

# Measuring Mathematical Problem Solving With the MATH Dataset

**Dan Hendrycks**  
UC Berkeley

**Collin Burns**  
UC Berkeley

**Saurav Kadavath**  
UC Berkeley

**Akul Arora**  
UC Berkeley

**Steven Basart**  
UChicago

**Eric Tang**  
UC Berkeley

**Dawn Song**  
UC Berkeley

**Jacob Steinhardt**  
UC Berkeley

## Abstract

Many intellectual endeavors require mathematical problem solving, but this skill remains beyond the capabilities of computers. To measure this ability in machine learning models, we introduce MATH, a new dataset of 12,500 challenging competition mathematics problems. Each problem in MATH has a full step-by-step solution which can be used to teach models to generate answer derivations and explanations. To facilitate future research and increase accuracy on MATH, we also contribute a large auxiliary pretraining dataset which helps teach models the fundamentals of mathematics. Even though we are able to increase accuracy on MATH, our results show that accuracy remains relatively low, even with enormous Transformer models. Moreover, we find that simply increasing budgets and model parameter counts will be impractical for achieving strong mathematical reasoning if scaling trends continue. While scaling Transformers is automatically solving most other text-based tasks, scaling is not currently solving MATH. To have more traction on mathematical problem solving we will likely need new algorithmic advancements from the broader research community.

## 1 Introduction

Mathematics is a highly effective tool in many intellectual endeavors. It enables us to count and quantify objects, and it can be relied upon because it is consistent and based on logic. Mathematics pervades the sciences and can be used to model planetary orbits, atomic motion, signal frequencies, and much more. These phenomena can be encoded with mathematics precisely and concisely. This has even led some to describe mathematics as being “unreasonably effective” (Wigner, 1960). These observations speak to the broad reach and domain-generality of mathematics.

In machine learning, mathematics is a valuable testbed for *problem-solving ability*: the ability to analyze a problem, pick out good heuristics from a large set of possibilities, and chain them together to produce an answer. This contrasts with plug-and-chug calculations, a skill which ML models can already exhibit (Henighan et al., 2020). Visual or linguistic reasoning may involve limited problem-solving ability for tasks such as image classification, but unlike math this is not the focus of these domains.

To measure the problem-solving ability of machine learning models, we introduce the MATH dataset, which consists of 12,500 problems from high school math competitions. Given a problem from MATH, machine learning models generate a sequence, such as  $\$\\frac{2}{3}\$$ , that encodes the final answer. These answers are unique after normalization, allowing MATH to be scored with exact match rather than with heuristic metrics such as BLEU. In addition, MATH problems are tagged by difficulty from 1 to 5, and span seven subjects including geometry, where diagrams can be specified in text with the Asymptote language. This enables a fine-grained assessment of mathematical problem-solving ability across difficulties and subjects. Finally, problems come with full step-by-step solutions. By training on these, models can learn to generate their own step-by-step solutions, which can facilitate learning and make model outputs more interpretable.

The MATH dataset is challenging: large language models achieved accuracies ranging from 2.9% to 6.9%. Despite these low accuracies, models clearly possess some mathematical knowledge: they achieve up to 15% accuracy on the easiest difficulty level, and they are able to generate step-by-step solutions that are coherent and on-topic even when

## Metamath Theorem Proving

To prove:  $n \in \mathbb{N} \wedge \frac{n+1}{2} \in \mathbb{N} \implies \exists m \in \mathbb{N} : n = 2m + 1$ .  
 GPT-f’s generated proof:

```
| - ((N e. NN0 /\ ((N + 1)/2) e. NN0) ->
      ((N - 1) / 2) e. NN0)
| - (N e. NN0 -> N e. CC)
| - 1 e. CC
| - ((N e. CC /\ 1 e. CC) ->
      (N - 1) e. CC )
:
:
```

## DeepMind Mathematics Dataset

Problem: Divide 1136975704 by -142121963  
 Answer: -8  
 Problem: Calculate  $((-2)/3) / (-1 - (-24)/9)$   
 Answer: -2/5  
 Problem: Let  $k(u) = u^{**2} + u - 4$ . Find  $k(0)$   
 Answer: -4  
 Problem: Sort 2, 4, 0, 6  
 Answer: 0, 2, 4, 6  
 Problem: Solve  $4 - 4 - 4 = 188*m$  for  $m$   
 Answer: -1/47

## MATH Dataset (Ours)

**Problem:** Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

**Solution:** There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ( $\binom{4}{2} = 6$  results). The total number of distinct pairs of marbles Tom can choose is  $1 + 6 = \boxed{7}$ .

**Problem:** If  $\sum_{n=0}^{\infty} \cos^{2n} \theta = 5$ , what is  $\cos 2\theta$ ?

**Solution:** This geometric series is  $1 + \cos^2 \theta + \cos^4 \theta + \dots = \frac{1}{1 - \cos^2 \theta} = 5$ . Hence,

$$\cos^2 \theta = \frac{4}{5}. \text{ Then } \cos 2\theta = 2 \cos^2 \theta - 1 = \boxed{\frac{3}{5}}.$$

**Problem:** The equation  $x^2 + 2x = i$  has two complex solutions. Determine the product of their real parts.

**Solution:** Complete the square by adding 1 to each side. Then  $(x + 1)^2 = 1 + i = e^{\frac{i\pi}{4}} \sqrt{2}$ , so  $x + 1 = \pm e^{\frac{i\pi}{8}} \sqrt[4]{2}$ .

The desired product is then

$$(-1 + \cos(\frac{\pi}{8}) \sqrt[4]{2})(-1 - \cos(\frac{\pi}{8}) \sqrt[4]{2}) = \\ 1 - \cos^2(\frac{\pi}{8}) \sqrt{2} = 1 - \frac{(1+\cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1-\sqrt{2}}{2}}.$$

Figure 1: Previous work is based on formal theorem provers or straightforward plug-and-chug problems. Our dataset, MATH, has competition mathematics problems with step-by-step solutions written in L<sup>A</sup>T<sub>E</sub>X and natural language. Models are tasked with generating tokens to construct the final (boxed) answer.

incorrect. We also evaluated humans on MATH, and found that a computer science PhD student who does not especially like mathematics attained approximately 40% on MATH, while a three-time IMO gold medalist attained 90%, indicating that MATH can be challenging for humans as well.

The presence of step-by-step solutions allows models to utilize “scratch space”: rather than having to generate a final answer immediately, models can first generate solutions that may contain intermediate computations. Interestingly, we found that having models generate step-by-step solutions before producing an answer actually *decreased* accuracy relative to immediately outputting a final answer without generating solutions, indicating the solutions are currently not useful for models at test time. In contrast, having models *train* on solutions increases relative accuracy by 10% compared to training on the questions and answers directly. We also find that models do better with hints in the form of partial solutions. Our results show that models can make use of actual step-by-step solutions provided to them in various ways, but that they are still unable to effectively use their own generated solutions. Bridging this gap poses an interesting direction for further research.

While MATH covers advanced problem-solving techniques, models may first need to be trained thoroughly on the fundamentals of mathematics. To address this, we create the first large-scale mathematics pretraining dataset with hundreds of thousands of step-by-step solutions in natural language and L<sup>A</sup>T<sub>E</sub>X. We call this dataset the Auxiliary Mathematics Problems and Solutions (AMPS) pretraining corpus, which consists of Khan Academy and Mathematica data. AMPS has over 100,000 Khan Academy problems with step-by-step solutions in L<sup>A</sup>T<sub>E</sub>X; these exercises are used to teach human students concepts ranging from basic addition to Stokes’ Theorem. It also contains over 5 million problems generated using Mathematica scripts, based on 100 hand-designed modules covering topics such as conic sections, div grad and curl, KL divergence, eigenvalues, polyhedra, and Diophantine equations. In total AMPS contains 23GB of problems and solutions. Domain-specific pretraining (Gururangan et al., 2020) on AMPS improves relative accuracy by around 25%, equivalent to a 15 $\times$  increase in model size.

Altogether, while large Transformer models (Vaswani et al., 2017) make some progress on the MATH dataset, such as by AMPS pretraining or by training with step-by-step solutions, accuracy nonetheless remains relatively low. While enormous Transformers pretrained on massive datasets can now solve most existing text-based tasks, this low accuracy indicates that our MATH dataset is distinctly harder. Accuracy also increases only modestly with model size: assuming a log-linear scaling trend, models would need around  $10^{35}$  parameters to achieve 40% accuracy on math, which is impractical. Instead, to make large strides on the MATH dataset with a practical amount of resources, we will need new algorithmic advancements from the broader research community.

## 2 Related Work

**Neural Theorem Provers.** Much of the existing work on machine learning models for mathematical reasoning relies on automated theorem proving benchmarks. Huang et al. (2019) use the Coq theorem proving environment to create a machine learning benchmark with 1,602 theorems and lemmas. Bansal et al. (2019) introduce the HOList benchmark for automated theorem proving, which uses a formal language to enable automatic evaluation. Rather than use HOList, Polu and Sutskever (2020) use the Metamath formalization language for automated theorem proving with promising results. We show an example of Metamath in Figure 1. These benchmarks can be approached with seq2seq (Sutskever et al., 2014) Transformers which have traction on the problem (Polu and Sutskever, 2020; Rabe et al., 2020; Li et al., 2020).

