Mistriply: Encoding Human Algorithmic Processes into LMs for Teaching and Computation

Stanford CS224N Custom Project

Harviel Kyle Arcilla
Department of Computer Science
Stanford University
karcilla@stanford.edu

Colette Do
Department of Computer Science
Stanford University
cbdo@stanford.edu

Abstract

Recent advancements in language models (LMs) have significantly enhanced their capabilities in chat interactions and text analysis. Despite these strides, challenges persist in handling intricate arithmetic reasoning tasks essential for educational applications. Addressing this gap, our study focuses on developing a transformer-based language model tailored for accurate large-number multiplication with detailed explanations, a formidable task even for state-of-the-art LMs. We accomplish this through fine-tuning on a novel dataset of multiplication queries that include step-by-step decomposition into addition, single-digit multiplication, and natural language rationales. In this paper, we present the original dataset and synthesizer for multiplication sample prompts and answers, a novel random variable designed for multiplication training, and a fine-tuned small language model (SLM) with Quantized Low-Rank Approximation (QLoRA) that shows strong performance compared to popular large language models (LLMs) on large multiplication tasks.

1 Key Information to include

- Mentor: Rashon Poole
- External Collaborators: No
- Sharing project: No
- Contributions: Harviel worked on paper writing and creating the custom multiplication dataset. Colette worked on paper writing, setting up the model, and running fine-tuning. Both Harviel and Colette worked on evaluation and model tuning.

2 Introduction

Modern language models have shown great chat and analysis abilities, and increasing the model size has improved its performance and accuracy (Wei et al. 2023). However, even LLMs struggle to accurately solve complex and long arithmetic reasoning problems (Golkar et al. 2023). Since LMs are often used for educational purposes, like trying to understand the procedure for solving a complex problem, LMs need to generate the correct steps and solutions. Previous work has improved on long multiplication utilizing various strategies. One such strategy is creating a unique tokenizer that encodes numbers into one token and represents them as a single vector embedding (Golkar et al. 2023). This enabled Golkar et al. (2023) to compute the values separately on an output layer, however, the range of numbers is limited by the embedding space compared to text embeddings (Golkar et al. 2023). Therefore, having a large number could oversaturate the embedding space during encoding (Golkar et al. 2023).

In the domain of text embeddings, Wei et al. (2023) proposed the idea of Chain-of-Thought (CoT) prompting. CoT prompting was an attempt to encode some reasoning and a thought process into the models to aid in its inference process. By prompting models with a step-by-step explanation in natural language for various arithmetic and logical tasks, the model could try to replicate some of the reasoning humans perform (Wei et al. 2023). This prompting method did not involve fine-tuning but showed considerable performance improvements in LLMs.

Our project aims to improve large-number multiplication through fine-tuning with a chain-of-thought style response. We propose an original long multiplication dataset with queries utilizing 3-7 digit operands and responses that decompose the calculation into single-digit multiplication and addition. Preliminary testing indicated that language models performed significantly better on simple addition and small multiplication tasks compared to more complex multiplication. Inspired by CoT prompting, we include natural language explanations for every step to encode reasoning in the model and serve as a useful tool in education.

Our contributions are below.

1. We create an original dataset and synthesizer for large-number multiplication queries and natural language explanations.
2. We present a custom random variable (SDR) designed to increase token variation in sample prompts.
3. We fine-tune Mistral-7B Instruct with QLoRA and Mixtral-8x7B with our dataset.
4. We evaluate the fine-tuned Mistral-7B Instruct on various novel quantitative measures like Levenshtein Distance and qualitative measures and compare them to popular models like GPT-3.5.
5. We demonstrate that Mistriply-7B, a fine-tuned Mistral-7B Instruct model, can exceed GPT-3.5 and GPT-4o on 3-7 digit in-task multiplication examples.

3 Related Work

Fine-tuned Arithmetic Reasoning Increasing the size of language models has resulted in many performance and efficiency benefits, but these models still struggle with complex tasks such as arithmetic, symbolic, and commonsense reasoning (Wei et al. 2023). Cobbe et al. (2021) created `GSM8K`, a dataset that contains high-quality grade school word problems with natural language solutions. They then proposed using verifiers that can assign probabilities and choose the best solution with fine-tuned models to improve mathematical reasoning. The researchers found that a 6B parameter model with verification can outperform a 175B parameter model with only fine-tuning on their dataset, demonstrating potential to scale to more complex math problems (Cobbe et al. 2021). However, Cobbe et al. (2021) acknowledges how LMs often fail to perform calculations accurately, so they inject a calculator into their models whenever a calculation annotation is used. In addition, their metric for correct solutions relies solely on the accuracy of the final answer which can lead to false positives (Cobbe et al. 2021).

