Annals of Inquiry

Why Computers Won’t Make Themselves Smarter

We fear and yearn for “the singularity.” But it will probably never come.

By Ted Chiang

March 30, 2021

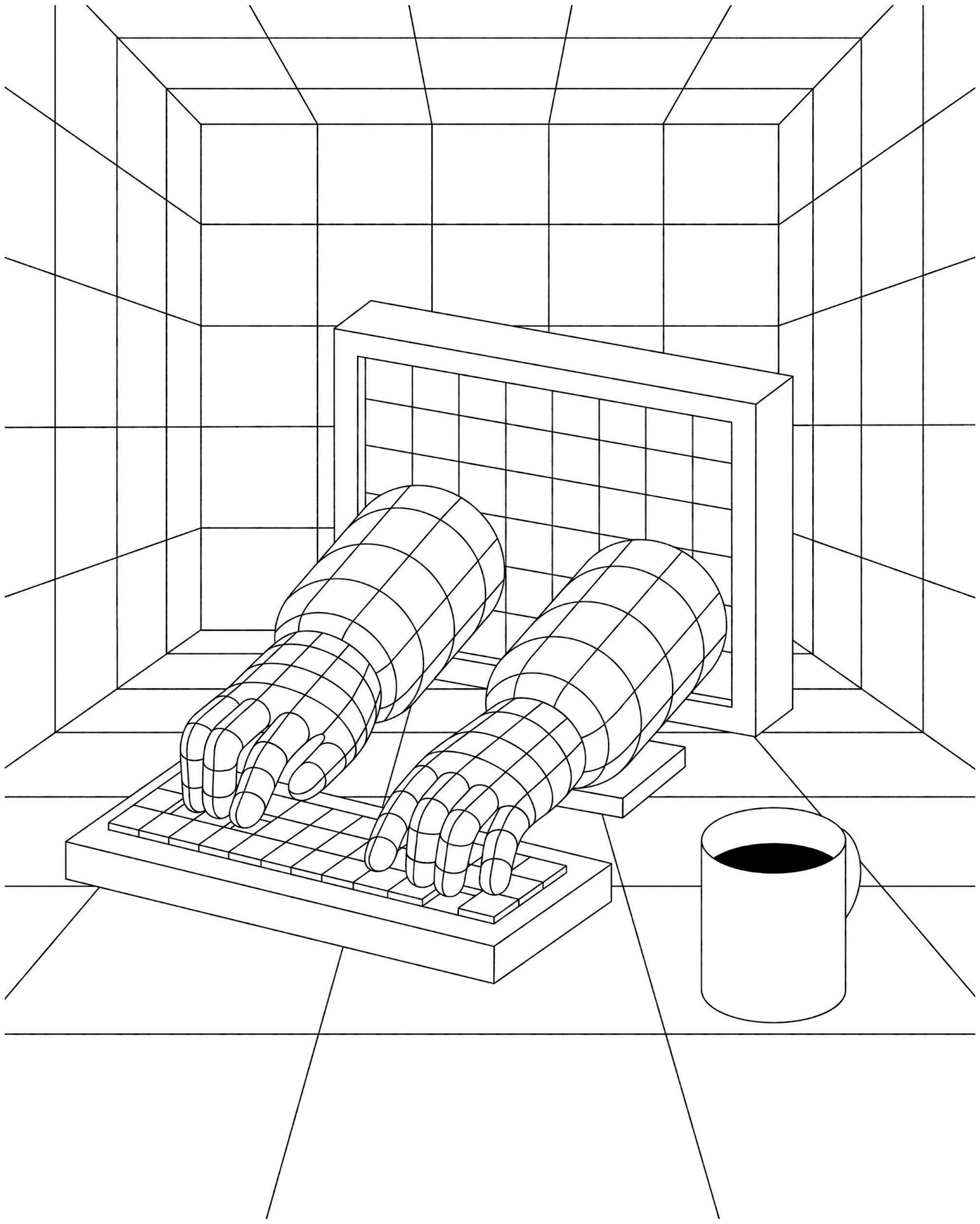
Illustration by Woshibai

Save this story

In the eleventh century, St. Anselm of Canterbury proposed an argument for the existence of God that went roughly like this: God is, by definition, the greatest being that we can imagine; a God that doesn't exist is clearly not as great as a God that does exist; ergo, God must exist. This is known as the ontological argument, and there are enough people who find it convincing that it's still being discussed, nearly a thousand years later. Some critics of the ontological argument contend that it essentially defines a being into existence, and that that is not how definitions work.

