manag Ann. **14**(1):366–410.*
9. ^*Parasuraman R, Sheridan T, Wickens C (2000). "A Model for Types and Levels of Human Interaction With Automation." IEEE Trans Syst Man Cybern. **30**(3):286–297.*
10. ^*Risko EF, Gilbert SJ (2016). "Cognitive Offloading." Trends Cogn Sci. **20**(9):676–688.*
11. ^*Sparrow B, Liu J, Wegner DM (2011). "Google Effects on Memory: Cognitive Consequences of Having Infor mation at Our Fingertips." Science. **333**(6043):776–778.*
12. ^*Clark A, Chalmers D (1998). "The Extended Mind." Analysis. **58**(1):7–19.*

13. ^Parasuraman R, Riley V (1997). "Humans and Automation: Use, Misuse, Disuse, Abuse." Hum Factors. **39** (2):230–253.

14. ^Shneiderman H (2020). "Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy." Int J Hum Comput Interact. **36**(6):495–504.

15. ^Zuboff S (2019). The Age of Surveillance Capitalism. PublicAffairs.

16. ^Bjork RA, Bjork EL (2011). "Making Things Hard on Yourself, But in a Good Way: Creating Desirable Difficulties to Enhance Learning." In: Psychology and the Real World.

17. ^Autor D (2015). "Why Are There Still So Many Jobs? The History and Future of Workplace Automation." J Econ Perspect. **29**(3):3–30.

18. ^McCosker J, Wilken A (2020). "Automating Creativity: Artificial Intelligence and Creative Labor." Convergence. **26**(5–6):1101–1115.

19. ^Kellogg A, Valentine M, Christin A (2020). "Algorithms at Work: The New Contested Terrain of Control." Acad Manag Ann. **14**(1):366–410.

20. ^Susskind R, Susskind D (2015). The Future of the Professions. Oxford University Press.

21. ^Gillespie T (2014). "Algorithmically Recognizable: Santorum's Google Problem, and Google's Santorum Problem." Inf Commun Soc. **17**(3):295–310.

22. ^Zuboff S (2019). The Age of Surveillance Capitalism. PublicAffairs.

23. ^Lee JD, See KA (2004). "Trust in Automation: Designing for Appropriate Reliance." Hum Factors. **46**(1):50–80.

24. ^Skitka M, Mosier K, Burdick M (1999). "Does Automation Bias Decision-Making?" Int J Hum Comput Stud. **51**(5):991–1006.

25. ^Bender E et al. (2021). "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: Proc FAccT.

26. ^Parasuraman R, Riley V (1997). "Humans and Automation: Use, Misuse, Disuse, Abuse." Hum Factors. **39** (2):230–253.

27. ^Wood J, Bruner D, Ross G (1976). "The Role of Tutoring in Problem Solving." J Child Psychol Psychiatry. **17** (2):89–100.

28. ^Floridi L, Cowls J (2019). "A Unified Framework of Five Principles for AI in Society." Harvard Data Sci Rev.

29. ^DiMaggio P, Powell W (1983). "The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality." Am Sociol Rev. **48**(2):147–160.

30. ^*Fincher S, Robins A (2019). The Cambridge Handbook of Computing Education Research. Cambridge University Press.*
31. ^*Ericsson K (1993). "Deliberate Practice and Acquisition of Expert Performance." Psychol Rev.*
32. ^*Susskind R, Susskind D (2015). The Future of the Professions. Oxford University Press.*
33. ^*Birhane R et al. (2022). "The Values Encoded in Machine Learning Research." Proc ACM Conf Fairness Accountability Transparency.*
34. ^*Bender E et al. (2021). "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" FAccT.*
35. ^*Lipton Z (2018). "The Mythos of Model Interpretability." Commun ACM. **61**(10):36–43.*
36. ^*Veale M, Zuiderveen Borgesius F (2021). "Demystifying the Draft EU Artificial Intelligence Act." Comput Law Rev Int.*
37. ^*Barocas S, Hardt M, Narayanan A (2019). Fairness and Machine Learning.*
38. ^*Warschauer M (2003). Technology and Social Inclusion: Rethinking the Digital Divide. MIT Press.*

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.