

# Math-Aid

Fine tuning an LLM to solve grade school math problems!

<a href="#">1. Overview</a>	<a href="#">2</a>
<a href="#">2. Experiment Setup</a>	<a href="#">3</a>
<a href="#">3. Experiment Results</a>	<a href="#">7</a>
<a href="#">4. Discussion</a>	<a href="#">8</a>
<a href="#">5. Conclusion</a>	<a href="#">9</a>
<a href="#">6. References</a>	<a href="#">9</a>
<a href="#">7. Team</a>	<a href="#">9</a>

# 1. Overview

Our project aims to enhance mathematical problem-solving through the fine tuning of large language models. By leveraging the GSM8K dataset, which is composed of 8.5K diverse grade school math word problems, the objective is to develop an interactive chatbot capable of not just solving mathematical equations but also elucidating solutions in a user-friendly manner. This project addresses the challenge of students struggling with math word problems, aiming to improve comprehension and problem-solving skills. Motivated by the need to enhance engagement in learning mathematics, the project not only provides solutions but also delivers them in a user-friendly, natural language format.

The project delves into the challenging intersection of language understanding and mathematical reasoning, pushing the boundaries of machine learning's role in educational technology. By catering to a foundational educational need, this project has the potential to significantly impact how students engage with and comprehend math problems. The broader societal impact lies in fostering better mathematical understanding among students, potentially boosting academic performance and confidence in mathematical problem-solving.

The approach outlined comprises a series of stages, starting with the initial exploration and pre-processing of the dataset. Subsequent steps involve utilizing base models for inference, such as GPT2 and BERT, fine-tuning models for conditional generation, implementing prompt engineering, and finally, integrating evaluation metrics and an interactive interface using Gradio.

This approach leverages the strengths of pre-trained models, tailoring them to our specific dataset through fine-tuning. The incorporation of prompt engineering serves as a crucial element in guiding the model's behavior, particularly in enhancing its reasoning abilities. By doing this, we aim to equip the model with the capability to not only solve equations but also articulate the step-by-step reasoning process.

We anticipate that this approach could effectively enhance the model's reasoning ability. The rationale behind this approach lies in leveraging pre-trained models and implementing prompt engineering aligns with our goal of improving the model's proficiency in tackling grade school math word problems. The sequential nature of the proposed approach aims to create a symbiotic relationship between the model's understanding of language and its capacity for mathematical reasoning, ultimately contributing to a more effective problem-solving tool.

The novelty of this approach stems from the focus on language understanding and mathematical reasoning, a relatively newer area in machine learning. While prior references exist for solving mathematical problems, the distinction lies in this project's emphasis on natural language explanations and interactive interfaces, which are crucial for educational applications.

Key components of the approach include employing pre-trained models for inference, fine-tuning models on the math dataset, incorporating prompt engineering to guide the model's reasoning, and integrating evaluation metrics for performance assessment. The results of the

fine-tuned language model (LLM) on the math dataset reveal a promising capability in decoding the steps required to reach a final answer. Notably, the generation of intermediate steps demonstrates the model's proficiency in reasoning. However, a notable limitation is in the effective execution of mathematical computations, leading to inaccuracies in the final answers.

## 1.1 Usage

You can either use the pre-trained model directly or engage with the model through the user-friendly chat interface. To engage the model through the user-friendly chat interface, create a conda environment and install the required packages listed in the `requirements.txt` file.

You can use the below commands:

```
conda create --name your_env_name python=3.8
conda activate your_env_name
pip install -r requirements.txt
```

## 2 Experiment Setup

Dataset:

### Overview:

The GSM8K (Grade School Math 8K) dataset comprises 8.5K linguistically diverse grade school math word problems, specifically designed to facilitate question answering involving multi-step reasoning. The dataset was crafted to cover problems ranging from 2 to 8 steps in complexity, requiring elementary calculations with basic arithmetic operations (+ - × ÷) for solution. Notably, the problems are designed to be within the grasp of a proficient middle school student, demanding no conceptual understanding beyond early Algebra.

