AI for Technology: Application Practice and Future Prospects of Technical Intelligence in High-Tech Fields (Postprint)

Authors: Chen Yunji, Guo Qi

Date: 2024-03-27T00:00:00+00:00

Abstract

The fifth paradigm of scientific research, centered on AI for Science, has been widely applied across multiple natural sciences and high-tech fields. Unlike the application of artificial intelligence (AI) in natural sciences, which emphasizes discovering new principles, mechanisms, and laws, high-tech fields place greater emphasis on utilizing AI technology to invent and create new solutions, tools, and products to address domain-specific problems. This article summarizes the typical characteristics and scientific problems of AI applications in high-tech fields– "AI for Technology" –and introduces past successful cases using fully automated CPU chip design as an example. Finally, the article points out that the goal of AI for Technology is not only to accelerate the innovation process and reduce manual effort, but also to endow it with stronger creative capabilities, ultimately surpassing human-level performance.

Full Text

AI for Technology: Applied Practices and Future Perspectives of Technological Intelligence in High-Tech Areas

Chen Yunji[1],[2]*, **Guo Qi**[1]

[1]State Key Laboratory of Processors, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract

As the core of the fifth research paradigm, AI for Science has been widely applied across multiple natural science and high-tech fields. However, the application of

artificial intelligence in natural sciences emphasizes discovering new principles, mechanisms, and laws, whereas high-tech fields focus more on using AI to invent and create new solutions, tools, and products to solve specific domain problems. This article summarizes the typical characteristics and scientific challenges of AI applications in high-tech domains—termed "AI for Technology"—and presents a successful case study on fully automated CPU chip design. Finally, we argue that the goal of AI for Technology is not only to accelerate innovation processes and reduce human effort, but also to endow machines with superior creative capabilities that eventually surpass human levels.

Keywords: technological intelligence, AI for Science, invention and creation, CPU design

1. Concept and Connotation of AI for Technology

Recent breakthroughs in deep learning and other AI technologies have enabled widespread applications in mathematics, physics, chemistry, biology, materials science, manufacturing, and other natural science and high-tech fields. For instance, DeepMind has used machine learning methods to assist in mathematical conjecture discovery and theorem proving [1]; in biology, AlphaFold2 can now predict over 350,000 human genome proteins and 214 million proteins across more than one million species, covering nearly all known proteins on Earth and solving a fifty-year-old challenge in structural biology [2]; DeepMind and the Swiss Plasma Center have collaborated to apply reinforcement learning for optimizing tokamak plasma control in nuclear fusion [3]; and Professor David Baker's team at the University of Washington has used AI to precisely design macrocyclic polypeptide molecules capable of crossing cell membranes, innovating new approaches for oral drug design [4]. These successful applications mark AI for Science as the core of the fifth research paradigm [5], becoming a powerful tool for enhancing research efficiency, advancing scientific discovery, and driving technological innovation with the potential to bring about major societal transformations.

Although AI for Science has broad applications, its implementation varies across disciplines. We propose further subdividing it into broad and narrow senses. Broad AI for Science encompasses diverse AI applications across scientific and technological fields, including both discovery of natural laws and knowledge in natural sciences (such as proving mathematical conjectures and discovering physical laws) and solving critical technical challenges in high-tech fields (such as ultra-short-term weather forecasting, tokamak control, and biopharmaceuticals). Narrow AI for Science focuses specifically on discovering intrinsic laws, knowledge, and structures in natural sciences, such as Kepler's laws of planetary motion or human genome protein structures. Unlike narrow AI for Science, applying AI to solve critical technical challenges in high-tech fields primarily relies on inventing and creating new artifacts—including novel solutions, methods, tools, and products. Due to differences in application objectives and technical approaches from narrow AI for Science, we argue that AI applications in

high-tech fields are better categorized as AI for Technology.