God isn't the only being that people have tried to argue into existence. "Let an ultraintelligent machine be defined as a machine that can far surpass all the intellectual activities of any man however clever," the mathematician Irving John Good wrote, in 1965:

> Since the design of machines is one of these intellectual activities, an ultraintelligent machine could design even better machines; there would then unquestionably be an "intelligence explosion," and the intelligence of man would be left far behind. Thus the first ultraintelligent machine is the *last* invention that man need ever make, provided that the machine is docile enough to tell us how to keep it under control.

The idea of an intelligence explosion was revived in 1993, by the author and computer scientist Vernor Vinge, who called it "the singularity," and the idea has since achieved some popularity among technologists and philosophers. Books such as Nick Bostrom's "Superintelligence: Paths, Dangers, Strategies," Max Tegmark's "Life 3.0: Being Human in the age of Artificial Intelligence," and Stuart Russell's "Human Compatible: Artificial Intelligence and the Problem of Control" all describe scenarios of "recursive self-improvement," in which an artificial-intelligence program designs an improved version of itself repeatedly.

I believe that Good's and Anselm's arguments have something in common, which is that, in both cases, a lot of the work is being done by the initial definitions. These definitions seem superficially reasonable, which is why they are generally accepted at face value, but they deserve closer examination. I think that the more we scrutinize the implicit assumptions of Good's argument, the less plausible the idea of an intelligence explosion becomes.

What might recursive self-improvement look like for human beings? For the sake of convenience, we'll describe human intelligence in terms of I.Q., not as an endorsement of I.Q. testing but because I.Q. represents the idea that intelligence can be usefully captured by a single number, this idea being one of the assumptions made by proponents of an intelligence explosion. In that case, recursive self-improvement would look like this: Once there's a person with an I.Q. of, say, 300, one of the problems this person can solve is how to convert a person with an I.Q. of 300 into a person with an I.Q. of 350. And then a person with an I.Q. of 350 will be able to solve the more difficult problem of converting a person with an I.Q. of 350 into a person with an I.Q. of 400. And so forth.

Do we have any reason to think that this is the way intelligence works? I don't believe that we do. For example, there are plenty of people who have I.Q.s of 130, and there's a smaller number of people who have I.Q.s of 160. None of them have been able to increase the intelligence of someone with an I.Q. of 70 to 100, which is implied to be an easier task. None of them can even increase the intelligence of animals, whose intelligence is considered to be too low to be measured by I.Q. tests. If increasing someone's I.Q. were an activity like solving a set of math puzzles, we ought to see successful examples of it at the low end, where the problems are easier to solve. But we don't see strong evidence of that happening.

Maybe it's because we're currently too far from the necessary threshold; maybe an I.Q. of 300 is the minimum needed to increase anyone's intelligence at all. But, even if that were true, we still don't have good reason to believe that endless recursive self-improvement is likely. For example, it's entirely possible that the best that a person with an I.Q. of 300 can do is increase another person's I.Q. to 200. That would allow one person with an I.Q. of 300 to grant everyone around them an I.Q. of 200, which frankly would be an amazing accomplishment. But that would still leave us at a plateau; there would be no recursive self-improvement and no intelligence explosion.

The I.B.M. research engineer Emerson Pugh is credited with saying "If the human brain were so simple that we could understand it, we would be so simple that we couldn't." This statement makes intuitive sense, but, more importantly, we can point to a concrete example in support of it: the microscopic roundworm *C. elegans*. It is probably one of the best-understood organisms in history; scientists have sequenced its genome and know the lineage of cell divisions that give rise to each of the nine hundred and fifty-nine somatic cells in its body, and have mapped every connection between its three hundred and two neurons. But they still don't completely understand its behavior. The human brain is estimated to have eighty-six billion neurons on average, and we will probably need most of them to comprehend what's going on in *C. elegans's* three hundred and two; this ratio doesn't bode well for our prospects of understanding what's going on within ourselves.