**Structure:** The dataset instances are structured with a 'question' field containing a grade-school level math question and an 'answer' field providing the solution in natural language. The solutions include multiple steps of reasoning and calculator annotations within double angle brackets (<< >>). To obtain the final numeric solution for a particular question, one needs to parse the completion and extract the numeric value immediately following the ##### token.

Example:

```
{
  'question': 'Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?',
  'answer': 'Natalia sold 48/2 = <<48/2=24>>24 clips in May.\nNatalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.\n##### 72',
```

}

The dataset is divided into training and test sets, with 7473 instances in the training set and 1319 instances in the test set. The selection of this dataset is driven by its inclusion of reasoning steps, enabling the training of models to enhance their reasoning abilities.

## 2.1 Implementation

Initially, our approach was to treat the problem as a Question Answering task, given our intention to answer questions based on the reasoning steps provided in the answer context. However, upon closer examination, we realized that this approach assumed the solution to be present within the context, limiting its applicability to a variety of problems. Consequently, we opted for a Casual Language Modeling Task, treating it as a text generation problem.

In our exploration, we employed Hugging Face pipelines with base models such as GPT-2, T5, and BERT to gauge their performance in generating responses for the given math word problems. Among these, BERT and T5 stood out as they provided responses closely related to the questions.

Further research led us to Flan-T5. Flan is a pre-training method based on prompting. Flan-T5 are T5 models fine-tuned on a mixture of tasks on the Flan collection of datasets. Flan-T5 models come in various sizes, ranging from small to xxl. We began with the base model and attempted to fine-tune it on the GSM8K dataset.

### Details:

1. We formatted the dataset by structuring each question as ``Question: {original\_question}``. The answer was annotated to indicate the reasoning steps, identifying it with ``Steps:`` and the final answer, with ``Answer:``. Following this, tokenization was applied separately to questions and answers, ensuring uniform length through padding. This processed input is then given to the model during the training phase.

The parameters used for training:

```
learning_rate: 1e-04
train_batch_size: 4
eval_batch_size: 4
test_batch_size = 4
optimizer: AdamW
lr_scheduler_type: linear
num_epochs: 3
max_length = 150
num_warmup_steps=0
```

`training_steps = 3* no_of train batches`

## 2.2 Evaluation

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is the metric used for evaluation. It operates by comparing automatically generated outputs(model generated text-solution)) against reference outputs, which are typically human-produced summaries or translations('answer' field).

ROUGE computes precision and recall scores based on the overlap between the generated and reference outputs. This overlap is measured at different levels, such as unigrams (rouge1), bigrams (rouge2), and longest common substrings (rougeL and rougeLsum). In the context of math word problems, where the answers are accompanied by reasoning steps and explanations, the quality of the generated response goes beyond simple token overlap. ROUGE-Lsum, by considering the longest matching sequences over the entire summary, is well-suited for capturing the coherence and relevance of the generated reasoning steps and explanations.

In our initial exploration with a base language model, we observed commendable abilities in problem comprehension for simple math word problems. However, the model faced challenges in accurately generating equations, particularly for more complex questions. To address this, we decided to upgrade to the flan-t5-large model, equipped with 783 million parameters.

The sheer size of the flan-t5-large model posed computational challenges for traditional fine-tuning approaches. Given the impracticality of fine-tuning all parameters, we looked into PEFT.

**PEFT(Parameter-Efficient Fine-Tuning)** enables fine-tune a small number of (extra) model parameters - significantly decreasing computational and storage costs - while yielding performance comparable to a fully fine-tuned model. Within the PEFT framework, we opted for

**Prompt Tuning** —a method that strategically guides language model behavior by incorporating task-specific and prompts. We systematically varied prompts to observe the model's performance. This way, you can use one pretrained model whose weights are frozen, and train and update a smaller set of prompt parameters for each downstream task instead of fully finetuning a separate model.

The strategic adoption of flan-t5-large with PEFT and Prompt Tuning yielded promising results. Despite the challenges posed by the increased model size, the model showcased improved capabilities in forming equations, particularly for more complex math word problems. The enhancement was notable, demonstrating a relative improvement in equation generation compared to previous iterations.

