Heterogeneous Fact-Checking Agents for Mathematical Reasoning: A Consensus-Based Framework for Solution Assessment

Abdessalam Ed-dib
NYU Tandon School of Engineering
6 MetroTech Center, Brooklyn, NY 11201
ae2842@nyu.edu

Abstract

Large Language Models (LLMs) are currently at the forefront of natural language processing tasks, but they still tend to produce plausible yet incorrect information, known as "hallucinations." In specialized fields like mathematics, fact-checking these hallucinations requires rigorous frameworks to evaluate not only answer correctness but also logical coherence in solution steps. Current single-model approaches often fail to ensure consistency across diverse mathematical domains and reasoning styles. This paper proposes a multi-agent verification framework that employs a panel of specialized models, each trained on distinct mathematical datasets. A weighted consensus mechanism is used to aggregate the outputs of these models, enhancing the reliability and robustness of the fact-checking process. Experiments conducted on a dataset of 10,000 mathematical problems demonstrate a verification accuracy of 81.13%, outperforming single-model baselines by 6%. Ablation studies further highlight the importance of training dataset diversity, particularly in improving performance on edge cases and novel problem types. These findings underscore the potential of distributed verification systems to advance AI-driven mathematical reasoning, with applications in automated tutoring, proof validation, and the development of robust AI reasoning frameworks. The code is available on GitHub, and the model weights and dataset shards are publicly accessible on Hugging Face.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across various domains, from natural language understanding (Brown et al., 2020; Chowdhery et al., 2022) to complex problem-solving tasks. Indeed, recent models like GPT-4 exhibits human-level performance on various professional and academic benchmarks, including tasks requiring mathematical reasoning (OpenAI et al., 2024). However, their tendency to generate plausible but incorrect information—commonly known as hallucinations (Huang et al., 2023)—poses significant challenges, particularly in domains requiring precise reasoning and factual accuracy. This limitation becomes particularly problematic in mathematical contexts, where even minor errors in reasoning can lead to fundamentally incorrect conclusions.

The challenge of verification in mathematical reasoning differs fundamentally from general fact-checking. Traditional fact-checking approachesare primarily designed to evaluate natural language assertions. These methods typically rely on aligning claims with pre-existing knowledge bases and verifying sources (Vykopal et al., 2024). Moreover, fact-checking is often framed as an entailment task, where the goal is to determine whether a given claim logically follows from a supporting text (Poliak, 2020). However, this approach is not directly applicable to mathematical reasoning due to several factors: the structured nature of mathematical reasoning, the interdependence of solution steps, and the necessity of verifying both answer correctness and logical validity of its explanation (Ahn et al., 2024).

Existing approaches to mathematical verification can be broadly categorized in three main research streams:

Single-Model Verification Systems. These approaches leverage large-scale language models fine-tuned on mathematical datasets (Cobbe et al., 2021; Liang et al., 2024; Liu et al., 2023). While they have shown promise, they face several significant challenges. Specifically, they tend to perform well only within the domains they were pre-trained on and struggle to generalize to other areas. As a result, their performance degrades when encountering problems outside the scope of their training data (Ahn et al., 2024). Additionally, these models

often prioritize answer verification over the validity of explanations. They may produce correct answers but provide flawed reasoning steps (Huang et al., 2024). Finally, they tend to propagate systematic biases, leading to consistent misclassification of certain types of errors (Stephan et al., 2024).

Formal Verification Methods. These techniques leverage formal mathematical languages (de Moura et al., 2018; Paulson, 1994; Harrison, 2009) to enable automated proof verification, significantly impacting mathematics and computer science. They aim to train general pupose LLMs as Lean4 experts capable of writing formal proofs. However, challenges remain. First, even with guidance from natural language proofs, systems like TheoremLlama (Wang et al., 2024) struggle with solving complex IMO-level problems, highlighting LLMs' limited understanding of intricate human proofs. Second, the computational complexity of the Lean4 kernel prevents the application of reinforcement learning (RL) and iterative feedback mechanisms for refining proofs. Lastly, formal systems may unintentionally "correct" errors in natural language inputs, leading to flawed proofs being validated and propagated.

Hybrid Approaches. These approaches attempt to combine neural and symbolic approaches (Pan et al., 2023; Gaur and Saunshi, 2023), but they face significant limitations. These include the limited expressiveness of symbolic solvers and the challenges of converting natural language problems into symbolic representations, particularly when dealing with complex grammatical structures.