Rather than prove theorems with standard pretrained Transformers, McAllester (2020) proposes that the community create theorem provers that bootstrap their mathematical capabilities through open-ended self-improvement. For bootstrapping to be feasible, models will also need to understand mathematics as humans write it, as manually converting advanced mathematics to a proof generation language is extremely time-consuming. This is why Szegedy (2020) argues that working on formal theorem provers alone will be an impractical path towards world-class mathematical reasoning. We address Szegedy (2020)'s concern by creating a dataset to test understanding of mathematics written in natural language and commonplace mathematical notation. This also means that the answers in our dataset can be assessed without the need for a cumbersome theorem proving environment, which is another advantage of our evaluation framework.

**Neural Calculators.** Recent work shows that Transformers can sometimes perform laborious calculations around as well as calculators and computer algebra systems. Lample and Charton (2020) use Transformers to solve algorithmically generated symbolic integration problems and achieve greater than 95% accuracy. Amini et al. (2019); Ling et al. (2017) introduce plug-and-chug multiple choice mathematics problems and focus on sequence-to-program generation. Saxton et al. (2019) introduce the DeepMind Mathematics dataset, which consists of algorithmically generated plug-and-chug problems such as addition, list sorting, and function evaluation, as shown in Figure 1. Recently, Henighan et al. (2020) show that nearly all of the DeepMind Mathematics dataset can be straightforwardly solved with large Transformers.

**Benchmarks for Enormous Transformers.** There are few existing natural language benchmarks left to solve, as tasks that aggregate multiple subtasks such as SuperGLUE (Wang et al., 2019) are solved by simply training enormous Transformers (He et al., 2020). Kaplan et al. (2020); Henighan et al. (2020) show that the performance of Transformers predictably increases with an increase in model size and dataset size, raising the question of whether natural language processing can be solved by simply increasing compute and funding. In Appendix A.1, we even find that large GPT-3 models can perform remarkably well on a sequence completion test similar to an IQ test, the C-Test (Hernández-Orallo, 1998; Legg and Hutter, 2007). Even difficult logical understanding tasks such as LogiQA (Liu et al., 2020) will soon be straightforwardly solved by enormous Transformers should trends continue, which we also show in Appendix A.1. Hendrycks et al. (2021) create a multiple-choice benchmark covering 57 subjects which are difficult for enormous Transformers. However, unlike our benchmark, which is a text generation task with 12,500 mathematical reasoning questions, their benchmark is a multiple choice task that includes only a few hundred questions about mathematics. We find that our MATH benchmark is especially challenging for current models and, if trends continue, simply using bigger versions of today's Transformers will not solve our task in the foreseeable future.

## 3 Datasets

In this section, we introduce two new datasets, one for benchmarking mathematical problem-solving ability (MATH) and one for pretraining (AMPS).

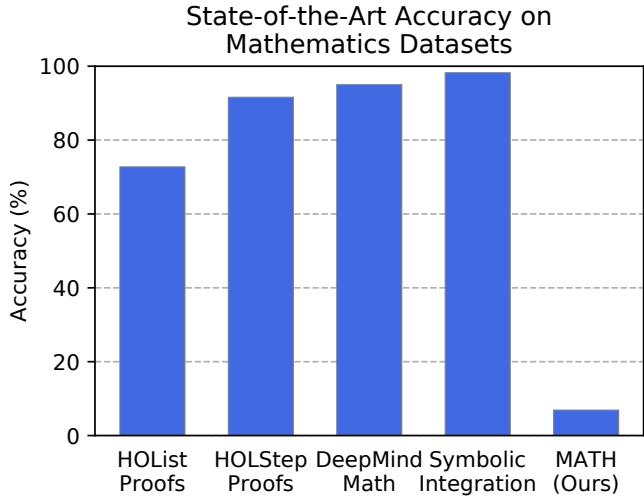


Figure 2: Compared to existing proof and plug-and-chug tasks, our mathematical problem solving task is considerably more challenging. HOList results are from Wu et al. (2021). HOLStep results are from Crouse et al. (2019). DeepMind Math accuracy is the median IID accuracy from Henighan et al. (2020). Symbolic Integration accuracy is from Lample and Charton (2020).

The DeepMind Mathematics dataset, which consists of algorithmically generated plug-and-chug problems such as addition, list sorting, and function evaluation, as shown in Figure 1. Recently, Henighan et al. (2020) show that nearly all of the DeepMind Mathematics dataset can be straightforwardly solved with large Transformers.

There are few existing natural language benchmarks left to solve, as tasks that aggregate multiple subtasks such as SuperGLUE (Wang et al., 2019) are solved by simply training enormous Transformers (He et al., 2020). Kaplan et al. (2020); Henighan et al. (2020) show that the performance of Transformers predictably increases with an increase in model size and dataset size, raising the question of whether natural language processing can be solved by simply increasing compute and funding. In Appendix A.1, we even find that large GPT-3 models can perform remarkably well on a sequence completion test similar to an IQ test, the C-Test (Hernández-Orallo, 1998; Legg and Hutter, 2007). Even difficult logical understanding tasks such as LogiQA (Liu et al., 2020) will soon be straightforwardly solved by enormous Transformers should trends continue, which we also show in Appendix A.1. Hendrycks et al. (2021) create a multiple-choice benchmark covering 57 subjects which are difficult for enormous Transformers. However, unlike our benchmark, which is a text generation task with 12,500 mathematical reasoning questions, their benchmark is a multiple choice task that includes only a few hundred questions about mathematics. We find that our MATH benchmark is especially challenging for current models and, if trends continue, simply using bigger versions of today's Transformers will not solve our task in the foreseeable future.

Algebra	Conic sections, polynomial GCD, De Moivre’s theorem, function inverses, parametric equations, ...
Calculus	Arclength, Jacobian, Laplacian, divergence, curl, gradients, integrals, power series expansion, ...
Statistics	Expectation, geometric mean, harmonic mean, KL divergence, variance and standard deviation, ...
Geometry	Triangle area, triangle inradius, triangle orthocenter, polygon angles, polyhedron diameter, ...
Linear Algebra	Characteristic polynomials, eigenvalues, point to plane distance, reduced row echelon form, ...
Number Theory	Modular inverse, Euler’s totient function, Diophantine equations, Chinese remainder theorem, ...

Table 1: A subset of the topics covered by our 100 hand-designed Mathematica scripts, which is part of our Auxiliary Mathematics Problems and Solutions (AMPS) pretraining dataset. Of these scripts, 37 also generate step-by-step solutions. We generated around 50,000 exercises with each Mathematica script, or around 5 million problems.

### 3.1 The MATH Dataset

The MATH dataset consists of problems from mathematics competitions including the AMC 10, AMC 12, AIME, and more. Many of these problems can be collected from [aops.com/community/c3158\\_usa\\_contests](https://aops.com/community/c3158_usa_contests). These competitions span decades and assess the mathematical problem-solving ability of the best young mathematical talent in the United States. Unlike most prior work, most problems in MATH cannot be solved with a straightforward application of standard K-12 mathematics tools. Instead, humans often solve such problem by applying problem solving techniques and “heuristics” (Pólya, 1945).

The Mathematics Aptitude Test of Heuristics dataset, abbreviated MATH, has 12,500 problems (7,500 training and 5,000 test). With this many training problems, models can learn many useful heuristics for problem solving. Each problem has a step-by-step solution and a final boxed answer. Example problems with step-by-step solutions are shown in Figure 1.

**Categorizing Problems.** Problems span various subjects and difficulties. The seven subjects are Prealgebra, Algebra, Number Theory, Counting and Probability, Geometry, Intermediate Algebra, and Precalculus. While subjects like Prealgebra are generally easier than Precalculus, within a subject problems can take on different difficulty levels. We encode a problem’s difficulty level from ‘1’ to ‘5,’ where a subject’s easiest problems for humans are assigned a difficulty level of ‘1,’ and a subject’s hardest problems are assigned a difficulty level of ‘5.’ Concretely, the first few problems of an AMC 8 exam are often level 1, while AIME problems are level 5. This allows us to assess performance across both different subjects and different levels of difficulty.

**Formatting.** Problems and solutions are consistently formatted using L<sup>A</sup>T<sub>E</sub>X and the Asymptote vector graphics language. Our usage of L<sup>A</sup>T<sub>E</sub>X allows us to flexibly encode mathematical problems while avoiding unusual symbols or cumbersome formal languages. Meanwhile, mathematical figures are encoded in the Asymptote language rather than as raster images. This enables pure language models to process figures, diagrams, and graphics, making it possible to assess these models on subjects such as geometry for the first time.

To assess models using exact match, we force the final boxed answers to follow consistent formatting rules. Specifically, probabilities are expressed as simplified fractions. Moreover, matrix entry fractions are encoded with  $x/y$ , while all other fractions are consistently encoded with the  $\frac{x}{y}$  command. Coefficients are encoded without a multiplication symbol (e.g.  $5x$  not  $5*x$ ). Expressions with multiple variables are entered in alphabetical order; polynomials are expressed in decreasing degree order. Different fraction encodings equivalent, such as  $\frac{x}{y}$  and  $\frac{dfrac{x}{y}}$  and  $x/y$ . Different parenthesis encodings, such as  $\left($  and  $($ , are treated as equivalent.

We also allow units to be included or omitted from an answer, we ignore spaces, and we treat common equivalent ways of expressing the same number (e.g., 0.5 and  $1/2$ , or 0.1 and .1) as the same. When the answer is a factorized polynomial, we permit different orderings of the factors, so that  $4(x+1)(x-1)$  is equivalent to  $4(x-1)(x+1)$ , and so on. These rules cover nearly all ways that different generated or actual solutions can be equivalent in practice.