Research on AI for Technology has attracted significant attention since AI' s inception. In 1969, Herbert Simon—a Nobel laureate in Economics and Turing Award winner, and one of AI' s founders—distinguished between "natural objects" and "artificial objects" in his book *The Sciences of the Artificial* [6], establishing that inventing artificial objects to meet human needs is itself a science that can be modeled through general problem-solving systems based on computer programs to achieve "design without human intervention." Simon and fellow AI pioneer Allen Newell implemented such general problem-solving systems to automatically solve diverse problem types [7], essentially modeling human problem-solving as a machine-executed search process. The core component is a "Generator-Test"cycle, where a generator produces numerous candidate solutions and a test determines whether candidates meet requirements, iterating until a satisfactory solution is found.

Building on this framework, AI for Technology can be modeled as a "search+verification" process. The core involves using search algorithms to select appropriate candidates, automatically verifying whether they meet requirements, and if not, automatically modifying and adjusting to generate new candidates until the final output satisfies all specifications. Recent rapid AI advancements promise to enhance both search and verification efficiency, expanding application domains while accelerating the entire problem-solving workflow.

Table 1 summarizes the differences between narrow AI for Science and AI for Technology. From an application purpose perspective, narrow AI for Science aims to discover unknown operating mechanisms, principles, laws, and structures in natural sciences, whereas AI for Technology emphasizes inventing solutions, methods, tools, and artifacts that meet specific needs. Using information theory as an analogy, AI for Science can be viewed as information encoding and compression—transforming large observational datasets into symbolic laws or knowledge through AI—while AI for Technology represents information decoding and decompression—using AI to decode numerous example specifications into concrete design details and components for artificial objects. Regarding output characteristics, AI for Science is inherently exploratory with unknown results, whereas AI for Technology designs artifacts to meet predefined specifications with precisely known expected outputs. Technically, AI for Science primarily leverages AI' s modeling capabilities for accurate fitting of observational data, while AI for Technology emphasizes AI' s generative capacity to produce target artifacts meeting specifications. In terms of algorithmic precision, AI for Science pursues statistical acceptability across large datasets, requiring results that reasonably explain natural phenomena (e.g., conforming to statistical distributions), whereas AI for Technology demands individual precision, requiring each output to exactly meet predefined specifications—for instance, automatically designed program code must correctly satisfy functional and performance specifications. Thus, AI for Technology imposes higher precision requirements

on AI algorithms.

2. Scientific Problems and Key Challenges of AI for Technology

Implementing the "search+verification" cycle in AI for Technology essentially requires solving the problem of finding optimal solutions that precisely satisfy complex constraints within vast, high-dimensional spaces. For real engineering problems, the search space typically contains massive numbers of potential candidates. In Go, for example, the board has 361 positions with three possibilities each, yielding a state space of 3^{361}. In protein design, a protein with 200 amino acids has 20^{200} possible sequences. In software programming, a small program of only 100 instructions (where actual SPEC CPU programs typically contain millions of instructions [8]) already has a state space of 2^{6400}. This means computer programs must search within enormous high-dimensional spaces.

The search objective is to find outputs that satisfy human needs, which involve multiple dimensions including functionality, performance, and even psychological perception, making constraints exceptionally complex. Mobile phone design, for instance, must satisfy not only core functional and performance parameters but also subjective constraints related to visual, tactile, and interactive experiences. Traditional manual methods, due to limited human search capacity and verification overhead, typically restrict themselves to finding satisfactory solutions within heavily pruned spaces using expert domain knowledge. AI methods, lacking domain knowledge or facing difficulties in formalizing it, must search directly within the vast original space, considering far more potential candidates than human experts and thus discovering superior unknown solutions. However, because the space is so enormous that even computer programs cannot exhaustively explore it, precise pruning through AI techniques to compress the space by multiple orders of magnitude without losing optimal solutions becomes crucial.

Three major challenges emerge in solving these scientific problems: search efficiency, constraint expression, and verification precision.

Challenge 1: How to effectively prune vast high-dimensional spaces. For traditional manual methods, human cognitive limitations and verification costs necessitate introducing expert domain knowledge to drastically prune the space before searching within the limited remaining space. AI techniques, without such knowledge or facing formalization difficulties, must search directly in the enormous high-dimensional space. While this allows consideration of more potential candidates than human experts, enabling discovery of superior unknown solutions, the space remains too large for exhaustive exploration even by computer programs. Therefore, precise space pruning through AI to compress the space by orders of magnitude without losing optimal solutions is essential.

