Google Fed a Language Algorithm Math Equations. It Learned How to Solve New Ones. - ExtremeTech

By Adam Dachis on June 5, 2019 at 12:40 pm 46 Comments

Many people think of computers as “math experts” compared with us humans. While we can’t solve equations as quickly as a machine, we shouldn’t put too much faith in their accuracy because computers can’t know, understand, or calculate every possibility in an infinite series of numbers no matter how much time they have available. This limitation of computational hardware leads to strange quirks in how computers perform math, but Google’s new method of training AI to both understand and solve complex math problems may result in a sharp increase in future computational accuracy.

First, let’s take a look at what Google did because it’s an impressive approach in and of itself. For training data, DeepMind received a series of equations along with their solutions—like a math textbook, only without any explanation of how those solutions can be reached. Google then created a modular system to procedurally generate new equations to solve, with a controllable level of difficulty, and instructed the AI to provide answers in any form. Without any structure, DeepMind had to intuit how to solve new equations solely based on seeing a limited number of completed examples.

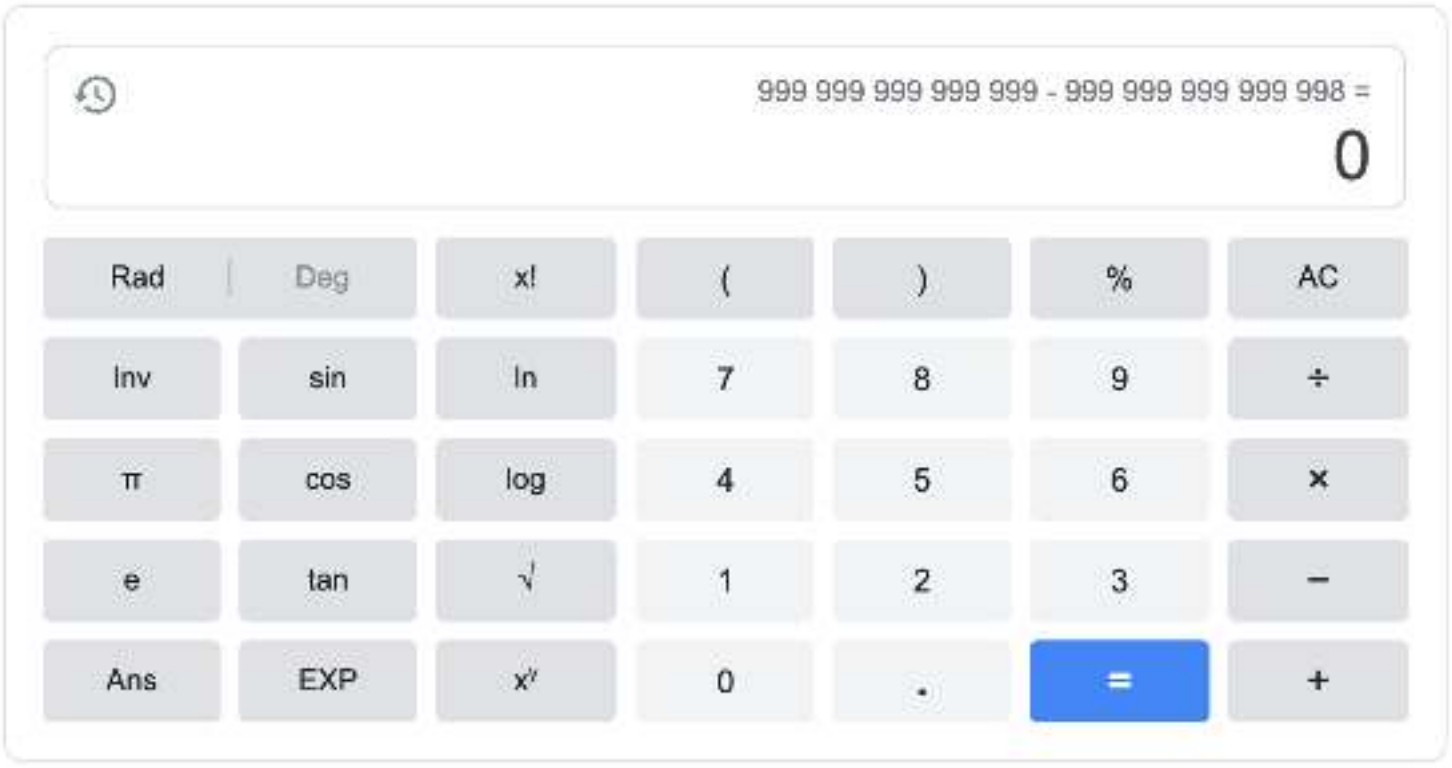
Challenging existing deep learning algorithms with modular math presents a very difficult challenge to an AI and existing neural network models performed at relatively similar levels of accuracy. The best-performing model, known as Transformer, managed to provide correct solutions to 50 percent of the time and it was designed for the purpose of natural language understanding—not math. When only judging Transformer on its ability to answer questions that utilized numbers seen in the training data, its accuracy shot up to 76 percent.

While the best results represent a failing grade and a solid C, they're nevertheless extremely impressive. Aside from this method offering a simple and effective means of gauging a model's aptitude at certain types of tasks, it could lead to a solution to the biggest flaw in computer math proficiency.