**Automatically Assessing Generated Answers.** Due to design choices in MATH, we can assess the answers generated by a model *automatically*, even though the space of model outputs is combinatorially large. Automatic assessment starts by determining the beginning and end of the answer. This is possible to do even if a model generates step-by-step solutions because the final answers in MATH are wrapped and delimited with the  $\boxed{\cdot}$  command. We can consequently evaluate a model’s output by parsing what is inside the  $\boxed{\cdot}$  command and comparing that with the ground truth answer, while accounting for the equivalent ways of formatting a string described above. Together, the box delimiter and formatting rules provide a unique answer in a well-defined location, which allows us to test for equivalence and use accuracy as our primary metric.

**Human-Level Performance.** To estimate human-level performance, we randomly sampled 20 problems from the MATH test set and gave them to humans. We artificially require that the participants have 1 hour to work on the problems and must perform calculations by hand. All participants are university students. One participant who does not like mathematics got  $8/20 = 40\%$  correct. A participant ambivalent toward mathematics got  $13/20$ . Two participants who

Model	Prealgebra	Algebra	Number Theory	Counting & Probability	Geometry	Intermediate Algebra	Precalculus	Average
GPT-2 (0.1B)	5.2	5.1	5.0	2.8	5.7	6.5	7.3	5.4 (+0%)
GPT-2 (0.3B)	6.7	6.6	5.5	3.8	6.9	6.0	7.1	6.2 (+15%)
GPT-2 (0.7B)	6.9	6.1	5.5	5.1	8.2	5.8	7.7	6.4 (+19%)
GPT-2 (1.5B)	8.3	6.2	4.8	5.4	8.7	6.1	8.8	6.9 (+28%)
GPT-3 (2.7B)	2.8	2.9	3.9	3.6	2.1	2.5	2.6	2.9 (-46%)
GPT-3 (175B)	7.7	6.0	4.4	4.7	3.1	4.4	4.0	5.2 (-4%)

Table 2: MATH accuracies across subjects for GPT-2 and *few-shot* GPT-3 models. The character ‘B’ denotes the number of parameters in billions. The gray text indicates the *relative* improvement over the 0.1B baseline. All GPT-2 models pretrain on AMPS, and all values are percentages. A  $15\times$  increase in model parameters increased accuracy by 1.5%, a 28% relative improvement. Model accuracy is increasing very slowly, so much future research is needed.

like mathematics got 14/20 and 15/20. A participant who got a perfect score on the AMC 10 exam and attended USAMO several times got 18/20. A three-time IMO gold medalist got 18/20 = 90%, though missed questions were exclusively due to small errors of arithmetic. Expert-level performance is theoretically 100% given enough time, though even 40% accuracy for a machine learning model would be impressive.

### 3.2 AMPS (Khan + Mathematica) Dataset

Since pretraining data can greatly influence performance (Hernandez et al., 2021; Gururangan et al., 2020) and since mathematics is a small fraction of online text, we introduce a large and diverse mathematics pretraining corpus. Our pretraining dataset, the Auxiliary Mathematics Problems and Solutions (AMPS) dataset, has problems and step-by-step solutions typeset in L<sup>A</sup>T<sub>E</sub>X. AMPS contains over 100,000 problems pulled from Khan Academy and approximately 5 million problems generated from manually designed Mathematica scripts.

**Khan Academy.** The Khan Academy subset of AMPS has 693 exercise types with over 100,000 problems and full solutions. Problem types range from elementary mathematics (e.g. addition) to multivariable calculus (e.g. Stokes’ theorem), and are used to teach actual K-12 students. Many of the exercises can be regenerated using code from [github.com/Khan/khan-exercises/](https://github.com/Khan/khan-exercises/). We show the full list of problem types in the Figures 13 to 16.

**Mathematica.** To make AMPS larger, we also contribute our own Mathematica scripts to generate approximately 50 $\times$  more problems than our Khan Academy dataset. With Mathematica, we designed 100 scripts that test distinct mathematics concepts, 37 of which include full step-by-step L<sup>A</sup>T<sub>E</sub>X solutions in addition to final answers. We generated around 50,000 exercises from each of our scripts, or around 5 million problems in total. This results in over 23 GB of mathematics problems, making it larger than the 16 GB of natural language used to train BERT (Devlin et al., 2019).

Problems include various aspects of algebra, calculus, counting and statistics, geometry, linear algebra, and number theory (see Table 1 for a sampling of topics). Unlike prior approaches to algorithmically generating mathematics problems, we use Mathematica’s computer algebra system so that we can manipulate fractions, transcendental numbers, and analytic functions.

## 4 Experiments

In this section, we perform experiments to investigate performance on the MATH dataset. We find that accuracy remains low even for the best models. Furthermore, unlike for most other text-based datasets, we find that accuracy is increasingly very slowly with model size. If trends continue, then we will need algorithmic improvements, rather than just scale, to make substantial progress on MATH. Nevertheless, we show that making progress is also possible today. We find that pretraining on AMPS increases relative accuracy by 25%, which is comparable to the improvement due to a 15 $\times$  increase in model size.

We also experiment with using step-by-step solutions. We find that having models generate their own step-by-step solutions before producing an answer actually *degrades* accuracy. We qualitatively assess these generated solutions and find that while many steps remain illogical, they are often related to the question. Finally, we show that step-by-step solutions can still provide benefits today. We find that providing partial ground truth step-by-step solutions can improve performance, and that providing models with step-by-step solutions at training time also increases accuracy.

### 4.1 Experimental Setup

**Models and Hyperparameters.** Because MATH answers must be generated, we use autoregressive language models, namely GPT-2 (Radford et al., 2016) and GPT-3 (Brown et al., 2020), which are decoder models pretrained on natural language text. Our GPT-2 models tokenize numbers so that each digit is its own token (Henighan et al., 2020). We were

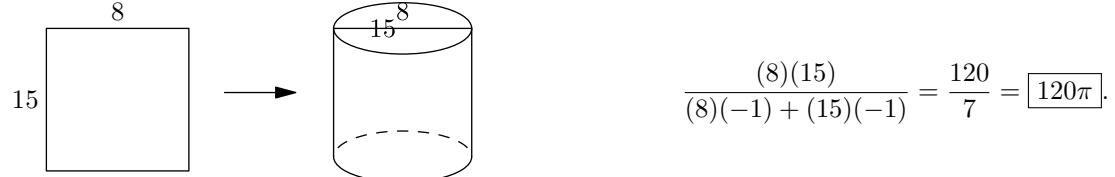
**Problem:** Suppose  $a$  and  $b$  are positive real numbers with  $a > b$  and  $ab = 8$ . Find the minimum value of  $\frac{a^2+b^2}{a-b}$ .

**Generated solution:** Let  $s = a + b$ . By QM-AM,  $\sqrt{\frac{a^2+b^2}{a-b}} \geq \frac{a+b}{2} = \frac{s}{2}$ . Then  $\frac{a^2+b^2}{a-b} \geq \frac{s^2}{4}$ , so  $a^2 + b^2 \geq \frac{s^2}{2}$ . Hence,  $\frac{a^2+b^2}{a-b} \geq \frac{s^2}{2} + \frac{2s^2}{a-b} = \frac{s^2}{2} + 2s^2 = \boxed{8}$ .

**Ground truth solution:** We can write  $\frac{a^2+b^2}{a-b} = \frac{a^2+b^2-2ab+16}{a-b} = \frac{(a-b)^2+16}{a-b} = a-b+\frac{16}{a-b}$ . By AM-GM,  $a-b+\frac{16}{a-b} \geq 2\sqrt{(a-b) \cdot \frac{16}{a-b}} = 8$ . Equality occurs when  $a-b=4$  and  $ab=8$ . We can solve these equations to find  $a=2\sqrt{3}+2$  and  $b=2\sqrt{3}-2$ . Thus, the minimum value is  $\boxed{8}$ .

**Problem:** Right  $\Delta ABC$  has legs measuring 8 cm and 15 cm. The triangle is rotated about one of its legs. What is the number of cubic centimeters in the maximum possible volume of the resulting solid? Express your answer in terms of  $\pi$ .

**Generated solution:** Rotating the triangle about one leg gives us the following triangle:



**Ground truth solution:** If the triangle is rotated about the shorter leg, then the radius is the longer leg and the height is the shorter leg, and the volume is  $\frac{1}{3} \cdot (15^2\pi)(8) = 600\pi$  cubic centimeters. If the triangle is rotated about the longer leg, then the radius is the shorter leg and the height is the longer leg, and the volume is  $\frac{1}{3}(8^2\pi)(15)$ , which is  $\frac{8}{15}$  of the volume we found earlier. So, the maximum possible volume is  $\boxed{600\pi}$  cubic centimeters.

Figure 3: Problems, step-by-step solutions generated by our GPT-2 1.5B model, and ground truth solutions. Observe that models can provide the right answer yet generate a misleading and wrong explanation. The second generated solution demonstrates that models are capable of generating Asymptote commands to create figures and graphics.

unable to get T5 (Raffel et al., 2020), which has a tokenizer that removes many L<sup>A</sup>T<sub>E</sub>X symbols, to have competitive accuracy after a broad hyperparameter sweep.

Before fine-tuning on MATH, models pretrain on AMPS. We pretrain for one epoch, using AdamW (Loshchilov and Hutter, 2019), using a batch size of 128, and using a weight decay of 0.05. We use the standard autoregressive language modeling objective. During pretraining, we upsample Khan Academy data by a factor of 5 and we downsample Mathematica by a factor of 2 to account for the large difference in dataset sizes.