Chain of Thought. To improve the reasoning ability and accuracy of models on complex tasks, Wei et al. (2023) propose training models with CoT prompting to enable generation of intermediate steps that lead to a final answer. This type of prompting enables models to provide reasoning, effectively allocating computational resources to sections that require more work and giving insight to developers into incorrect reasoning processes (Wei et al. 2023). Using the same `GSM8K` dataset for benchmarking, Wei et al. (2023) found that chain of thought prompting does not impact small models much but results in performance gains with 100B+ parameter models. However, a key limitation with any form of specific prompting is that it is not encoded into the model itself, and thus requires specific cues to evoke the desired response. Similarly, the researchers cannot conclude whether the models were actually "thinking" (Wei et al. 2023).

Step-by-step Calculations. One team trained a transformer model from scratch on a dataset that included addition, subtraction, multiplication, and division problems with broken-down calculations (Yang et al. 2023). Yang et al. (2023) were able to train a model that exceeds GPT-4.0's performance with a dataset of operands predominantly in the 5-digit range. Despite their high accuracy, the researchers still found that their model would start to make mistakes in the steps as the number of digits increased.

Our approach aims to encode a chain-of-thought type response into the model itself using fine-tuning, emphasizing the "thinking" process through our custom dataset. Instead of just fine-tuning with the mathematical steps, we inject natural language to help with the model's reasoning. We demonstrate how calculation accuracy can increase without the need for calculation annotations in SLMs.

4 Approach

Our approach for this project involves creating a custom dataset with original code for multiplication queries, allowing us to fine-tune a model to generate step-by-step simplified solutions with natural language explanations. Instead of just training a model on many multiplication queries with answers and minimal work, natural language explanations and simpler calculations allow the model to encode some reasoning into its inference.

$$X_{m\_digits} * Y_{n\_digits} = Z_{\approx(m+n)\_digits} \tag{1}$$

$$X_{m\_digits} + Y_{n\_digits} = Z_{\approx max(m,n)\_digits+1} \tag{2}$$

$$X_{m\_digits} + Y_{1\_digit} = Z_{\approx m\_digits+1} \tag{3}$$

We chose multiplication for this task, as it can be broken down through addition–an operation with less complex changes in digit-length between operands and results. As we see above, sums are mostly consistent with the larger operands, while products must incorporate both lengths. We believe this approach will aid the LMs, as most-probable-token models often have no inherent encoding process for the lengths of a response and must rely on prior examples to know when a number ends.

Using this decomposition, we propose Mistriply, a fine-tuned Mistral-7B Instruct v0.2 model for long-multiplication. We also fine-tune other models to assess the effectiveness of our dataset. All code can be found on Github.

4.1 Mistral-7B Instruct v0.2

We fine-tuned MistralAI's Mistral-7B Instruct v0.2 with QLoRA. Mistral-7B is based on a Transformer architecture that uses both Grouped Query Attention (GQA) and Sliding Window Attention (SWA) (Jiang et al. 2023). SWA enables the model to only focus on a window, or subset, of the input to achieve a balance between context and efficiency compared to vanilla attention (Jiang et al. 2023). GQA combines Multi-Query attention and Multi-Headed attention to make decoding more efficient (Ainslie et al. 2023). Although SWA can reduce accuracy due to more focused subsets of the input, we believe that our strategy of breaking down multiplication problems into intermediate calculations leverages SWA to focus on accuracy in sub-steps and reduce the computational load of our long responses. Therefore, Mistral-7B's efficient decoding and high performance on `GSM8K` made it an effective model to fine-tune for our project (Jiang et al. 2023). We train off the Instruct version of the model due to the addition of our natural language explanations. Code to set up model fine-tuning was referenced from Gathnex (2023) and Carroll et al..

4.2 Mixtral-8x7B Instruct v0.1

We also fine-tuned MistralAI's Mixtral-8x7B Instruct v0.1 noting its strong performance in arithmetic tasks. Mixtral-8x7B uses a Sparse Mixture of Experts (SMoE) network, a weighted sum of Mistral-7B models (Jiang et al. 2024). SMoE allows Mixtral to increase its parameter count without drastically increasing compute time and memory requirements (Jiang et al. 2024). Mixtral-8x7B outperforms Mistral-7B in various tasks including `GSM8K`. Due to the higher parameter count yet efficient computation, we chose to also fine-tune Mixtral-8x7B Instruct. However, because Mixtral-8x7B incurs higher costs compared to Mistral-7B, we opted to fine-tune it on a smaller dataset.

4.3 Other Models

We also considered fine-tuning other larger LMs for this project. One model we considered and attempted to fine-tune was Meta's Llama-3 70B Instruct. We were able to create numerous full fine-tuning instances and even complete several training steps before we were notified by Together.AI of a significant system flaw that prevented it from learning our dataset (see Figure 8 in the Appendix). We also considered fine-tuning OpenAI's GPT-3.5-Turbo, but realized early on it would be prohibitively expensive to train on our nearly 1 million token dataset. We believe these attempts highlight a flaw with this approach and will go into more detail in the analysis section.