In this work, we introduce a multi-agent verification framework designed to overcome existing limitations through a panel-based approach. Our framework offers three key contributions:

- **Distributed Verification:** We implement N independent verification agents, each trained on distinct mathematical datasets, to provide diverse perspectives on solution evaluation. This approach mitigates systematic biases present in the training data.
- **Cross-Domain Robustness:** Through diverse training data and specialized agents, our system maintains consistent performance across various mathematical problems.
- **Weighted Consensus Mechanism:** Rather than simple aggregation, we use a weighted consensus mechanism that incorporates the performance of individual classifiers, ensuring more reliable and accurate verification outcomes.

2 Problem Overview

2.1 Problem Definition

We formalize the mathematical solution verification problem as follows: Let M be a mathematical question-answer-explanation triplet:

$$M = (q, a, e) \in Q \times A \times E \tag{1}$$

where:

- $q \in Q$ represents a mathematical question,
- $a \in A$ represents a proposed answer,
- $e \in E$ represents the provided explanation/solution steps.

The objective is to develop a verification function $V : M \to \{0, 1\}$ that maps each triplet to a binary correctness assessment, where 1 indicates a correct solution and 0 indicates an incorrect solution.

2.2 Input Structure

In Figure 1, we illustrate an instance of our dataset, which consists of:

- **Mathematical Question (q):** A well-formed mathematical problem expressed in natural language, potentially including mathematical notation in LaTeX format.
- **Proposed Answer (a):** A numerical, or structured response to the question.
- **Solution Explanation (e):** A detailed step-by-step explanation of the problem-solving process, which may include:
  - Intermediate calculations,
  - Mathematical reasoning steps,
  - Formula applications,
  - Computational procedures.

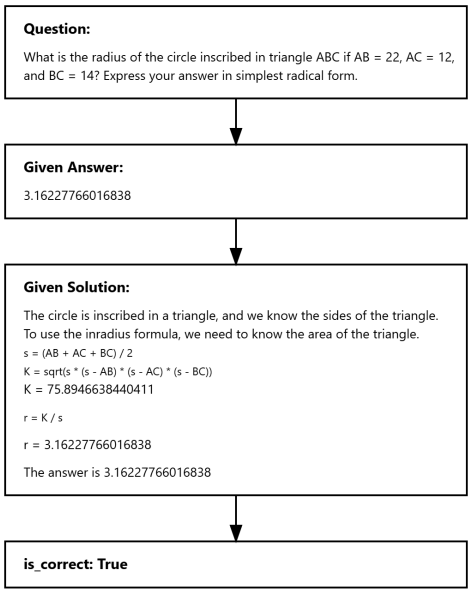


Figure 1: An example from our dataset illustrating a mathematical solution verification instance, comprising four key components: (i) the original question in natural language with mathematical notation; (ii) the proposed numerical answer; (iii) the detailed solution steps, including mathematical reasoning and computational procedures; and (iv) the ground truth correctness label.

2.3 Dataset Description

In our work, we use the Math Question Answer Verification Competition dataset (Kumar, 2024). It consists of mathematical question-answer pairs, which are designed to assess the correctness of the provided answers and include corresponding explanations. Each data instance is structured with four columns: **"question," "answer," "solution," and "is_correct,"** as outlined in Section 2.2. The dataset is divided into a training subset containing 1 million samples and a test subset comprising $10,000$ samples. Hereafter, we present an exploratory data examination of both subsets.

2.3.1 Training Subset

Figure 2 illustrates the distribution of labels in the training set. We observe that correct labels make up 60% of the data, while incorrect labels account for 40%. This imbalance is noteworthy, especially considering the large size of the dataset. This may introduce bias during training, potentially affecting the model's performance.