During fine-tuning, models predict final answers and solutions. Concretely, if  $\langle P \rangle$  is the problem statement, we train with an equal mix of “ $\langle P \rangle$  Final Answer: <Answer>” and “ $\langle P \rangle$  Full Solution: <Step-by-Step Solution>” sequences. This makes it possible for the model to both generate full solutions and also to output just the final answer. For fine-tuning we use the same batch size and weight decay as in pretraining.

Unless otherwise specified, for GPT-2 we use the default HuggingFace (Wolf et al., 2020) generation parameters, except that we use beam search. Our beam search has a beam size of 20 when only generating the final answer, and a beam size of 10 when generating full step-by-step solutions. By default, we evaluate models by prompting them with “ $\langle P \rangle$  Final Answer :” so that they directly generate the final answer to each problem, not the step-by-step solution.

We also evaluate GPT-3 in a few-shot setting (no fine-tuning) using the OpenAI API. We use the ‘Ada’ GPT-3 model which has approximately 2.7 billion parameters, and the ‘Davinci’ model which has approximately 175 billion parameters. Since we are performing few-shot evaluation, we construct our prompt by prepending 8 problems with correct answers (but not step-by-step solutions due to space). Using temperature 0, models output up to 20 tokens for the final answer.

## 4.2 Analyzing Model Performance

**Model Size.** While increasing model parameters often automatically solves many tasks (Brown et al., 2020), we find that MATH is unusually challenging for enormous Transformers. Table 2 shows that the average accuracy across subjects for the smallest model, GPT-2 with 0.1 billion parameters, is 5.4%. Meanwhile, a GPT-2 model with 15× the number of parameters attains 6.9% accuracy, a 28% relative improvement. This indicates that while having more parameters helps, absolute accuracy remains far from the ceiling and is only increasing slowly, quite unlike most other text-based tasks.

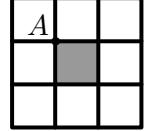
**Problem Difficulty.** We also analyze model accuracy while controlling for problem difficulty. Higher levels of difficulty correspond to lower accuracy, as expected. These results are visualized in Figure 10 of Appendix A.3. The accuracy of

**Problem:** The three-digit number " $ab5$ " is divisible by 3. How many different three-digit numbers can " $ab5$ " represent?  
**Generated solution:** A number is divisible by 3 if and only if the sum of its digits is divisible by 3. So  $ab5$  is divisible by 3 if and only if the two-digit number  $ab$  leaves a remainder of 1 when divided by 3. There are 90 two-digit numbers, of which  $90/3 = 30$  leave a remainder of 1 when divided by 3.

**Ground truth solution:** If the number is divisible by 3, the sum of the digits must be divisible by 3. So  $a + b$  must be one more than a multiple of 3, since 5 is one less than a multiple of 3. We have several options: If  $a + b = 1$ , it must be  $(1, 0)$ , one possibility. For  $a + b = 4$ ,  $a$  can be 1 to 4, for four possibilities. For  $a + b = 7$ ,  $a$  can be 1 to 7, so seven possibilities. For  $a + b = 10$ , anything from  $(1, 9)$  to  $(9, 1)$  works, so nine possibilities. For  $a + b = 13$ , the pairs range from  $(4, 9)$  to  $(9, 4)$ , for six possibilities. If  $a + b = 16$ , we can have  $(7, 9), (8, 8)$ , or  $(9, 7)$ , so three possibilities.  $a + b = 19$  and up is not possible. So the total number is  $1 + 4 + 7 + 9 + 6 + 3 = 30$

---

**Problem:** Beginning at point  $A$  in the diagram, Dora selects one of the four possible directions with equal probability. Each time she comes to an intersection, she again randomly selects one of the possible directions. What is the probability that, in her first four steps, she will walk completely around the gray square? Express your answer as a common fraction.



**Generated solution:** The total probability that Dora chooses a direction completely around the gray square is  $\frac{5}{10} = \frac{1}{2}$ .

**Ground truth solution:** The only way for the Dora to end up at her starting point in four steps is for her to traverse the four sides of the gray square. She can do this in two ways: clockwise and counterclockwise. The probability of each of these two paths is  $(\frac{1}{4})^4 = \frac{1}{256}$ . Therefore, the probability that she ends up where she started is  $\frac{1}{256} + \frac{1}{256} = \frac{1}{128}$ .

Figure 4: Additional example problems, generated solutions, and ground truth solutions from our MATH dataset. The first problem’s generated solution has the right answer with a correct and simple explanation. The second problem is a combinatorics problem specified with a figure, which the model gets wrong.

GPT-2 (1.5B) is around 15% for level 1 (easy) and around 4% for level 5 (hard). Even our benchmark’s easiest problems are more challenging than previous benchmarks that focused on straightforward plug-and-chug problems.

**AMPS Pretraining.** As an ablation, we test how models with AMPS pretraining compare with models that were not pretrained on AMPS. Without pretraining on AMPS, a GPT-2 (1.5B) model fine-tuned on MATH attains 5.5% accuracy. In contrast, a GPT-2 (1.5B) model both pretrained on AMPS and fine-tuned on MATH attains 6.9%, a 25% relative improvement in accuracy. Consequently AMPS increases accuracy about as much as a 15 $\times$  increase in parameters, indicating its value as a pretraining dataset.

We tried additionally pretraining on StackExchange, a real-world but less curated source of mathematics text. A GPT-2 (0.3B) model pretrained on both AMPS and questions and answers from Math StackExchange ( $\sim 3$  GB) had 6.0% accuracy, which is actually less than the 6.2% accuracy attained by pretraining on AMPS alone. Thus our dataset is more useful for pretraining even than diverse real-world mathematics data.

**Error Detection.** To determine whether we can trust the answers from a model, we analyze model confidence to see whether confidence tends to be higher for correct answers. We define confidence as the average prediction probability of the tokens that make up a generated answer. We histogram confidences for correct and incorrect answers in Figure 5. GPT-2 (1.5B) is highly overconfident, with confidences that are typically around 100%, and there is substantial overlap between correct and incorrect answers. Following Hendrycks and Gimpel (2017), we computed the probability that a correct answer has higher confidence than an incorrect answer. To do this, we compute the Area Under the Receiver Operating Characteristic curve (AUROC). An AUROC of 100% corresponds to being able to perfectly detect correct and incorrect answers, while 50% corresponds to random chance. We find that with GPT-2 (1.5B), the AUROC is quite low at 68.8%. This suggests there is substantial room for improvement in detecting model errors.

### 4.3 Analyzing Step-by-Step Solutions

**Scratch Space.** Our MATH dataset and AMPS pretraining dataset provide full step-by-step solutions, an important and rare type of side information (Murty et al., 2020) that can in principle teach models how to derive answers and use scratch space. By training a language model on these solutions, we can have models generate full step-by-step solutions. This may be especially useful for difficult problems, for which outputting the correct answer after just a few forward passes may be insufficient. By allowing the model to use several steps of processing before outputting a final answer, the model could adaptively use computation and have higher performance, in addition to making its reasoning more interpretable.

We test this by prompting models with “ $\langle P \rangle$  Full Solution:” to generate a full solution along with a final boxed answer, rather than the boxed answer alone. We evaluated this for GPT-2 (1.5B) and found that this actually makes

MATH Confidence Distribution

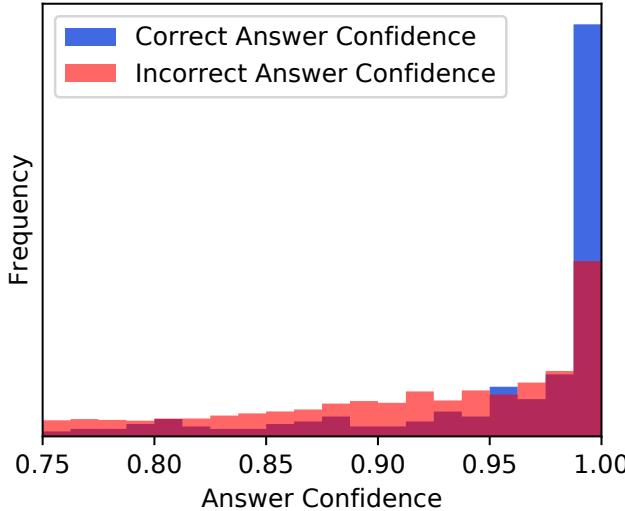


Figure 5: Histogram of the answer confidences of GPT-2 1.5B. The incorrect answer histogram is translucent and overlays the correct answer histogram. GPT-2 1.5B is overconfident and has an AUROC of 68.8%, indicating that it is not good at detecting errors from confidence alone. Confidences below 0.75 are omitted for ease of visualization.

performance worse, dropping accuracy to 5.3%. We hypothesize that the drop in accuracy from using scratch space arises from a snowballing effect, in which partially generated “solutions” with mistakes can derail subsequent generated text. Nevertheless, when generation becomes more reliable and models no longer confuse themselves by their own generations, our dataset’s solutions could in principle teach models to use scratch space and attain higher accuracy.

**Examples.** We can also qualitatively assess the step-by-step solutions that the model generates. We show examples of generated solutions in Figures 3 and 4. We find that the model can consistently generate correct LATEX and often performs steps that appear related to the question at hand, but still makes many logical mistakes, both in terms of what the question seems to be asking and in individual steps that are part of a larger derivation.