Following successful model training, we prioritized user accessibility by integrating an interactive GUI using **Gradio**. This user-friendly platform simplifies input of math word problems and retrieval of model-generated solutions, ensuring a seamless experience for users, regardless of programming expertise.

#### Computing environment

GPU: Nvidia GPU

CUDA: version 11.8

#### Memory:

12 GB GPU memory

#### Software and Frameworks:

Pytorch

JupyterLabs

## 2.3 Model architecture

Transfer learning is a technique where a model trained on one task is adapted for another related task. Instead of training a model from scratch for a specific task, transfer learning leverages knowledge gained from solving one problem to solve a different but related problem. The model can then be trained on smaller, labeled datasets to achieve different tasks.

T5 employs an encoder-decoder architecture, making it well-suited for a variety of tasks, especially those involving sequence-to-sequence problems with input and output sequences that are of varying lengths. This is effective in handling diverse datasets and tasks by leveraging a pre-trained model on a range of unsupervised and supervised tasks. Each task is converted to a text-to-text format and the model is trained using teacher forcing, which means the training requires an input and a corresponding output sequence. So, the model is exposed to input-output pairs, refining its ability to generate accurate predictions.

The encoder decoder model architecture consists of two key components: the encoder and the decoder. The encoder processes the input sequence and compresses the information into a fixed-size context vector. This vector encapsulates all the information in the input to help the decoder make more accurate predictions. The decoder uses this context vector to generate the output sequence. Attention mechanisms are used to enhance the ability to capture long-range dependencies and improve the overall performance of the model, across diverse applications.

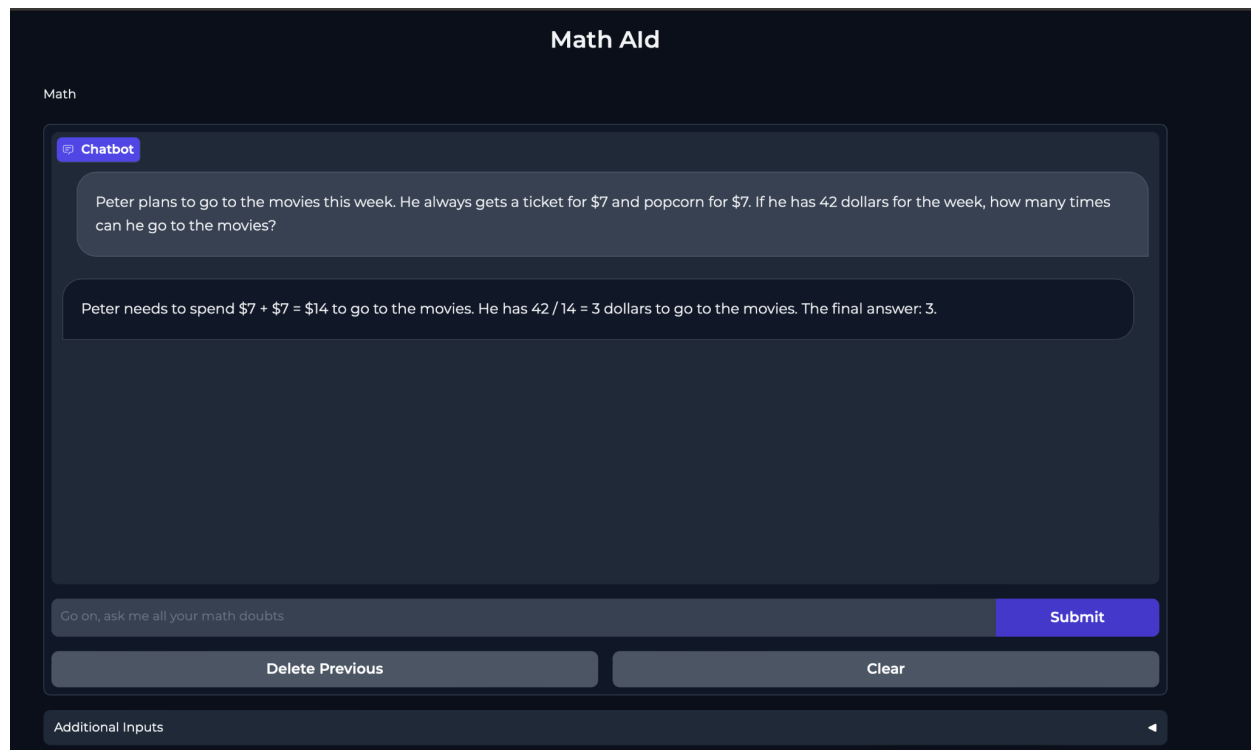
Beyond the original T5 model, several variants have been developed to address specific needs. For instance, UL2 and Flan-T5 are noteworthy adaptations. Flan, a pre-training method based on prompting, serves as the foundation for Flan-T5 models. These variants, such as google/flan-t5-small, google/flan-t5-base, google/flan-t5-large, google/flan-t5-xl, and