Some proponents of an intelligence explosion argue that it's possible to increase a system's intelligence without fully understanding how the system works. They imply that intelligent systems, such as the human brain or an A.I. program, have one or more hidden "intelligence knobs," and that we only need to be smart enough to find the knobs. I'm not sure that we currently have many good candidates for these knobs, so it's hard to evaluate the reasonableness of this idea. Perhaps the most commonly suggested way to “turn up” artificial intelligence is to increase the speed of the hardware on which a program runs. Some have said that, once we create software that is as intelligent as a human being, running the software on a faster computer will effectively create superhuman intelligence. Would this lead to an intelligence explosion?

Let’s imagine that we have an A.I. program that is just as intelligent and capable as the average human computer programmer. Now suppose that we increase its computer’s speed a hundred times and let the program run for a year. That’d be the equivalent of locking an average human being in a room for a hundred years, with nothing to do except work on an assigned programming task. Many human beings would consider this a hellish prison sentence, but, for the purposes of this scenario, let’s imagine that the A.I. doesn’t feel the same way. We’ll assume that the A.I. has all the desirable properties of a human being but doesn’t possess any of the other properties that would act as obstacles in this scenario, such as a need for novelty or a desire to make one’s own choices. (It’s not clear to me that this is a reasonable assumption, but we can leave that question for another time.)

So now we’ve got a human-equivalent A.I. that is spending a hundred person-years on a single task. What kind of results can we expect it to achieve? Suppose this A.I. could write and debug a thousand lines of code per day, which is a prodigious level of productivity. At that rate, a century would be almost enough time for it to single-handedly write Windows XP, which supposedly consisted of forty-five million lines of code. That's an impressive accomplishment, but a far cry from its being able to write an A.I. more intelligent than itself. Creating a smarter A.I. requires more than the ability to write good code; it would require a major breakthrough in A.I. research, and that's not something an average computer programmer is guaranteed to achieve, no matter how much time you give them.

When you're developing software, you typically use a program known as a compiler. The compiler takes the source code you've written, in a language such as C, and translates it into an executable program: a file consisting of machine code that the computer understands. Suppose you're not happy with the C compiler you're using—call it CompilerZero. CompilerZero takes a long time to process your source code, and the programs it generates take a long time to run. You're confident that you can do better, so you write a new C compiler, one that generates more efficient machine code; this new one is known as an optimizing compiler.

You've written your optimizing compiler in C, so you can use CompilerZero to translate your source code into an executable program. Call this program CompilerOne. Thanks to your ingenuity, CompilerOne now generates programs that run more quickly. But CompilerOne itself still takes a long time to run, because it's a product of CompilerZero. What can you do?

You can use CompilerOne to compile itself. You feed CompilerOne its own source code, and it generates a new executable file consisting of more efficient machine code. Call this CompilerTwo. CompilerTwo also generates programs that run very quickly, but it has the added advantage of running very quickly itself. Congratulations—you have written a self-improving computer program.

But this is as far as it goes. If you feed the same source code into CompilerTwo, all it does is generate another copy of CompilerTwo. It cannot create a CompilerThree and initiate an escalating series of ever-better compilers. If you want a compiler that generates programs that run insanely fast, you will have to look elsewhere to get it.

The technique of having a compiler compile itself is known as bootstrapping, and it's been employed since the nineteen-sixties. Optimizing compilers have come a long way since then, so the differences between a CompilerZero and a CompilerTwo can be much bigger than they used to be, but all of that progress was achieved by human programmers rather than by compilers improving themselves. And, although compilers are very different from artificial-intelligence programs, they offer a useful precedent for thinking about the idea of an intelligence explosion, because they are computer programs that generate other computer programs, and because when they do so optimization is often a priority.