**The Benefits of MATH Solutions.** We find that giving models partial step-by-step MATH solutions during inference can improve accuracy. We test performance when we allow models to predict the final answer given a “hint” in the form of a portion of the ground truth step-by-step solution. To do so, for this experiment we prompt models with “ $\langle P \rangle$  <Partial Step-by-Step Solution without Final Answer> Final Answer:” during both fine-tuning and evaluation for different partial fractions of the step-by-step solution. This is the same as the default setting when we let models see 0% of the step-by-step solution. When models see “99%” of the solution, they are given the whole step-by-step solution except for the final answer. We show results with GPT-2 (0.7B) for different fractions of the solution in Figure 6. Observe that the model still only attains approximately 40% when given 99% of the solution, indicating room for improvement.

Finally, we also find that providing models with step-by-step during training can further improve performance. We run an ablation by fine-tuning models on MATH with the same setup as before, except that we only show examples with the final answer and no step-by-step solution. If we fine-tune with only the final answer, the GPT-2 (1.5B) accuracy decreases by 0.6% to 6.3%, a 10% relative reduction.

## 5 Conclusion

In this paper, we laid groundwork for future research in machine learning for mathematical problem solving. We introduced the MATH benchmark, which enables the community to measure mathematical problem-solving ability. In addition to having answers, all MATH problems also include answer explanations, which models can learn from to generate their own step-by-step solutions. We also introduce AMPS, a diverse pretraining corpus that can enable future models to learn virtually all of K-12 mathematics. While most other text-based tasks are already nearly solved by enormous Transformers, MATH is notably different. We showed that accuracy is slowly increasing and, if trends continue, the community will need to discover conceptual and algorithmic breakthroughs to attain strong performance on MATH. Given the broad reach and applicability of mathematics, solving the MATH dataset with machine learning would be of profound practical and intellectual significance.

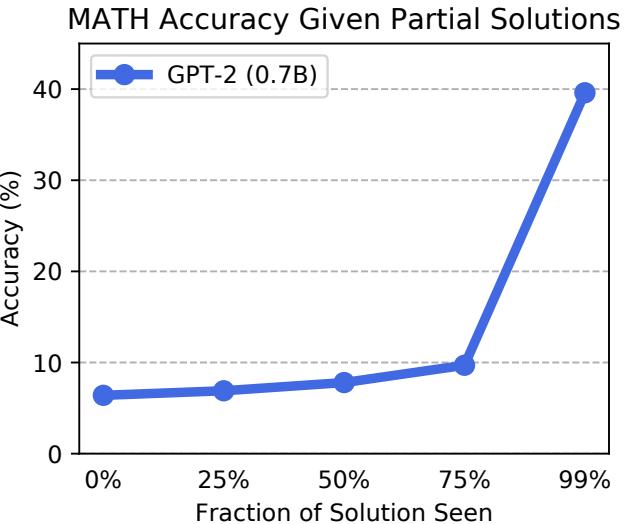


Figure 6: Models conditioned on most of a problem’s step-by-step solution can often understand the solution to predict the final answer. ‘99%’ of a solution is all the solution text before the final answer. Not all solutions have an answer that is immediate from from the preceding solution text.

## Acknowledgements

We would like to thank Jose Hernández-Orallo, Alex Gunning, and Neeraj Kapoor. DH is supported by the NSF GRFP Fellowship and an Open Philanthropy Project Fellowship.

## References

- Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. *ArXiv*, abs/1905.13319, 2019.
- K. Bansal, S. Loos, Markus N. Rabe, Christian Szegedy, and S. Wilcox. Holist: An environment for machine learning of higher-order theorem proving (extended version). *ArXiv*, abs/1904.03241, 2019.
- T. Brown, B. Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, G. Krüger, T. Henighan, R. Child, Aditya Ramesh, D. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, E. Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, J. Clark, Christopher Berner, Sam McCandlish, A. Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.
- Maxwell Crouse, I. Abdelaziz, Cristina Cornelio, Veronika Thost, L. Wu, Kenneth D. Forbus, and Achille Fokoue. Improving graph neural network representations of logical formulae with subgraph pooling. *ArXiv*, abs/1911.06904, 2019.
- J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *ArXiv*, abs/2004.10964, 2020.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654, 2020.
- Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*, 2017.
- Dan Hendrycks, C. Burns, Steven Basart, Andy Zou, Mantas Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. *ICLR*, 2021.
- T. Henighan, J. Kaplan, Mor Katz, Mark Chen, Christopher Hesse, J. Jackson, Heewoo Jun, T. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, B. Mann, A. Radford, Aditya Ramesh, Nick Ryder, D. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *ArXiv*, abs/2010.14701, 2020.
- Danny Hernandez, J. Kaplan, T. Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv*, 2021.
- J. Hernández-Orallo. Beyond the turing test. *Journal of Logic, Language and Information*, 9:447–466, 2000.
- Jose Hernández-Orallo. A formal definition of intelligence based on an intensional variant of algorithmic complexity. *EIS*, 1998.
- Daniel Huang, Prafulla Dhariwal, D. Song, and Ilya Sutskever. Gamepad: A learning environment for theorem proving. *ArXiv*, abs/1806.00608, 2019.
- J. Kaplan, Sam McCandlish, T. Henighan, T. Brown, Benjamin Chess, R. Child, Scott Gray, A. Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *ArXiv*, abs/2001.08361, 2020.
- Daniel Khashabi, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. Unifiedqa: Crossing format boundaries with a single qa system, 2020.
- Guillaume Lample and Francois Charton. Deep learning for symbolic mathematics. *ArXiv*, abs/1912.01412, 2020.
- S. Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. *Minds and Machines*, 17: 391–444, 2007.
- Wenda Li, L. Yu, Yuhuai Wu, and Lawrence C. Paulson. Modelling high-level mathematical reasoning in mechanised declarative proofs. *ArXiv*, abs/2006.09265, 2020.
- W. Ling, Dani Yogatama, Chris Dyer, and P. Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*, 2017.
- J. Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. LogiQA: A challenge dataset for machine reading comprehension with logical reasoning. In *IJCAI*, 2020.
- I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

- David A. McAllester. MathZero, the classification problem, and set-theoretic type theory. *ArXiv*, abs/2005.05512, 2020.
- Shikhar Murty, Pang Wei Koh, and Percy Liang. Expbert: Representation engineering with natural language explanations. In *ACL*, 2020.
- Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *ArXiv*, abs/2009.03393, 2020.
- George Pólya. *How to Solve It*. Princeton University Press, 1945.
- Markus N. Rabe, Dennis Lee, K. Bansal, and Christian Szegedy. Mathematical reasoning via self-supervised skip-tree training. *arXiv: Learning*, 2020.
- A. Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2016.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 2020.
- D. Saxton, Edward Grefenstette, Felix Hill, and P. Kohli. Analysing mathematical reasoning abilities of neural models. *ArXiv*, abs/1904.01557, 2019.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.
- Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In *CICM*, 2020.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Ł. Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Super glue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*, 2019.
- Eugene Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure and Applied Mathematics*, 1960.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020.
- Yuhuai Wu, M. Rabe, Wenda Li, Jimmy Ba, Roger B. Grosse, and Christian Szegedy. Lime: Learning inductive bias for primitives of mathematical reasoning. *ArXiv*, abs/2101.06223, 2021.

## A Appendix

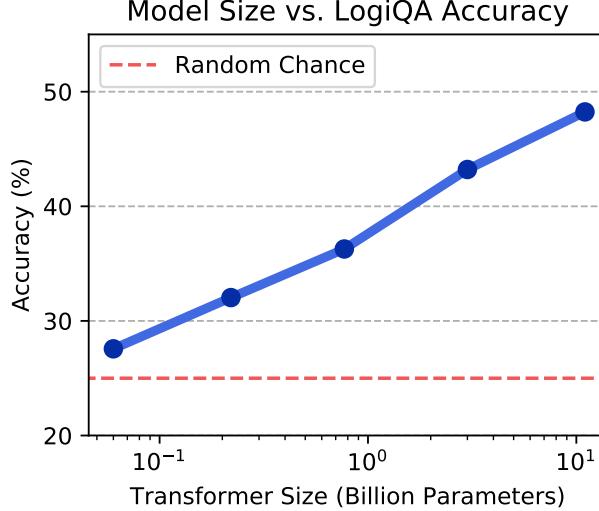


Figure 7: Difficult natural language tasks such as LogiQA will soon be solved just by making models larger, assuming trends continue. The Transformers in this figure are UnifiedQA (Khashabi et al., 2020) models of various sizes.

### A.1 Logic and Intelligence Tests

While enormous Transformers perform poorly on MATH, they do well on other logic and intelligence tests.

We analyze Transformers on LogiQA (Liu et al., 2020), a task with logical reasoning questions such as ‘‘David knows Mr. Zhang’s friend Jack, and Jack knows David’s friend Ms. Lin. Everyone of them who knows Jack has a master’s degree, and everyone of them who knows Ms. Lin is from Shanghai. Who is from Shanghai and has a master’s degree?’’ As shown in Figure 7, Transformers are improving on LogiQA, so much so that they will attain human-level performance relatively soon, should trends continue.

We also find that Transformers also do well on the C-Test, a pattern completion test that has a 77% correlation with human IQ (Hernández-Orallo, 2000). An example of a problem from C-Test is the sequence ‘‘a, a, z, c, y, e, x, \_’’ which has the answer ‘‘g.’’ We regenerated hundreds of C-Test examples to test GPT-3 (175B) in a 5-shot setting. While GPT-3 had abysmal performance when the sequences were letters, converting letters to numbers helped. After changing ‘‘a’’ to 0, ‘‘b’’ to 1, . . . , and ‘‘z’’ to 25, accuracy became approximately 40% on the hardest examples (C-Test questions with complexity ‘‘13’’). For comparison, on these same examples, average humans attained around 20% accuracy (Hernández-Orallo, 2000).