google/flan-t5-xxl, are trained on the Flan collection of datasets. Each variant is tailored to different scales of tasks, providing a spectrum of options based on the complexity and size of the target applications.

### 3. Experiment Results

Various prompts were utilized during experimentation to observe model outcomes. Examples include:

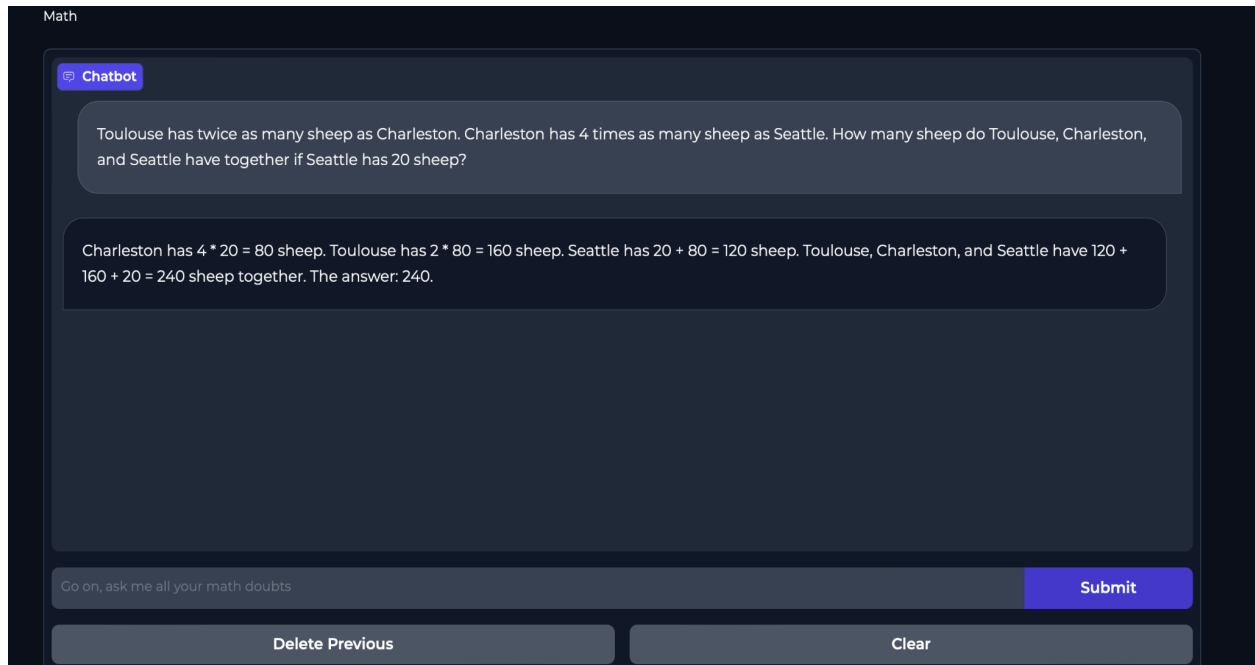
1. Let's first prepare relevant information and make a plan. Execute the plan, ensuring accurate numerical calculation and logical consistency. Present the answer step-by-step.
2. Answer the following question by reasoning step-by-step.
3. Start by analyzing the problem and assigning relevant variables and constants. Construct equations for the question using these elements. Pay special attention to intermediate variables and determine their suitability for reuse in subsequent equations. Solve the problem step by step and show the answer.