The more you know about the intended use of a program, the better you can optimize its code. Human programmers sometimes hand-optimize sections of a program, meaning that they specify the machine instructions directly; the humans can write machine code that's more efficient than what a compiler generates, because they know more about what the program is supposed to do than the compiler does. The compilers that do the best job of optimization are compilers for what are known as domain-specific languages, which are designed for writing narrow categories of programs. For example, there's a programming language called Halide designed exclusively for writing image-processing programs. Because the intended use of these programs is so specific, a Halide compiler can generate code as good as or better than what a human programmer can write. But a Halide compiler cannot compile itself, because a language optimized for image processing doesn't have all the features needed to write a compiler. You need a general-purpose language to do that, and general-purpose compilers have trouble matching human programmers when it comes to generating machine code.

A general-purpose compiler has to be able to compile anything. If you feed it the source code for a word processor, it will generate a word processor; if you feed it the source code for an MP3 player, it will generate an MP3 player; and so forth. If, tomorrow, a programmer invents a new kind of program, something as unfamiliar to us today as the very first Web browser was in 1990, she will feed the source code into a general-purpose compiler, which will dutifully generate that new program. So, although compilers are not in any sense intelligent, they have one thing in common with intelligent human beings: they are capable of handling inputs that they have never seen before.

Compare this with the way A.I. programs are currently designed. Take an A.I. program that is presented with chess moves and that, in response, needs only to spit out chess moves. Its job is very specific, and knowing that is enormously helpful in optimizing its performance. The same is true of an A.I. program that will be given only "Jeopardy!" clues and needs only to spit out answers in the form of a question. A few A.I. programs have been designed to play a handful of similar games, but the expected range of inputs and outputs is still extremely narrow. Now, alternatively, suppose that you're writing an A.I. program and you have no advance knowledge of what type of inputs it can expect or of what form a correct response will take. In that situation, it's hard to optimize performance, because you have no idea what you're optimizing for.

How much can you optimize for generality? To what extent can you simultaneously optimize a system for every possible situation, including situations never encountered before? Presumably, some improvement is possible, but the idea of an intelligence explosion implies that there is essentially no limit to the extent of optimization that can be achieved. This is a very strong claim. If someone is asserting that infinite optimization for generality is possible, I'd like to see some arguments besides citing examples of optimization for specialized tasks.

Obviously, none of this proves that an intelligence explosion is impossible. Indeed, I doubt that one could prove such a thing, because such matters probably aren't within the domain of mathematical proof. This isn't a question of proving that something is impossible; it's a question of what constitutes good justification for belief. The critics of Anselm's ontological argument aren't trying to prove that there is no God; they're just saying that Anselm's argument doesn't constitute a good reason to believe that God exists. Similarly, a definition of an "ultraintelligent machine" is not sufficient reason to think that we can construct such a device.

There is one context in which I think recursive self-improvement is a meaningful concept, and it's when we consider the capabilities of human civilization as a whole. Note that this is different from individual intelligence. There's no reason to believe that humans born ten thousand years ago were any less intelligent than humans born today; they had exactly the same ability to learn as we do. But, nowadays, we have ten thousand years of technological advances at our disposal, and those technologies aren't just physical—they're also cognitive.

Let's consider Arabic numerals as compared with Roman numerals. With a positional notation system, such as the one created by Arabic numerals, it's easier to perform multiplication and division; if you're competing in a multiplication contest, Arabic numerals provide you with an advantage. But I wouldn't say that someone using Arabic numerals is smarter than someone using Roman numerals. By analogy, if you're trying to tighten a bolt and use a wrench, you'll do better than someone who has a pair of pliers, but it wouldn't be fair to say you're stronger. You have a tool that offers you greater mechanical advantage; it's only when we give your competitor the same tool that we can fairly judge who is stronger. Cognitive tools such as Arabic numerals offer a similar advantage; if we want to compare individuals' intelligence, they have to be equipped with the same tools.

Simple tools make it possible to create complex ones; this is just as true for cognitive tools as it is for physical ones. Humanity has developed thousands of such tools throughout history, ranging from double-entry bookkeeping to the Cartesian coördinate system. So, even though we aren't more intelligent than we used to be, we have at our disposal a wider range of cognitive tools, which, in turn, enable us to invent even more powerful tools.