### A.2 Further Dataset Information

**Rendering Graphics.** For the first time, our dataset makes it possible for text-based models to process graphical mathematical figures by expressing figures in asymptote code. For example, Figure 8 shows asymptote code and the figure it produces. In short, it is possible to concisely specify many visual mathematics problems with code, sidestepping the complexity of multi-modal models.

**AMPS Examples.** We show concrete examples from AMPS in Figure 9. AMPS is a mixture of examples from Khan Academy and our 100 Mathematica modules.

**Contrasting AMPS and DeepMind Mathematics.** AMPS has several hundred exercise types or modules (Khan Academy has 693 modules and Mathematica has 100), while DeepMind mathematics (DM) has only a few dozen. We show all Khan Academy modules in Figures 13 to 16. Most DM exercises increase the diversity of problems by simply having a wide range of coefficients and constants. For example, its derivatives module exclusively covers polynomial derivatives with wide-ranging coefficients, while ours covers mixtures of dozens of major analytic functions. DM opts not to cover concepts and subjects such as logarithms and geometry, unlike AMPS. While DM is formatted in plaintext, AMPS is formatted in L<sup>A</sup>T<sub>E</sub>X. Finally, while DM solely has final answers, all 693 Khan Academy modules and 37 of our Mathematica modules have full step-by-step solutions.

```

size(40);
draw(shift(1.38,0)*yscale(0.3)*Circle((0,0), .38));

draw((1,0)--(1,-2));
draw((1.76,0)--(1.76,-2));

draw((1,-2)..(1.38,-2.114)..(1.76,-2));
path p =(1.38,-2.114)..(1.74,-1.5)..(1,-0.5)..(1.38,-.114);
pair a=(1.38,-2.114), b=(1.76,-1.5);
path q =subpath(p, 1, 2);
path r=subpath(p,0,1);
path s=subpath(p,2,3);
draw(r);
draw(s);
draw(q, dashed);

label("$5$",midpoint((1.76,0)--(1.76,-2)),E);

```



Figure 8: Example of asymptote code and the figure it produces.

#### Example from a Khan Academy module:

Problem: In history class, the girl to boy ratio is 9 to 6. If there are a total of 60 students, how many boys are there?

Solution: A ratio of 9 girls to 6 boys means that a set of 15 students will have 9 girls and 6 boys. A class of 60 students has 4 sets of 15 students. Because we know that there are 6 boys in each set of 15 students, the class must have 4 groups of 6 boys each. There is a total of 24 boys in history class.

Example Mathematica code that generates practice problems:

```

In[1]:= For[i=0,i<50000,i++,
  roundbasis = RandomChoice[{0.8,0.1,0.05,0.05}→{1,1/2,1/3,1/5}];
  d1 = RandomInteger[{1,6}];
  d2 = RandomInteger[{1,3}];
  q=0;
  p=0;
  While[q==0,
    For[j=0,j<d1,j++,
      q += Round[RandomReal[{-5,5}], roundbasis]*x^j;
    ];
  ];
  While[p==0,
    For[j=0,j<d2,j++,
      p += Round[RandomReal[{-5,5}], roundbasis]*x^j;
    ];
  ];
  p = RandomChoice[{p,Expand[q*p]}];
  Export["/amps/mathematica/algebra/polynomial_gcd/">>ToString[i]<>.txt",
  {"Problem:\nFind the greatest common divisor of $">>ToString[TeXForm[p//TraditionalForm]]<>"$ and $">>ToString[TeXForm[q//TraditionalForm]]<>"$.",
  "Answer:\n$">>ToString[TeXForm[PolynomialGCD[p,q]//TraditionalForm]]<>"$"}]
]

```

Figure 9: A Khan Academy problem and solution, followed by the code for a simple Mathematica module used to generate polynomials GCD problems. These problems are available in AMPS.

### A.3 Difficulty Analysis

We break down MATH accuracy by difficulty levels. In Figure 10, we observe that human difficulty and machine difficulty track each other. In Figure 11, we find that accuracy can vary by level and subject substantially. Finally, in Figure 12a and Figure 12b, we analyze the relation between accuracy and problem and solution length, and find that problems with long questions or ground truth solutions indeed tend to be more difficult than problems with short questions or solutions.

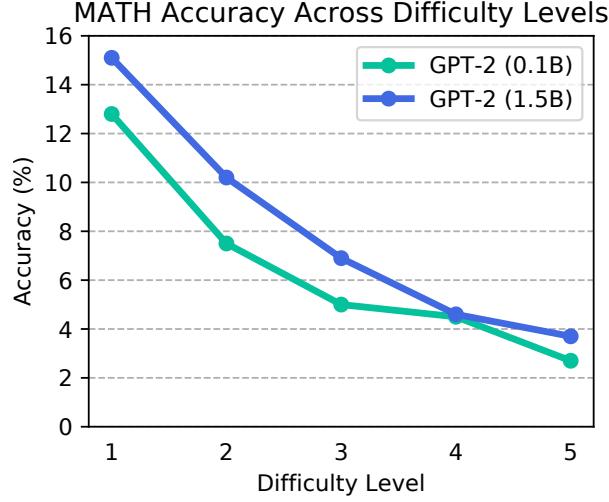


Figure 10: Problems that are more difficult for humans are also more difficult for GPT-2.

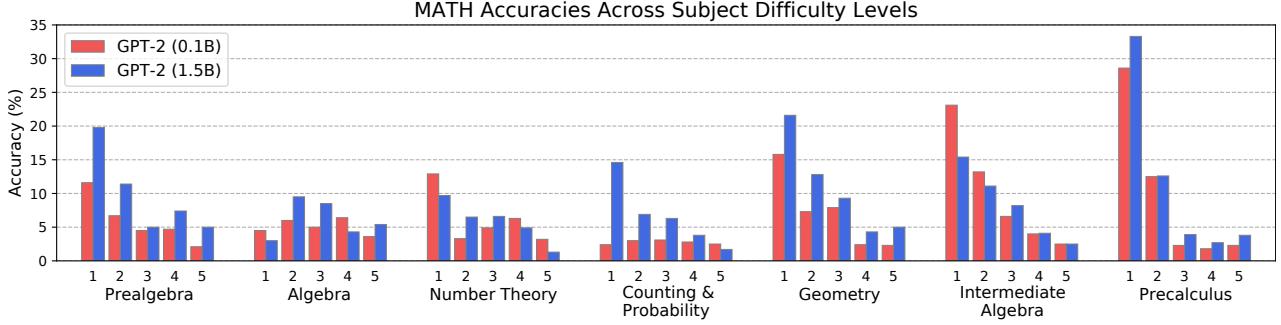
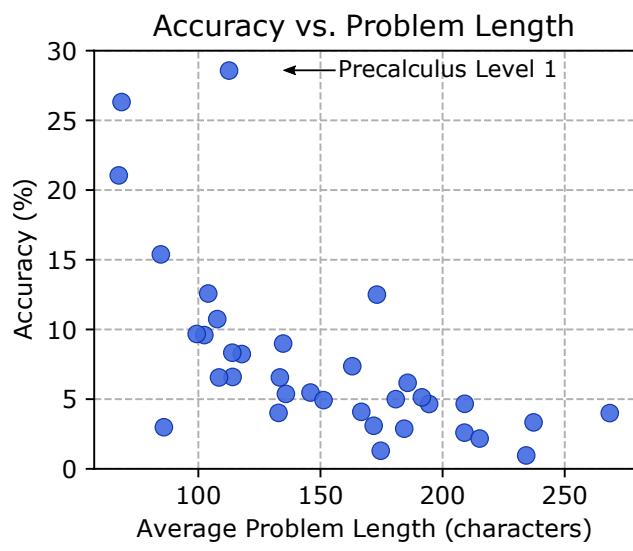
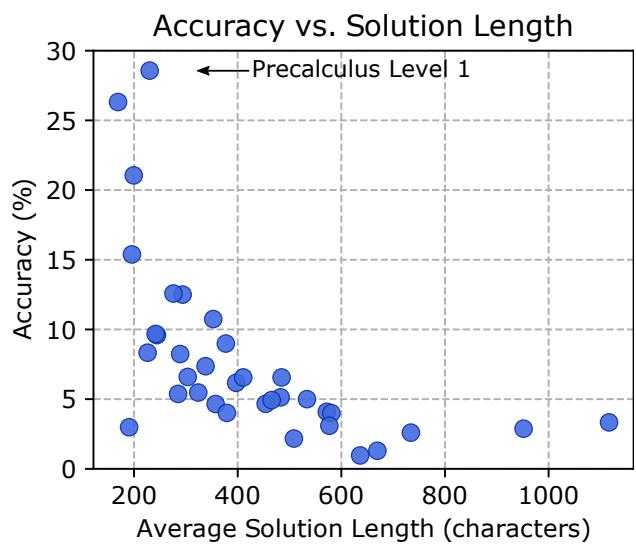


Figure 11: Accuracy per subject per difficulty level.



(a) Subject accuracy vs problem length. Each point represents a subject at a specific difficulty level. We exclude problems with asymptote figures. Results are from GPT-2 (1.5B).



(b) Subject accuracy vs solution length. Each point represents a subject at a specific difficulty level. We exclude problems with asymptote figures. Results are from GPT-2 (1.5B).