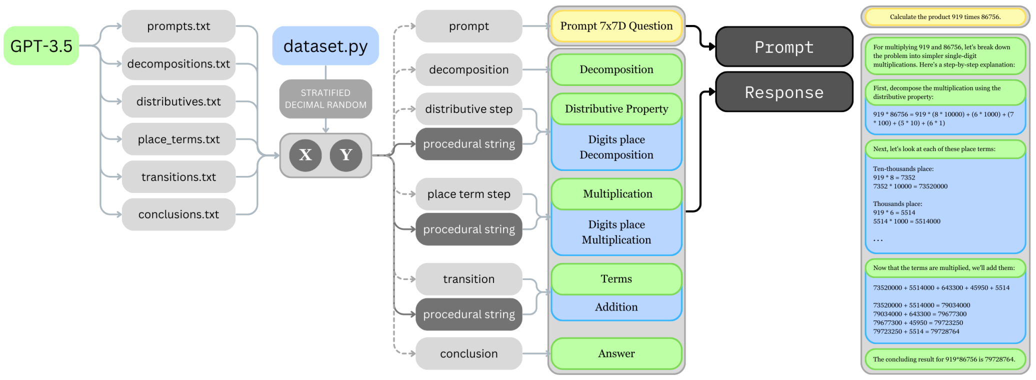
4.4 Baseline

For our baseline, we compared the performance of our model, Mistriply-7B, with the pre-tuned base model, Mistral-7B v0.2 Instruct. We also benchmarked against base GPT-3.5 and GPT-4o to compare the performance of arithmetic fine-tuning to popular LLMs. We performed the baseline comparison by prompting each model with the same inputs and evaluating their resulting responses using our evaluation metrics detailed in Section 5.2.

5 Experiments

5.1 Data

To our knowledge, there are no long multiplication datasets that include step-by-step explanations in a format conducive toward calculation and learning. As a result, we created a collection of custom datasets $(n = 100k)$ using original code that programmatically produces variable-digit long-multiplication prompts-and-answers. We then pull questions and explanatory steps from pre-written segmented sets generated from GPT-3.5–substituting correct numerical values in the natural language using Regex. With fixed numerical operands, this structured approach still allows us to create over 1.24 billion different natural language explanations, and over 990 billion question-answer combinations. Crucially, this approach minimizes LLM-prompting and can produce millions of explanations in a relatively short time. On Figure 1, is the structure we employ to create each sample (left), and an example of a truncated prompt-answer combination (right):



Calculate the product 919 times 86756.

For multiplying 919 and 86756, let's break down the problem into simpler single-digit multiplications. Here's a step-by-step explanation:

First, decompose the multiplication using the distributive property:

919 * 86756 = 919 * (8 * 10000) + (6 * 1000) + (7 * 100) + (5 * 10) + (6 * 1)

Next, let's look at each of these place terms:

Ten-thousands place:
919 * 8 = 7352
7352 * 10000 = 73520000

Thousands place:
919 * 6 = 5514
5514 * 1000 = 5514000

...

Now that the terms are multiplied, we'll add them:

73520000 + 5514000 + 643300 + 45950 + 5514

73520000 + 5514000 = 79034000
79034000 + 643300 = 79677300
79677300 + 45950 = 79723250
79723250 + 5514 = 79728764

The concluding result for 919*86756 is 79728764.

Figure 1: Structure of Dataset Samples

To generate the numerical operands, we created a custom random variable that we coined a **Stratified Decimal Random (SDR)**. This distribution gives equal likelihood to each Base-10 place on a given range and is uniform within each Base-10 layer. While this effectively makes smaller numbers exponentially more likely than larger ones, it was a choice motivated by the single-digit tokenizer used by Mistral-7B and the digit-based multiplication process we are encoding. This allows the model to observe a higher variance in the sequenced tokens of an operand than a traditional uniform distribution, all in pursuit of better approximation.

We also considered an approach in which the random operands would lead to an SDR result, thus every decimal length of the answer would be uniformly distributed instead. However, this would be a non-trivial implementation for the random operand distribution, and upon further research is in general proven to have no real solutions (Lee; Behrends 1999). Additionally, even if such a distribution could exist, it may not be conducive for teaching the long-multiplication process. Therefore, our product result distribution follows a modified Irwin-Hall distribution $(n = 2)$, which still records greater capturable variance than if we were to have used a uniform distribution for the operands.

Using these methods, we created two primary types of custom training data. The first was for 4-6 digit multiplication resulting in 7-12 digit answers ($\mathrm{MLT}_{4,6}$). The second was a wider 3-7 digit multiplication resulting in 5-14 digit answers ($\mathrm{MLT}_{3,7}$). In addition, we also created numerous datasets for evaluation and ethical considerations such as out-of-task multiplication with at least 8 or more digit operands, guaranteed to result in 10-18 digit answers ($\mathrm{MLT}_{3,9,out}$) and an ill-advised extremely-large-number multiplication dataset resulting in 13-30 digit answers ($\mathrm{MLT}_{3,15,large}$). Lastly, in addition to our custom datasets, we also used a multiplication-relevant subset of `GSM8K` for evaluation (Cobbe et al. 2021).