This is how recursive self-improvement takes place—not at the level of individuals but at the level of human civilization as a whole. I wouldn't say that Isaac Newton made himself more intelligent when he invented calculus; he must have been mighty intelligent in order to invent it in the first place. Calculus enabled him to solve certain problems that he couldn't solve before, but he was not the biggest beneficiary of his invention—the rest of humanity was. Those who came after Newton benefitted from calculus in two ways: in the short term, they could solve problems that they couldn't solve before; in the long term, they could build on Newton's work and devise other, even more powerful mathematical techniques.

This ability of humans to build on one another's work is precisely why I don't believe that running a human-equivalent A.I. program for a hundred years in isolation is a good way to produce major breakthroughs. An individual working in complete isolation can come up with a breakthrough but is unlikely to do so repeatedly; you're better off having a lot of people drawing inspiration from one another. They don't have to be directly collaborating; any field of research will simply do better when it has many people working in it.

Consider the study of DNA as an example. James Watson and Francis Crick were both active for decades after publishing, in 1953, their paper on the structure of DNA, but none of the major breakthroughs subsequently achieved in DNA research were made by them. They didn’t invent techniques for DNA sequencing; someone else did. They didn’t develop the polymerase chain reaction that made DNA synthesis affordable; someone else did. This is in no way an insult to Watson and Crick. It just means that if you had A.I. versions of them and ran them at a hundred times normal speed, you probably wouldn’t get results as good as what we obtained with molecular biologists around the world studying DNA. Innovation doesn’t happen in isolation; scientists draw from the work of other scientists.

The rate of innovation is increasing and will continue to do so even without any machine able to design its successor. Some might call this phenomenon an intelligence explosion, but I think it’s more accurate to call it a technological explosion that includes cognitive technologies along with physical ones. Computer hardware and software are the latest cognitive technologies, and they are powerful aids to innovation, but they can’t generate a technological explosion by themselves. You need people to do that, and the more the better. Giving better hardware and software to one smart individual is helpful, but the real benefits come when everyone has them. Our current technological explosion is a result of billions of people using those cognitive tools.

Could A.I. programs take the place of those humans, so that an explosion occurs in the digital realm faster than it does in ours? Possibly, but let’s think about what it would require. The strategy most likely to succeed would be essentially to duplicate all of human civilization in software, with eight billion human-equivalent A.I.s going about their business. That’s probably cost-prohibitive, so the task then becomes identifying the smallest subset of human civilization that can generate most of the innovation you’re looking for. One way to think about this is to ask: How many people do you need to put together a Manhattan Project? Note that this is different from asking how many scientists actually worked on the Manhattan Project. The relevant question is: How large of a population do you need to draw from in order to recruit enough scientists to staff such an effort?

In the same way that only one person in several thousand can get a Ph.D. in physics, you might have to generate several thousand human-equivalent A.I.s in order to get one Ph.D.-in-physics-equivalent A.I. It took the combined populations of the U.S. and Europe in 1942 to put together the Manhattan Project. Nowadays, research labs don’t restrict themselves to two continents when recruiting, because building the best team possible requires drawing from the biggest pool of talent available. If the goal is to generate as much innovation as the entire human race, you might not be able to dramatically reduce that initial figure of eight billion after all.

We’re a long way off from being able to create a single human-equivalent A.I., let alone billions of them. For the foreseeable future, the ongoing technological explosion will be driven by humans using previously invented tools to invent new ones; there won’t be a “last invention that man need ever make.” In one respect, this is reassuring, because, contrary to Good’s claim, human intelligence will never be “left far behind.” But, in the same way that we needn’t worry about a superhumanly intelligent A.I. destroying civilization, we shouldn’t look forward to a superhumanly intelligent A.I. saving us in spite of ourselves. For better or worse, the fate of our species will depend on human decision-making.

Ted Chiang is the author of two collections of science-fiction short stories, “Stories of Your Life and Others,” and “Exhalation.”

More:TechnologyArtificial Intelligence (A.I.)Machine LearningResearchIntelligenceScience