**Khan Academy Modules (1/4):** 2 step equations; 2-step addition word problems within 100; 2-step subtraction word problems within 100; 2-step word problems; absolute minima and maxima (closed intervals); absolute minima and maxima (entire domain); absolute value equations; absolute value of complex numbers; add and subtract complex numbers; add and subtract matrices; add and subtract polynomials; add and subtract rational expressions; add and subtract rational expressions: factored denominators; add and subtract rational expressions: like denominators; add and subtract rational expressions: unlike denominators; add and subtract vectors; add 1 or 10; add 1s or 10s (no regrouping); add 3 numbers; add and subtract fractions; add and subtract fractions word problems; add and subtract within 20 word problems; add fractions with unlike denominators; add within 10; add within 1000; add within 20; add within 5; adding and subtracting decimals word problems; adding and subtracting in scientific notation; adding and subtracting negative fractions; adding and subtracting negative numbers; adding and subtracting rational numbers; adding and subtracting decimals word problems; adding and subtracting fractions; adding and subtracting mixed numbers 0.5; adding and subtracting mixed numbers 1; adding and subtracting polynomials; adding and subtracting radicals; adding and subtracting rational expressions 0.5; adding and subtracting rational expressions 1; adding and subtracting rational expressions 1.5; adding and subtracting rational expressions 2; adding and subtracting rational expressions 3; adding and subtracting rational numbers; adding and subtracting with unlike denominators 5; adding and subtracting with unlike denominators 6; adding decimals (hundredths); adding decimals (tenths); adding decimals and whole numbers (hundredths); adding decimals and whole numbers (tenths); adding decimals: thousandths; adding fractions; adding fractions 0.5; adding up to four 2-digit numbers; adding vectors; addition and subtraction word problems; addition and subtraction word problems 2; addition word problems within 100; age word problems; amplitude of sinusoidal functions from equation; analyze concavity; angle addition postulate; angle of complex numbers; approximation with local linearity; arc length; area and perimeter of rectangles word problems; area between two curves; area between two curves given end points; area between two polar curves; area bounded by polar curves; area bounded by polar curves intro; area of a circle; area of parallelograms; area problems; areas of circles and sectors; arithmetic sequences 1; arithmetic sequences 2; arithmetic series; average value of a function; average word problems; basic division; basic multiplication; basic partial derivatives; basic set notation; binomial probability formula; calculating binomial probability; center and radii of ellipses from equation; chain rule capstone; chain rule intro; change of variables: bound; change of variables: factor; circles and arcs; circulation form of green's theorem; classifying critical points; combinations; combined vector operations; combining like terms; combining like terms with distribution; combining like terms with negative coefficients; combining like terms with rational coefficients; complementary and supplementary angles; complete solutions to 2-variable equations; completing the square; completing the square (intermediate); completing the square (intro); complex numbers from absolute value and angle; complex plane operations; composite exponential function differentiation; composite numbers; conditional statements and truth value; construct exponential models; construct sinusoidal functions; continuity at a point (algebraic); converting between point slope and slope intercept form; converting between slope intercept and standard form; converting decimals to fractions 1; converting decimals to fractions 2; converting decimals to percents; converting fractions to decimals; converting mixed numbers and improper fractions; converting multi digit repeating decimals to fractions; converting multi-digit repeating decimals to fractions; converting percents to decimals; converting recursive and explicit forms of arithmetic sequences; converting recursive and explicit forms of geometric sequences; counting 1; counting 2; cube roots; cube roots 2; cumulative geometric probability; defined and undefined matrix operations; definite integral as the limit of a riemann sum; definite integrals of piecewise functions; definite integrals: common functions; definite integrals: reverse power rule; degrees to radians; density word problems; dependent probability; derivatives 1; derivatives of  $a^x$  and  $\log_a x$ ; derivatives of  $\sin(x)$  and  $\cos(x)$ ; derivatives of  $\tan(x)$ ,  $\cot(x)$ ,  $\sec(x)$ , and  $\csc(x)$ ; derivatives of  $e^x$  and  $\ln(x)$ ; determinant of a 2x2 matrix; determinant of a 3x3 matrix; difference of squares; differentiability at a point: algebraic; differential equations: exponential model equations; differentiate integer powers (mixed positive and negative); differentiate polynomials; differentiate products; differentiate quotients; differentiate rational functions; differentiate related functions; differentiating using multiple rules; direct comparison test; direct substitution with limits that don't exist; direction of vectors; disc method: revolving around other axes; disc method: revolving around x- or y-axis; discount, markup, and commission word problems; discount, tax, and tip word problems; disguised derivatives; distance between point and line; distance formula; distributive property with variables; divide by 1; divide by 10; divide by 2; divide by 3; divide by 4; divide by 5; divide by 6; divide by 7; divide by 8; divide by 9; divide complex numbers; divide decimals by whole numbers; divide fractions by whole numbers; divide mixed numbers; divide polynomials by linear expressions; divide polynomials by monomials (with remainders); divide polynomials by x (no remainders); divide polynomials by x (with remainders); divide polynomials with remainders; ...

Figure 13: Khan Academy modules in our AMPS pretraining dataset (Part 1).

**Khan Academy (2/4):** divide powers; divide quadratics by linear expressions (no remainders); divide quadratics by linear expressions (with remainders); divide whole numbers by 0.1 or 0.01; divide whole numbers by decimals; divide whole numbers by fractions; divide whole numbers to get a decimal (1-digit divisors); divide whole numbers to get a decimal (2-digit divisors); divide with remainders (2-digit by 1-digit); dividing complex numbers; dividing decimals 1; dividing decimals 2; dividing decimals: hundredths; dividing decimals: thousandths; dividing fractions; dividing fractions word problems; dividing fractions word problems 2; dividing mixed numbers with negatives; dividing negative numbers; dividing polynomials by binomials 1; dividing polynomials by binomials 2; dividing polynomials by binomials 3; dividing positive and negative fractions; dividing positive fractions; dividing rational numbers; dividing whole numbers by fractions; dividing whole numbers by unit fractions; dividing whole numbers like 56/35 to get a decimal; divisibility 0.5; divisibility tests; domain of a function; double integrals with variable bounds; empirical rule; equation of a circle in factored form; equation of a circle in non factored form; equation of a hyperbola; equation of a parabola from focus and directrix; equation of an ellipse; equation of an ellipse from features; equations and inequalities word problems; equations of parallel and perpendicular lines; equations with parentheses; equations with parentheses: decimals and fractions; equations with variables on both sides; equations with variables on both sides: decimals and fractions; equivalent fractions; estimating square roots; evaluate composite functions; evaluate function expressions; evaluate functions; evaluate logarithms; evaluate logarithms (advanced); evaluate logarithms: change of base rule; evaluate piecewise functions; evaluate radical expressions challenge; evaluate sequences in recursive form; evaluating composite functions; evaluating expressions in 2 variables; evaluating expressions in one variable; evaluating expressions with multiple variables; evaluating expressions with multiple variables: fractions and decimals; evaluating expressions with one variable; evaluating expressions with variables word problems; evaluating logarithms; evaluating logarithms 2; expected value; explicit formulas for arithmetic sequences; explicit formulas for geometric sequences; exponent rules; exponential expressions word problems (algebraic); exponential model word problems; exponential vs. linear growth over time; exponents with integer bases; exponents with negative fractional bases; expressing ratios as fractions; expressions with unknown variables; expressions with unknown variables 2; extend arithmetic sequences; extend geometric sequences; extend geometric sequences: negatives and fractions; extraneous solutions to rational equations; factor higher degree polynomials; factor polynomials using structure; factor quadratics by grouping; factor using polynomial division; factor with distributive property (variables); factoring difference of squares 1; factoring difference of squares 2; factoring difference of squares 3; factoring linear binomials; factoring polynomials by grouping; factoring polynomials with two variables; factoring quadratics 1; factoring quadratics with a common factor; features of a circle from its expanded equation; features of a circle from its standard equation; features of quadratic functions; find area elements; find composite functions; find critical points; find critical points of multivariable functions; find inflection points; find inverses of rational functions; find missing divisors and dividends (1-digit division); find missing factors (1-digit multiplication); find missing number (add and subtract within 20); find the inverse of a 2x2 matrix; find the missing number (add and subtract within 1000); find trig values using angle addition identities; finding absolute values; finding curl in 2d; finding curl in 3d; finding derivative with fundamental theorem of calculus; finding derivative with fundamental theorem of calculus: chain rule; finding directional derivatives; finding divergence; finding gradients; finding inverses of linear functions; finding partial derivatives; finding percents; finding perimeter; finding tangent planes; finding the laplacian; finite geometric series; finite geometric series word problems; foci of an ellipse from equation; fraction word problems 1; fractional exponents; fractional exponents 2; fractions as division by a multiple of 10; function as a geometric series; function inputs and outputs: equation; function rules from equations; gcf and lcm word problems; general triangle word problems; geometric probability; geometric sequences 1; geometric sequences 2; geometric series formula; graphing points and naming quadrants; graphing systems of equations; greatest common factor; greatest common factor of monomials; higher order partial derivatives; identify composite functions; identify separable equations; identifying numerators and denominators; identifying slope of a line; imaginary unit powers; implicit differentiation; improper integrals; increasing and decreasing intervals; indefinite integrals:  $e^x$  and  $1/x$ ; indefinite integrals: sin and cos; independent probability; inequalities word problems; infinite geometric series; integer sums; integral test; integrals and derivatives of functions with known power series; integrals in spherical and cylindrical coordinates; integrate and differentiate power series; integrating trig functions; integration by parts; integration by parts: definite integrals; integration using completing the square; integration using long division; integration using trigonometric identities; integration with partial fractions; intercepts from an equation; interpret quadratic models; interval of convergence; inverse of a 3x3 matrix; inverses of functions; iterated integrals; jacobian determinant; l'hopital's rule (composite exponential functions); l'hopital's rule: 0/0; l'hopital's rule:  $\infty/\infty$ ; lagrange error bound; least common multiple; ...