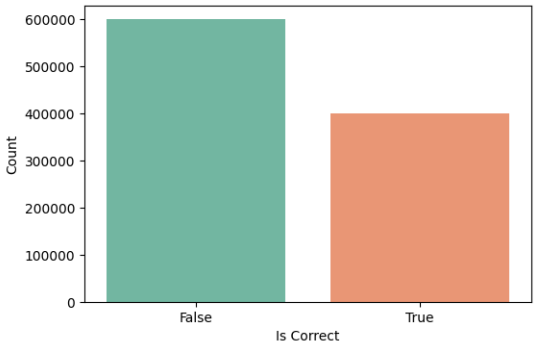


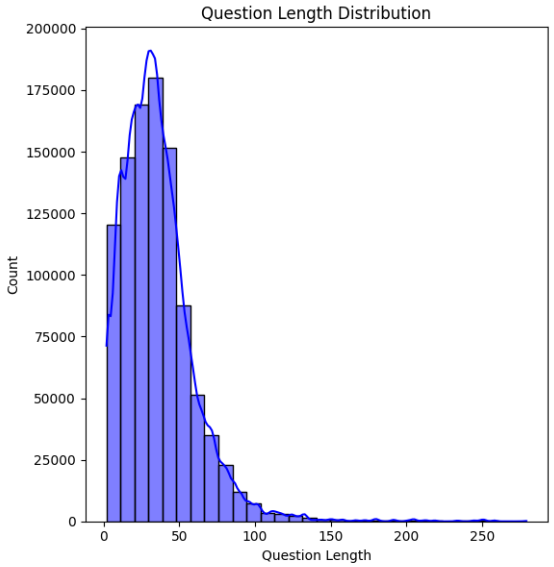
Figure 2: Distribution of labels in the training set, showing 60% correct labels and 40% incorrect labels.

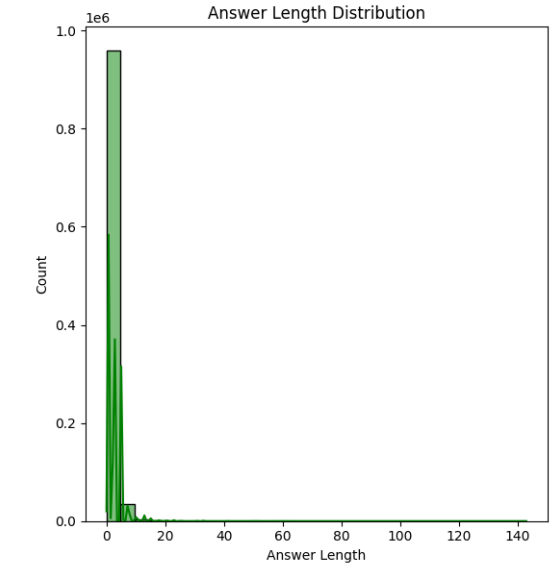
Figure 3 illustrates the length distribution of the questions, answers, and explanations in the training set. The question length distribution follows a right-skewed pattern, peaking around 25-50 tokens. Most questions fall within the 0-100 token range, with a tail extending beyond 200 tokens, indicating occasional verbose queries. The answer length distribution displays a sharp, right-skewed pattern, with a prominent peak near 0-25 tokens and minimal presence beyond 50 tokens, suggesting that responses are generally concise and focused. In contrast, the explanation length distribution is broader and right-skewed, peaking around 150-250 tokens, with a tail extending beyond 1000 tokens. This indicates that explanations tend to be much more detailed compared to both questions and answers.

2.3.2 Test Subset

Similarly, the test subset also shows an imbalance, with approximately 38% of the samples labeled as correct and the remaining samples labeled as incorrect[1]. Figure 4 illustrates the length distribution of the questions, answers, and explanations. The question length distribution (top, blue) displays a right-skewed pattern with a peak around 30-40 units and gradually decreases until approximately 175 units, demonstrating a moderate spread of question lengths. The answer length distribution (middle, green) shows an extremely concentrated distribution with a sharp peak at approximately 5 units and rapidly diminishing frequencies beyond 10 units, suggesting a dataset dominated by very brief answers. The explanation length distribution (bottom, red) exhibits a right-skewed pattern peaking around 50-75 units with a tail extending to 700 units, though more concentrated.

[1]We submit a file with all predictions labeled as True to determine the number of True samples.