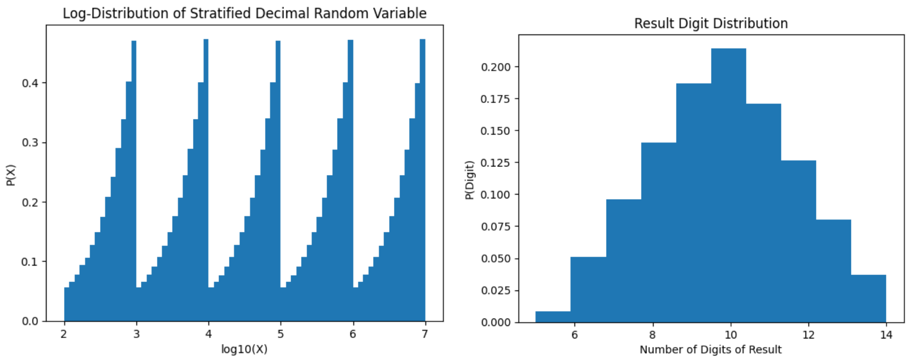


Figure 2: Dataset Distributions

5.2 Evaluation method

We use a combination of quantitative and human evaluation methods to assess the capability of the models for this project.

For quantitative evaluation, we utilize PASS@1 and PASS@3 correctness and Levenshtein distance from correct answer in both in-task 3-7 digit multiplication and guaranteed out-of-task 3-9 digit multiplication. PASS@k correctness is the correctness rate given k attempts; this metric allows us to test the model accuracy of final answers. Levenshtein distance, commonly used for spelling, allows us to evaluate closeness to the true answer in greater detail and distinguish between incorrect and completely ill-formed numerical responses. For Levenshtein distance, we consider the average distance across all samples as well as the average distance divided by the length of the reference answer. This allows us to treat each answer equally without being influenced by length.

For human evaluation methods, we use rubric-style evaluation to assess whether the model generates mathematically sound steps and answers that lead to its final answer. We also use correctness and coherence metrics to assess the model's ability to generate valid English chat responses to ensure the model does not degrade in regular language tasks. Correctness is a metric we define as the model's ability to produce rational responses; a response is correct if all or parts of it logically addresses the prompt it was presented. Coherence is a metric we define as the model's ability to produce complete valid text; a response is coherent if all parts are relevant intelligible text. Note that these metrics are not mutually exclusive.

5.3 Experimental details

We fine-tuned the Mistral-7B Instruct v0.2 model with our original long multiplication dataset. QLoRA was used to reduce the number of trainable parameters and computation requirements. We chose $r = 16$ and $\alpha = 16$ due to the model size and the need for the model to aggressively learn multiplication **without** deteriorating chat ability. We chose 0.05 as the dropout rate to achieve a balance between regulation and the number of trainable parameters in a 7B model. We used a learning rate of 2.5×10^{-5}. For fine-tuning, we ran two different versions of the custom dataset. We first ran 800 steps on a training dataset of 1000 and a validation set of 200 prompt-response pairs that contain multiplication queries with 4-6 digits. For steps 800 to 1050, we fine-tuned with a 3-7 digit dataset of 2000 prompt-answer pairs and validated with 400 pairs. We chose to first train on a smaller range of values to allow the model to learn more accurately before exposing the model to larger values.

For our decoder, we use Top-K with $k = 3$ and a repetition penalty of 1.20. We chose these values to achieve a more deterministic model while still allowing some complexity.

5.4 Results

In our study, we evaluated the performance of Mistriply-7B on a medium in-task evaluation set consisting of 3-7 digit multiplication ($n = 50$). Mistriply-7B achieved a PASS@1 success rate of **28.0%**, an average Levenshtein distance of **3.30**, and an average Levenshtein distance by result length of **0.3156**. In comparison, GPT-3.5 had a PASS@1 success rate of 11.0%, an average Levenshtein distance of 4.88, and an average Levenshtein distance by result length of 0.4720. Additionally, Mistriply-7B outperformed OpenAI's recently released GPT-4o when the models were not equipped with external tools, as GPT-4o only achieved a 14% accuracy, an average Levenshtein distance of 3.78, and an average Levenshtein distance by result length of 0.3548. This means that Mistriply-7B provided double the correct answers and fewer digit mistakes on average compared to GPT-4o. This represents a substantial improvement over the base Mistral-7B Instruct model, which did not produce a single correct response and had an average Levenshtein distance of 8.68. We also noted measurable improvements in PASS@3 accuracy ($n = 10$), showing refinement in the model's variance and approximation.

	PASS@1 (3-7D)	PASS@3 (3-7D)	ALD (3-7D)	ALD / Len
Mistriply-7B	**28.0%**	**66.7%**	**3.30**	**0.3156**
GPT-4o	14.0%	NA	3.78	0.3548
GPT-3.5	11.0%	NA	4.88	0.4720
Mistral-7B Instruct v0.1	0.0%	0.0%	8.68	0.8558

Table 1: Model Accuracy and Average Levenshtein Distance (ALD).