Challenge 2: How to accurately express ambiguous, incomplete human requirement specifications. Human needs are often described in nat-

ural language, which is inherently ambiguous. Moreover, initial user requirements are frequently incomplete, requiring iterative interaction to refine and clarify specifications. Herbert Simon [9] used naval ship design as an example to illustrate this complexity, noting that continuous iterative interaction among commanders, combat personnel, designers, and component specialists is needed to transform requirements into "well-structured problems" suitable for computer solving. Recently, large language models, having learned extensive human commonsense and experience, could provide effective assistance in converting requirement descriptions into formal problem definitions.

Challenge 3: How to guarantee that individual outputs precisely satisfy complex constraints. As mentioned, AI for Technology requires that each individual artifact exactly meets predefined specifications—absolute correctness on single samples. This contradicts mainstream AI algorithms (such as neural networks) that emphasize statistical precision (where occasional image misclassification is acceptable). Even large language models, despite improved precision in many scenarios, cannot theoretically guarantee accuracy, limiting their application in critical contexts. Therefore, algorithmic innovations that can theoretically guarantee output precision or provide theoretical lower bounds are needed, enabling users to clearly judge whether results meet specifications.

3. Application Practice of AI for Technology: Fully Automated CPU Design

We have applied the fundamental principles of AI for Technology to the design and implementation of central processing units (CPUs)—the core physical carriers of information technology—achieving the first successful fully automated design of a 32-bit CPU, "Enlightenment 1," without human intervention [10]. Unlike traditional CPU design flows that typically require 2-3 years to produce an industrial-grade chip, our team completed the entire front-end design of Enlightenment 1 in only 5 hours, dramatically improving design efficiency and promising to transform traditional chip design workflows.

Unlike conventional manual CPU design that begins with requirement specifications and relies primarily on engineers for architecture design, logic design, and functional verification, our CPU design method is essentially verification-centric: starting from random circuits under guidance from verification plans, machines automatically complete iterative cycles of verification, debugging, and repair until obtaining target circuits that meet design requirements (Figure 1 [Figure 1: see original paper]). Automatic verification primarily checks whether results meet requirements and automatically generates new test cases. Automatic debugging searches for and locates erroneous circuit logic based on failed results. Automatic repair further searches for correct circuit logic based on identified errors. Both automatic debugging and repair can be viewed as search processes, and together with automatic verification, they form a complete workflow following the core "search+verification" paradigm of AI for Technology.

The verification-centric automated design flow faces enormous challenges in both search and verification. For search, Enlightenment 1 has 1,798 input and 1,826 output variables, requiring exploration of $2^{21798} \times {}^{1826}$ circuit combinations—a space far larger than Go' s search space. For verification, CPU design has zero tolerance for functional errors, as any flaw would cause catastrophic losses. Drawing from Intel Pentium processor verification experience, over 10 trillion instructions must execute correctly before tape-out, meaning AI-generated circuits must achieve precision higher than 99.99999999999%—far exceeding typical AI algorithm accuracy of around 90% for image or speech recognition. This renders many methods, including neural networks, reinforcement learning, and decision trees, helpless. Large language models, for example, cannot theoretically guarantee precision through more training samples, and still require human involvement in verification and debugging via prompting, preventing full automation.

To guarantee verification precision, we propose a design method based on Binary Speculation Diagrams (BSD). BSD builds upon traditional Binary Decision Diagrams (BDD) by replacing deterministic subgraphs in BDD with speculation nodes determined through Monte Carlo sampling. This approach naturally possesses good interpretability and monotonicity (each circuit modification brings the design closer to correctness), thereby addressing the aforementioned automatic debugging and repair challenges. Specifically, BDD' s tree structure can quickly search and determine relationships between node logic functions and external inputs/outputs to automatically locate errors, solving automatic debugging. As BDD expands through continuous searching, its corresponding logic functions can theoretically approximate the original function, solving automatic repair.