The screenshot shows a web interface for 'Math Aid'. At the top, the title 'Math Aid' is centered. Below it, the word 'Math' is on the left. A chatbot icon with the label 'Chatbot' is on the left. The main area contains a text input field with the question: 'Peter plans to go to the movies this week. He always gets a ticket for \$7 and popcorn for \$7. If he has 42 dollars for the week, how many times can he go to the movies?'. Below the input field is a text area with the solution: 'Peter needs to spend  $\$7 + \$7 = \$14$  to go to the movies. He has  $42 / 14 = 3$  dollars to go to the movies. The final answer: 3.' At the bottom, there is a 'Submit' button, a 'Delete Previous' button, and a 'Clear' button. Below these buttons is an 'Additional Inputs' field with a right arrow icon.

Sample output 1

In this instance, the generated equations are correct, but the explanation falls short



Sample output 2

Here, the model accurately analyzes the problem but falters in executing the correct mathematical computations.

## 4. Discussion

As specified earlier, the fine-tuned Language Model (LLM) exhibits promise in decoding steps for arriving at a final answer. Proficiency in reasoning is evident through the generation of intermediate steps. However, a notable limitation lies in the model's effectiveness in executing mathematical computations, resulting in inaccuracies within the final answers.

One potential explanation for this is in the emphasis of the training process. During the training phase, the prompts we utilized were designed to enhance the model's reasoning ability. This may have resulted in the observed discrepancy between the generated steps and the accurate computation of the final answer.

In contemplating future improvements, one option we aim to explore further is chain of thought code prompts as opposed to natural language prompts. This potential shift aims to investigate whether structuring prompts in a more code-oriented format could enhance the model's understanding of mathematical intricacies and potentially improve computational accuracy. There is relevant research supporting the same.

Another avenue for future exploration involves modifying the training strategy to strike a balance between enhancing reasoning abilities and refining mathematical computations.

In summary, while the current results showcase the model's prowess in reasoning, addressing the observed computational inaccuracies requires a nuanced approach. By refining prompts,



balancing training objectives, and exploring additional dimensions, the model can potentially evolve into a more impactful tool for enhancing mathematical reasoning abilities.

## 5. Conclusion

The project sought to enhance mathematical problem-solving by fine-tuning large language models using the GSM8K dataset. Leveraging pre-trained models and prompt engineering, the approach aimed to improve the model's reasoning ability and create a symbiotic relationship between language understanding and mathematical reasoning. The fine-tuned model displayed promising proficiency in decoding reasoning steps, achieving a RougeLsum score of 0.55. While demonstrating coherent intermediate solutions, the model showed limitations in precise mathematical computations, indicating room for improvement.

Carefully chosen parameters, including learning rate, batch sizes, and optimizer, contributed to computational efficiency. Various prompts were explored during training, providing comprehensive insights into the model's performance. Looking ahead, potential improvements involve experimenting with chain of thought code prompts and balancing training objectives to refine the model's computational accuracy. The project's holistic approach aims to create a more impactful tool for enhancing students' mathematical reasoning abilities.

## 6. References

1. [Understanding Encoder-Decoder Sequence to Sequence Model | by Simeon Kostadinov | Towards Data Science](#)
2. [T5](#)
3. [gsm8k · Datasets at Hugging Face](#)
4. [GitHub - openai/grade-school-math](#)
5. [\[2201.11903\] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#)
6. [\[2303.05398\] MathPrompter: Mathematical Reasoning using Large Language Models](#)
7. [CAME: Confidence-guided Adaptive Memory Efficient Optimization](#)

## 7. Team

1. Apeksha Suhas Joshi (002619010)
2. Aishwarya Suyamindra (002774595)
3. Akshay Gunjur Surya Prakash (002765803)