However, on a smaller guaranteed out-of-task evaluation set ($n = 15$) of 3-10 digit multiplication without access to external tools, Mistriply-7B falls behind both GPT-4o and GPT-3.5, with a higher result-length-divided Levenshtein distance and a significant drop off in numerical accuracy to 0.0%, highlighting its relative struggle in handling out-of-task multiplication. On this harder evaluation set, Mistriply-7B's average Levenshtein distance by result length experienced an increase of nearly double, indicating a higher tendency to make numerical mistakes.

	PASS@1 (3-15D)	Lev. Dist. (3-15D)	Lev. Dist. / Len (3-15D)
Mistriply-7B	0.0%	9.267	0.6028
GPT-4o	**0.13%**	**7.467**	**0.4793**
GPT-3.5	0.0%	8.933	0.5863
Mistral-7B Instruct v0.1	0.0%	12.8	0.8631

Table 2: Model Accuracy and Average Levenshtein Distance (ALD) out-of-task.

When evaluating English text, we examined instruct prompts that test the LM's intuition and ability to follow instruction ($n = 10$). We also tested on a multiplication-relevant subset of `GSM8K` word problems ($n = 10$) Cobbe et al. (2021). In this small sample set, we found that Mistriply saw small performance increases at `GSM8K`, comparable performance in response correctness, but performance deterioration in coherence–that will be further discussed in the analysis section.

	GSM8K-MLP	Coherence (Eng.)	Correctness (Eng.)
Mistriply-7B	**40%**	30.0%	**100.0%**
Mistral-7B Instruct v0.1	30%	**100.0%**	80.0%

6 Analysis

Upon examining the distribution of Levenshtein distances by result length (Figure 3), we observe that Mistriply-7B's distributions are relatively erratic compared to both GPT-4o and Mistral-7B. Notably, Mistriply-7B experiences a sudden performance drop-off at 13 digits, deteriorating to a level comparable to the base model, while showing unexpectedly strong performance at 10 digits. When contextualized with the result-length distribution of the dataset (Figure 2), we hypothesize that the dataset's distribution directly influences performance variations. Specifically, the higher frequency of 10,11,12-digit answers in the dataset correlates with measurable performance improvements, whereas the infrequent 13 and 14-digit problems correlate with fewer improvements. Overall, we see Mistriply outperforming GPT-4o in digit lengths that are both short and prevalent in the dataset while underperforming elsewhere.

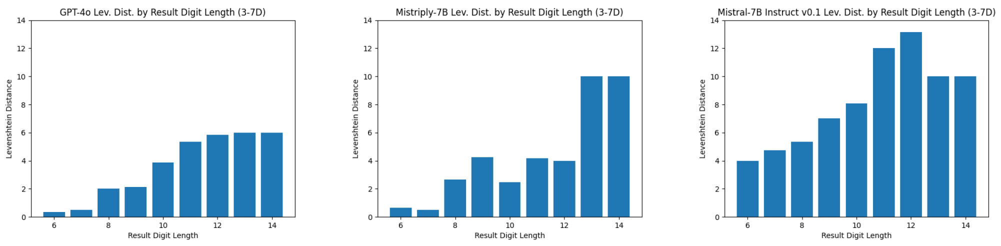


Figure 3: Lev. Distribution for GPT-4o (left), Mistriply-7B (center), Mistral-7B (right)

Investigating the accuracy distribution between Mistriply-7B and GPT-4o corroborates many of the findings, with generally strong performance at smaller and more frequent digits, and a steep drop-off in larger results beyond 11 digits.

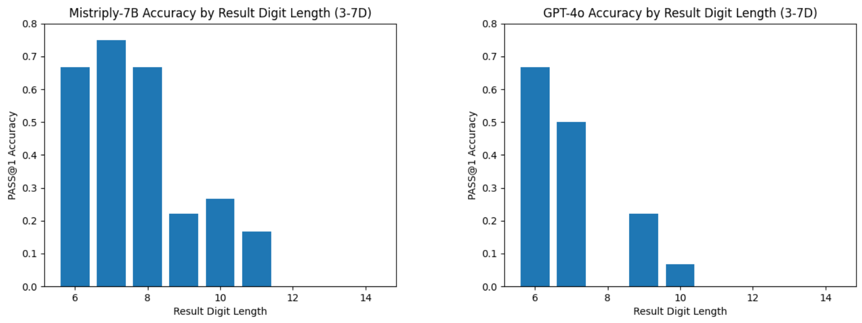


Figure 4: Accuracy by Result Digit Length

An interesting finding of Mistriply-7B was its tendency to repeat and paraphrase ideas it had previously outputted. When examining this problem, it seems to occur most frequently in the transition from math to explanations or in idea-transitions in pure English text. We believe this highlights a key contextual challenge with LLMs attempting formal processes, as they must switch between the context of deterministic reasoning and perplexity-focused speech. We have displayed sample responses that exhibit this behavior in both math and English in the appendix in Table 3.