To understand that problem, let's take a quick look at how computers make mathematical errors *by design*. Consider the following example. Despite two very large numbers, you can probably solve the following equation in a near instant:

> 999999999999999 – 999999999999998

While you should have no trouble determining that the second number is just a single digit smaller than the first, and therefore the answer is 1, a calculator (like Google's) will present an obviously incorrect result.


999 999 999 999 999 - 999 999 999 999 998 =

0

The reason for this flaw lies at the core of computational architecture. While we understand math through the base-10/decimal numeral system, computers see things differently through base-2/binary. You can see the difference when you take a look at the real number line as you know it.

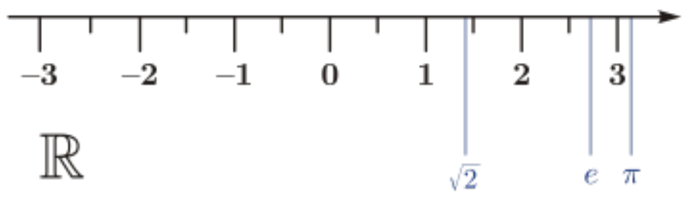


Image credit: Wikipedia

If you count whole numbers in a sequence you'll find yourself speaking the real number line aloud. We can create all the numbers we need from the digits that range from 0 to 9. Computers, on the other hand, only have 0 and 1 to determine numbers and that can result in some unusual errors like the one pictured above. All the data in our computers exist as a series of ones and zeros and that doesn't exempt numbers. Here's a look at how the real number line gets translated to binary for computers.

To put it lightly, things become a bit more complicated.

Imagine making a copy of the Oxford English Dictionary when you can only represent its contents using two letters in the alphabet. Computers *can* accomplish this because alphabets only contain a finite number of symbols and they only require order and representation. The real number line, on the other hand, is an infinite series. No human or computer can record and display the entirety of an infinite series or that series would become finite. If you understand that impossibility, it's not hard to imagine why a computer can only manage to calculate a fraction of that number. After all, it has to understand all those numbers in its own binary system. Where we see 9, computers see 1001. Take a look at how computers see the number 85:

Computers use sums of multiples of two to successfully represent a significant segment of the real number line—just not all of it. The decimal numeral system works in the same way—just with 10s instead of 2s—but it takes significantly fewer base 10 digits to represent any given number when compared to the same value in binary. All number systems, by nature, can represent smaller numbers more precisely than larger ones, but because binary offers fewer unique digits to represent each number it runs out of space faster (when compared to the decimal numeral system/base 10).

This prevents computers from representing every possible number on the number line—something you've experienced in base 10 if you've ever encountered the fraction of one-third. You know that the sum of 1/3 + 1/3 + 1/3 equals 1, but if you represent 1/3 as a decimal it becomes 0.3333333 and continues on infinitely. Unlike the fractional representation of 1/3, the decimal version appears to add up to 0.9999999 (etc.) and never reach 1 because the decimal numeral system cannot represent the fraction of 1/3 with the necessary precision.

The same thing happens with computers and so they employ strategic rounding to get to the closest number they *can* represent. This results in reduced precision but allows for a greater range of calculations. As a result, specific equations can exploit the weaknesses in the binary numeral system and cause rounding errors that make the computer yield an incorrect result.

By the way, you've just read a major oversimplification of how computer rounding errors work. Check out the video above if you want a more precise explanation of how all the math shakes out.

Just as fractional notation (e.g. 1/3) can help us overcome the limitations of base 10, engineers have provided special logic to help machines overcome the more problematic limitations of base 2. However, the processing limitations of CPUs, the larger number of representative digits required, and the high number of decimals with infinite representations in binary combine to present a problem without a perfect solution.

Computers aren't alone in experiencing these rounding errors. You can see the core of this dilemma represented literally everywhere you look when you consider how your proximity to other things determines the level of detail you can perceive about them. To the human eye, precision of detail diminishes with distance. That distance, however, can allow us to see a fuller picture by sacrificing precision. The same reality presents itself in different ways across all known number systems.

By creating an artificial intelligence training method that determines an algorithm's ability to approach computation using its own abstracted methodology, Google created a framework for achieving a much higher level of computational accuracy in the future. With the Transformer language model landing first prize for accuracy out of the gate, even though it only managed to get half the questions right it provides a clue that suggests the direction of the model that could someday achieve perfect accuracy throughout the spectrum of math that today's computers solve with imperfections. Given Transformer's much higher score for solving equations through interpolation (76 percent), increased accuracy may arrive through a combination of algorithmic changes and a more significant set of training data.

In any case, we won't have a perfect machine calculator that can understand all math by next week. Right now, that remains barely more than a pipe dream. After all, the modular set of equations already limit themselves to school-level difficulty and no model can achieve anywhere near perfect accuracy. With the code that generates these equations publicly available, we may have a way to get there someday.

Top image credit: Getty Images

Now read:

- This AI algorithm can match the average American on real SAT questions
- Intel, Marvell, Qualcomm Pledge Support for Glow AI Compiler
- Artificial neural networks are changing the world. What are they?