Figure 14: Khan Academy modules in AMPS (Part 2).

**Khan Academy Modules (3/4):** limits at infinity of quotients; limits at infinity of quotients with square roots; limits at infinity of quotients with trig; limits by direct substitution; limits by factoring; limits of piecewise functions; limits of trigonometric functions; limits using conjugates; limits using trig identities; line integrals in vector fields; linear equation and inequality word problems; linear equations with unknown coefficients; linear equations word problems; linear models word problems; logical arguments and deductive reasoning; maclaurin series of  $\sin(x)$ ,  $\cos(x)$ , and  $e^x$ ; make 10; manipulate formulas; markup and commission word problems; matrix addition and subtraction; matrix dimensions; matrix elements; matrix equations: addition and subtraction; matrix equations: scalar multiplication; matrix row operations; matrix transpose; mean, median, and mode; midline of sinusoidal functions from equation; midpoint of a segment; miscellaneous; model with one-step equations and solve; modeling with multiple variables; modeling with sinusoidal functions; modeling with sinusoidal functions: phase shift; motion along a curve (differential calc); motion problems (differential calc); motion problems (with integrals); multi-digit addition; multi-digit division; multi-digit multiplication; multi-digit subtraction; multi-step linear inequalities; multi-step word problems with whole numbers; multiplication and division word problems; multiplication and division word problems (within 100); multiply and divide complex numbers in polar form; multiply and divide powers (integer exponents); multiply and divide rational expressions (advanced); multiply binomials; multiply binomials by polynomials; multiply binomials intro; multiply by 0 or 1; multiply by 2 and 4; multiply by 5 and 10; multiply by tens word problems; multiply complex numbers; multiply decimals (1 and 2-digit factors); multiply decimals (up to 4-digit factors); multiply difference of squares; multiply matrices; multiply matrices by scalars; multiply mixed numbers; multiply monomials; multiply monomials by polynomials; multiply powers; multiply unit fractions and whole numbers; multiply whole numbers and decimals; multiplying and dividing in scientific notation; multiplying a matrix by a matrix; multiplying a matrix by a vector; multiplying and dividing complex numbers in polar form; multiplying and dividing negative numbers; multiplying and dividing rational expressions 1; multiplying and dividing rational expressions 2; multiplying and dividing rational expressions 3; multiplying and dividing rational expressions 4; multiplying and dividing rational expressions 5; multiplying and dividing scientific notation; multiplying by multiples of 10; multiplying complex numbers; multiplying decimals like  $0.847 \times 3.54$  (standard algorithm); multiplying decimals like  $2.45 \times 3.6$  (standard algorithm); multiplying decimals like  $4 \times 0.6$  (standard algorithm); multiplying expressions 1; multiplying fractions; multiplying fractions by integers; multiplying mixed numbers 1; multiplying negative numbers; multiplying polynomials; multiplying polynomials 0.5; multiplying positive and negative fractions; multiplying rational numbers; multivariable chain rule; multivariable chain rule intro; negative exponents; new operator definitions 1; new operator definitions 2; normal form of green's theorem; number of solutions of quadratic equations; one step equations; one step equations with multiplication; one-step addition and subtraction equations; one-step addition and subtraction equations: fractions and decimals; one-step equations with negatives (add and subtract); one-step equations with negatives (multiply and divide); one-step inequalities; one-step multiplication and division equations; one-step multiplication and division equations: fractions and decimals; operations with logarithms; order of operations; order of operations (no exponents); order of operations 2; order of operations challenge; order of operations with negative numbers; ordered pair solutions to linear equations; p-series; parametric curve arc length; parametric equations differentiation; parametric velocity and speed; partial derivatives of vector valued functions; partial fraction expansion; partial sums intro; particular solutions to differential equations; particular solutions to separable differential equations; parts of complex numbers; percent problems; perfect squares; period of sinusoidal functions from equation; permutations; permutations and combinations; planar motion (differential calc); planar motion (with integrals); polar and rectangular forms of complex numbers; polynomial special products: difference of squares; polynomial special products: perfect square; positive and zero exponents; positive exponents with positive and negative bases; potential functions; power rule (negative and fractional powers); power rule (positive integer powers); power rule (with rewriting the expression); powers of complex numbers; powers of fractions; powers of powers; prime numbers; probabilities of compound events; probability 1; probability in normal density curves; probability of "at least one" success; probability with permutations and combinations; problems involving definite integrals (algebraic); properties of exponents (rational exponents); proportion word problems; pythagorean identities; pythagorean theorem; quadratic word problems (factored form); quadratic word problems (standard form); quadratic word problems (vertex form); quadratics by factoring; quadratics by taking square roots; radians and degrees; radians to degrees; radical equations; radius, diameter, and circumference; range of a function; rate conversion; rate problems; rate problems 2; rates of change in other applied contexts (non-motion problems); rates with fractions; ratio test; ratio word problems; reciprocal trig functions; recursive formulas for arithmetic sequences; recursive formulas for geometric sequences; regroup when adding 1-digit numbers; relate addition and subtraction; related rates (advanced); related rates (multiple rates); related rates (pythagorean theorem); related rates intro; relationship between exponentials and logarithms; relative minima and maxima; remainder theorem; remainder theorem and factors; removable discontinuities; ...

Figure 15: Khan Academy modules in AMPS (Part 3).

**Khan Academy Modules (4/4):** represent linear systems with matrices; represent linear systems with matrix equations; reverse power rule; reverse power rule: negative and fractional powers; reverse power rule: rewriting before integrating; reverse power rule: sums and multiples; rewriting decimals as fractions challenge; right triangle trigonometry word problems; roots of decimals and fractions; sample and population standard deviation; scalar matrix multiplication; scalar multiplication; scientific notation; secant lines and average rate of change; secant lines and average rate of change with arbitrary points; secant lines and average rate of change with arbitrary points (with simplification); second derivative test; second derivatives (implicit equations); second derivatives (parametric functions); second derivatives (vector-valued functions); segment addition; separable differential equations; significant figures; simplify roots of negative numbers; simplify square roots (variables); simplify square-root expressions; simplifying expressions with exponents; simplifying fractions; simplifying radicals; simplifying radicals 2; simplifying rational expression with exponent properties; simplifying rational expressions 2; simplifying rational expressions 3; simplifying rational expressions 4; sinusoidal models word problems; slope-intercept from two points; solid geometry; solutions to quadratic equations; solutions to systems of equations; solve equations using structure; solve exponential equations using exponent properties; solve exponential equations using exponent properties (advanced); solve exponential equations using logarithms: base-10 and base-e; solving equations in terms of a variable; solving for the x intercept; solving for the y intercept; solving proportions; solving quadratics by completing the square 1; solving quadratics by completing the square 2; solving quadratics by factoring; solving quadratics by factoring 2; solving quadratics by taking the square root; solving rational equations 1; solving rational equations 2; special right triangles; square and cube challenge; square roots of perfect squares; standard deviation; standard deviation of a population; stokes' theorem; substitution with negative numbers; subtract decimals (hundredths); subtract decimals and whole numbers (hundredths); subtract within 10; subtract within 1000; subtract within 20; subtract within 5; subtracting decimals (tenths); subtracting decimals and whole numbers (tenths); subtracting decimals: thousandths; subtracting fractions; subtracting fractions with common denominators; subtracting fractions with unlike denominators; subtraction word problems within 100; summation notation intro; sums of consecutive integers; surface integrals to find surface area; switching bounds on double integrals; symbols practice: the gradient; systems of equations; systems of equations with elimination; systems of equations with simple elimination; systems of equations with substitution; systems of equations word problems; tangents to polar curves; taylor and maclaurin polynomials; the derivative and tangent line equations; the divergence theorem; the fundamental theorem of calculus and definite integrals; the hessian matrix; translate one-step equations and solve; trigonometry 0.5; trigonometry 1; trigonometry 1.5; trigonometry 2; triple integrals; two-step equations; two-step equations with decimals and fractions; two-step equations word problems; u-substitution: definite integrals; u-substitution: indefinite integrals; unit circle; unit vectors; use arithmetic sequence formulas; use geometric sequence formulas; use the properties of logarithms; use the pythagorean identity; using the mean value theorem; using the quadratic formula; using units to solve problems; variance; vector word problems; vector-valued functions differentiation; verify solutions to differential equations; vertex of a parabola; volume word problems; volumes with cross sections: squares and rectangles; volumes with cross sections: triangles and semicircles; washer method: revolving around other axes; washer method: revolving around x- or y-axis; word problems with "more" and "fewer" 2; write common decimals as fractions; write common fractions as decimals; write decimals as fractions; write differential equations; write equations of parallel and perpendicular lines; writing basic expressions with variables; writing basic expressions word problems; writing expressions; writing expressions 2; writing expressions with variables; writing expressions word problems; writing functions with exponential decay; writing linear functions word problems; writing proportional equations; writing proportions; wrong statements in triangle proofs; z scores 1; z scores 2; z scores 3; zero product property.

Figure 16: Khan Academy modules in AMPS (Part 4).