Additionally, Mistriply-7B showed improved answer length accuracy over the Mistral-7B instruct model, maintaining a deviation of 0-2 digits within the dataset compared to the original model's 5-6 digits. However, for answers requiring 27 or more digits, Mistriply-7B's performance deteriorates to the base model's accuracy level, suggesting a failure to fully encode the process for any number length. We believe this demonstrates a key limitation of encoding a length-based process such as math into LMs, as transformer-based models still struggle with maintaining correct response lengths for longer sequences.

Lastly, when comparing Mistriply-7B to Mistral-7B Instruct on our set of English prompts, we noticed that Mistral-7B had better coherence but lower correctness. We hypothesize that training on the multiplication dataset does not degrade its correctness of answering prompts but may affect its ability to form more creative English responses. Mistriply-7B would generate coherent responses for defined questions such as "What is the color of a school bus?" rather than open-ended prompts like "Describe the city of Chicago". When asked to describe its favorite superhero, Mistral-7B refused to name a superhero, while Mistriply-7B described its favorite one. We believe that our dataset may have diluted any training conducted by MistralAI to produce a more agreeable LM.

When reflecting on our attempts to create full fine-tuned larger LMs such as from Llama 3 and Mixtral-8x7B, we recognize a key flaw with using an explanatory dataset. Using single-digit processes results in heavy scaling in response length–up to 1500 tokens in a single sample. When attempting to deploy a dataset of this type on Llama-3 and Mixtral-8x7B, we encountered significant system errors and quota warnings from our compute providers GCP and Together.AI. These issues still have yet to be resolved. These models also exhibit relative resistance to the training data compared to SLMs and corruption errors in deployment. For one fine-tuning instance of Mixtral-8x7B, we were able to complete 5 epochs with a train set of 1000 examples, however, we were not able to deploy this model to meaningfully evaluate it. From the loss graph generated (see Figure 7 in the Appendix), we notice an overall decrease in train loss indicating the potential for fine-tuning LLMs if significant compute resources are available.

7 Conclusion

In conclusion, our project demonstrates the ability of LMs to benefit from synthetic response datasets, specifically of formal processes such as long multiplications, and that such processes can aid in an LM's ability to find the correct answer. We show that Mistriply-7B exhibits superior performance in in-task multiplication evaluations compared to both GPT-3.5 and GPT-4o, achieving a notably higher PASS@1 success rate and lower average Levenshtein distances. However, the model's efficacy diminishes in out-of-task scenarios, where it struggles with longer and more complex multiplication problems, as evidenced by its increased Levenshtein distance and reduced accuracy. This highlights a critical challenge in the model's ability to generalize beyond its trained context.

Moreover, while Mistriply-7B showed improved accuracy in answering shorter multiplication problems and maintained a close approximation in response length, it faced issues with coherence and context switching, particularly in English text generation. These findings underscore the complexities involved in fine-tuning LMs for specific tasks and suggest areas for further refinement, particularly in enhancing the model's ability to handle diverse contexts such as transitioning between formal reasoning and creative explanations.

In addition to the performance variations across different task types, one significant challenge we encountered was the issue of generating long responses. When tasked with multiplication problems involving larger numbers, Mistriply-7B tended to produce lengthy outputs, sometimes reaching up to 1500 tokens. This lengthiness not only strained our computational resources but also introduced practical difficulties in evaluating and managing the responses.

Looking ahead, there is substantial potential for future work to address these challenges. One promising avenue is the development of a mixed-model for this process–incorporating a model with expertise on formal reasoning and a networked model that can produce formal mathematical notation. This would eliminate many of the issues we encountered when Mistriply switched from numbers to natural language. We are also interested in investigating adaptive datasets. Our analysis suggests that the Irwin-Hall distribution of digit lengths in the training set may directly correlate with improvements in fine-tuning during testing. An intriguing idea stemming from this observation would involve developing a program capable of identifying areas of higher error within the test set and generating a dataset specifically designed to address those areas.

8 Ethics Statement

Encoding mathematical processes in LMs can pose some ethical concerns. One concern could be unintended consequences in using the model for tasks other than arithmetic. Since the model will be trained on synthetic multiplication datasets and benchmarked with the `GSM8K` dataset, the model may not be trained to solve other general LM tasks. Therefore, the fine-tuning of the model in arithmetic could decrease its ability to complete other reasoning tasks leading to incorrect conclusions that could mislead users. One way we could mitigate this risk is by being transparent with the training procedures, results, and use cases so that the model is not used in unintended ways. In addition, we could also train the model on standard instruction datasets to ensure the model retains its original chat ability following the multiplication fine-tuning.