To improve search efficiency, we designed a Boolean function distance to measure similarity between different BSD nodes for merging. First, Monte Carlo sampling determines Boolean function distances between BSD nodes. Nodes with similar distances are then clustered to unify their expansion order. Finally, BSD nodes with zero distance are merged. Through repeated node expansion and merging, this approach reduces the original design space from 10^{10540} to 10^6. Our team theoretically proved that BSD node merging does not reduce precision, thereby achieving high-efficiency circuit logic search while maintaining accuracy.

Fully automated CPU design represents a typical AI for Technology application—inventing and creating CPU designs through AI. We discovered that the automatically designed CPU not only satisfied the functional requirements predefined by the Instruction Set Architecture (ISA) but also autonomously discovered the von Neumann architecture, including controllers and arithmetic units. Since the machine had no prior knowledge of the von Neumann architecture, this simultaneously exhibits characteristics of AI for Science in terms of "scientific discovery" and "unknown results."

[Figure 1: see original paper]

4. Future Perspectives of AI for Technology

To enable deeper AI for Technology applications across more high-tech fields, future research should focus on the core "search+verification" workflow to further improve search and verification efficiency, accelerate innovation processes, enhance creative capabilities, and ultimately exceed human invention levels. Specific explorations can proceed along two directions: cross-fusion of AI paradigms and integration with the third research paradigm.

From the search perspective, the core objective is improving search algorithm efficiency to approach optimal solutions faster. Gradient descent has achieved tremendous success in neural networks and other domains, but many practical problems are non-differentiable or suffer significant precision loss from differentiable approximations, making direct gradient descent application difficult. In such cases, cross-fusion of multiple AI paradigms should be considered. For example, AlphaGo' s Monte Carlo Tree Search combines connectionism (deep learning) and behaviorism (reinforcement learning) [11], demonstrating that these paradigms have already converged in practical applications. The CPU design example described earlier primarily uses symbolism (BDD) for searching. Future deep integration of symbolism, connectionism, and behaviorism could substantially improve search efficiency, finding better results in larger search spaces.

From the verification perspective, judging whether outputs meet specifications often requires experimental validation in real environments. For instance, new material designs require actual experiments to thoroughly test mechanical properties and durability, causing substantial resource and time costs. To accelerate verification convergence, computer simulations can build response models for interaction-based specification verification. In CPU design practice, it is impossible to verify every possible processor design through actual tape-out; instead, accurate simulators determine whether requirements are met. Therefore, deep integration with the third research paradigm based on computer simulation to construct efficient and accurate response models promises to further accelerate verification and the entire innovation workflow.

References

1. Davies A, Veličković P, Buesing L, et al. Advancing mathematics by guiding human intuition with AI. Nature, 2021, 600: 70-74.
2. Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold. Nature, 2021, 596: 583-589.
3. Degrave J, Felici F, Buchli J, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. Nature, 2022, 602: 414-419.
4. Bhardwaj G, O' Connor J, Rettie S, et al. Accurate de novo design of membrane-traversing macrocycles. Cell, 2022, 185(19): 3520-3532.

5. Li G J. AI4R: The fifth scientific research paradigm. Bulletin of Chinese Academy of Sciences, 2024, 39(1): 1-9. (in Chinese)

6. Simon H A. The Science of the Artificial. Cambridge: The MIT Press, 1969.

7. Simon H, Newell A. Human problem solving: The state of the theory in 1970. American Psychologist, 1971, 26: 145-159.

8. Phansalkar A, Joshi A, John L K. Analysis of redundancy and application balance in the SPEC CPU2006 benchmark suite//Proceedings of International Symposium on Computer Architecture. San Diego: ACM, 2007: 412-423.

9. Simon H. The structure of ill-structured problems. Artificial Intelligence, 1973, 4(3-4): 181-201.

10. Guo Q, Luo Y, Li S, et al. Pushing the limits of machine design: Automated CPU design with AI. (2023-06-21). https://arxiv.org/abs/2306.12456.

11. Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 2016, 529: 484-489.

Note: Figure translations are in progress. See original paper for figures.