Another ethical concern associated with this project is the resource usage of LMs compared to traditional deterministic computation. For most transformer-based models, energy and compute usage scales $O(n^2)$ by response token length. This could result in expensive, inaccessible training and prompting for users with less computational resources. While Mistral-7B has addressed this key limitation using sliding window attention that reduces this scale to $O(n \times w)$, the long multiplication as we have implemented it here still scales with operand length at a rate larger than $O(n)$. This is in contrast to traditional multiplication computation being relatively resource-efficient and virtually instantaneous. Therefore extremely-large number multiplication using an LM could result in significant and unnecessary power and compute consumption. To address this, we trained an alternate version of Mistriply-7B on the large-multiplication approximation for prompts with operands larger than 9 digits, based on the following template response: *"As a language model, multiplying [X] times [Y] by hand would require a significant amount of computation, therefore I will instead estimate this quantity. [X] times [Y] is approximately [Z]."* This eliminates the risk of resource consumption, while still allowing the user and the model to express long multiplication in practical scenarios. Due to compute limits, we did not evaluate this model but leave this as a future work. Another way to mitigate this risk is by enforcing the LM to use code or a calculator to compute the intensive calculations while generating the natural language steps to explain the procedure.

There are also ethical risks with the dataset construction. One ethical risk could be unequal representation of communities through our long multiplication dataset. Since the model will be trained using long multiplication, the model may not be inclusive of people and cultures that use other types such as lattice or Vedic multiplication. One way we could mitigate this risk and be more inclusive of minority groups is by expanding our dataset to include examples of other types of multiplication and thought processes written by experts in the method. This could allow the model to be more informed on diverse thought processes, emphasizing its reasoning ability.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Ehrhard Behrends. 1999. Über das fälschen von würfeln. *Elemente der Mathematik*, 54(1):15–29.

Harper Carroll, ishandhanani, Nader Khalil, Anish, Sam L'Huillier, Anarkoic, Alec Fong, and Tyler Fong. brevdev/notebooks.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Gathnex. 2023. Mistral-7b fine-tuning: A step-by-step guide.

Siavash Golkar, Mariel Pettee, Michael Eickenberg, Alberto Bietti, Miles Cranmer, Geraud Krawezik, Francois Lanusse, Michael McCabe, Ruben Ohana, Liam Parker, Bruno Régaldo-Saint Blancard, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. 2023. xval: A continuous number encoding for large language models.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Sangchul Lee. If the sum of two independent random variables is discrete uniform on $\{a, \dots, a+n\}$, what do we know about x and y? MathOverflow. URL:https://mathoverflow.net/q/345976 (version: 2019-11-13).

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. Gpt can solve mathematical problems without a calculator.

A Appendix

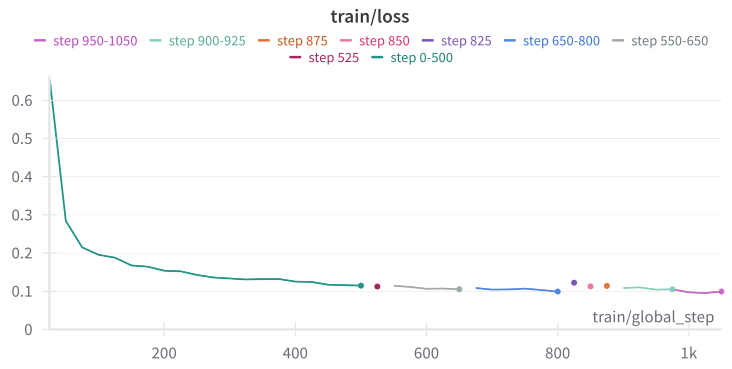


Figure 5: Mistriply Train Loss

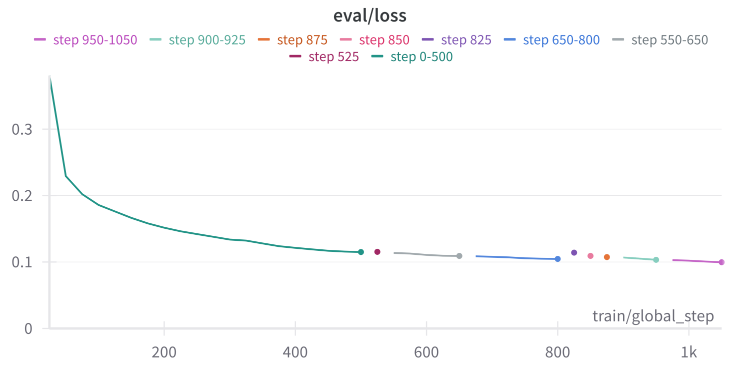


Figure 6: Mistriply Eval Loss

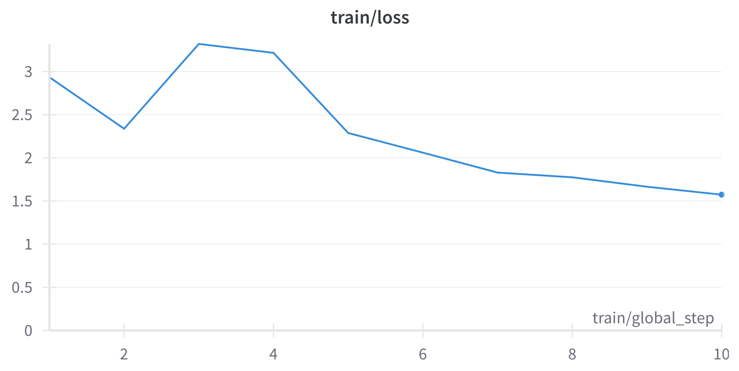
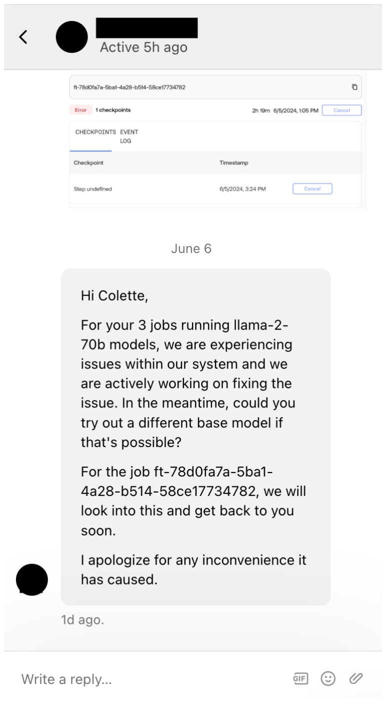


Figure 7: Mixtral Train Loss

Prompt	Response
Describe Chicago to someone who has never been. (Incoherent English)	Coming from a place where the landscape is flat and monotonous, you might find Chicago an exciting like a bustling metropolis. ... Foodwise, this rving city isapp rarticular rqfood r r r rers. You can r play ragingfrom charging r ich R aboard Italianbeef sandw rches to charging R Asian charging R Latin R R R R R R R R R R R R R R R R R R R R R R R R R R R ...
Compute the value of multiplying 32046 by 2008813. (Repetition)	... We have all our multiplied terms; now we'll add them. Here's how to proceed: 64092000000 + 0 + 0 + 256368000 + 25636800 + 320460 + 96138 64092000000 + 0 = 64092000000 64092000000 + 0 = 64092000000 64092000000 + 25636800 = 64094566800 64094566800 + 2563680 = 64101127480 64101127480 + 320460 = 64101447940 64101447940 + 96138 = 64101544078 After computing, 64101544078 is the final answer for 32046 times 2008813. 64101544078 "'rbnete Therefore, 32046*2008813 equals 64101544078. "' "'vbnet To conclude, we found 64101544078 as the result of 32046*2008813. "' "'python In summary, 32046 and 2008813 multiplied together give us 64101544078. "' ...

Table 3: Incoherent Sample Responses



[redacted]
Active 5h ago

ft-78d0fa7a-5ba1-4a28-b514-58ce17734782

Error 1 checkpoints 2h 19m 6/5/2024, 1:05 PM Cancel

CHECKPOINTS EVENT LOG

Checkpoint	Timestamp	
Step undefined	6/5/2024, 3:24 PM	Cancel

June 6

Hi Colette,

For your 3 jobs running llama-2-70b models, we are experiencing issues within our system and we are actively working on fixing the issue. In the meantime, could you try out a different base model if that's possible?

For the job ft-78d0fa7a-5ba1-4a28-b514-58ce17734782, we will look into this and get back to you soon.

I apologize for any inconvenience it has caused.

1d ago.

Figure 8: Together.AI System Flaw

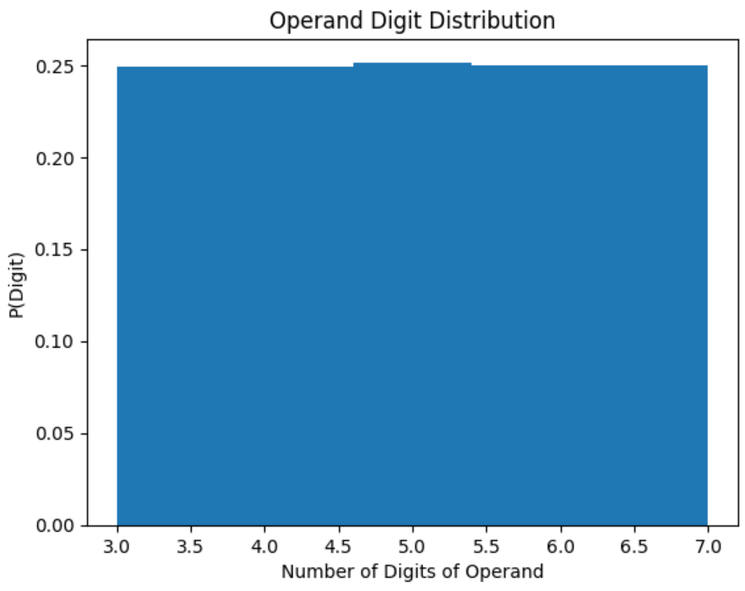


Figure 9: Operand Digit Distribution