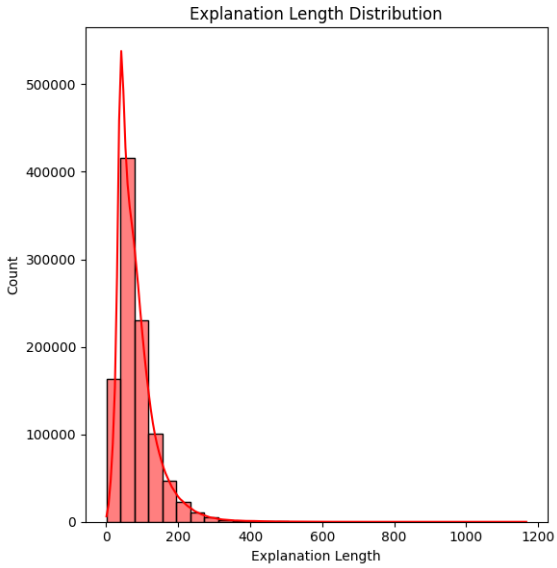
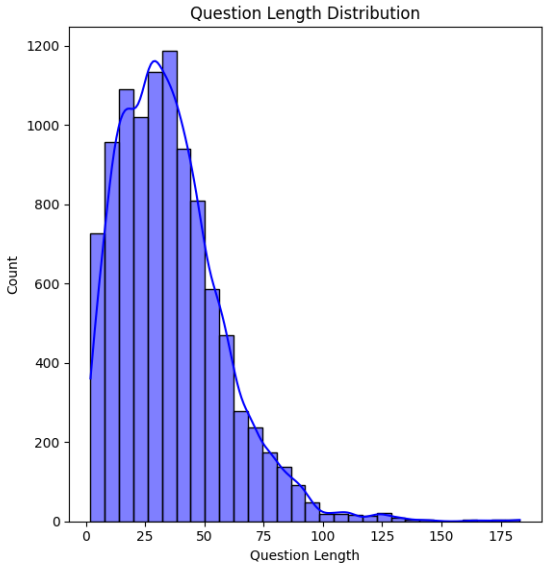
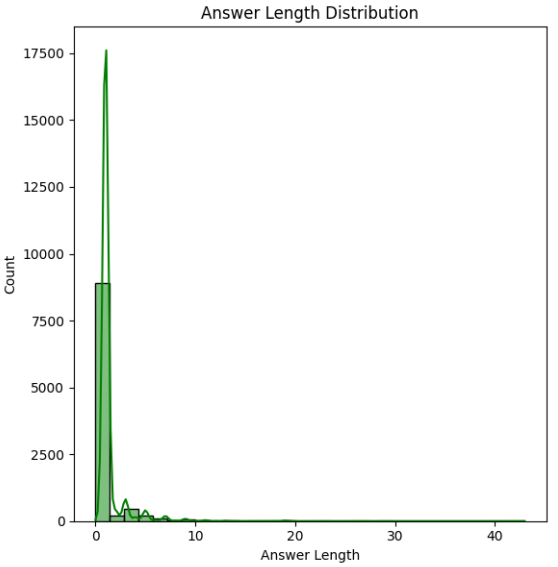


Figure 3: Length distribution analysis of the training set, comprising three histograms showing the frequency distributions of questions (left, blue), answers (middle, green), and explanations (right, red).





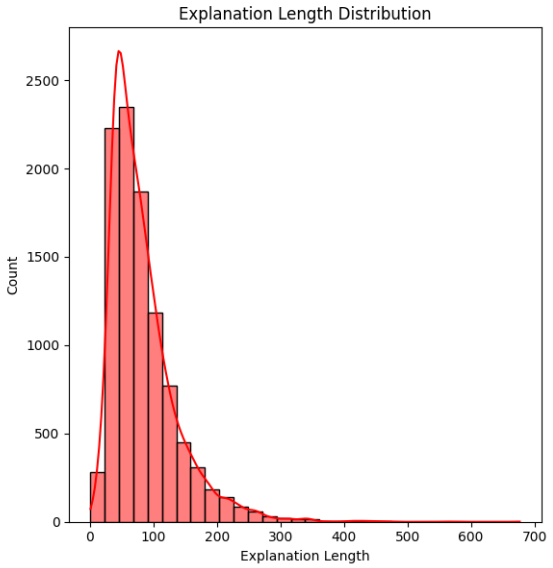


Figure 4: Length distribution analysis of the test subset, comprising three histograms showing the frequency distributions of questions (left, blue), answers (middle, green), and explanations (right, red).

2.4 Problem Challenges

This problem presents several challenges, primarily due to the large scale of the model [2], the size and imbalance of the dataset, and the need for the model to strictly adhere to instructions for accurate answer verification. The key challenges arising from this setup are outlined below:

2.4.1 Computational Complexity

The computational complexity of training and evaluating a model with 8 billion parameters on a dataset of 1 million samples is a significant concern. Handling a large number of parameters requires considerable computational resources, both in terms of hardware (e.g., high-performance GPUs) and optimization strategies. This scale of the model and dataset necessitates parallelization, distributed training, and efficient memory management. Additionally, the size of the dataset (1 million samples) combined with the large model can lead to slow training times, which can be exacerbated by the model's need for frequent fine-tuning and hyperparameter optimization to balance performance across multiple tasks.

2.4.2 Imbalanced Data

The dataset imbalance—where 40% of the samples are correct and 60% are incorrect—introduces several issues during model training and evaluation, especially when taking into account the dataset size (1 million samples). Given that the majority of the samples are incorrect, the model may develop a bias towards classifying answers and explanations as incorrect, potentially leading to false negatives. This bias can result in poor performance, particularly when the correct answers and explanations are more subtle or harder to distinguish from the incorrect ones. Additionally, imbalanced data can affect the model's ability to learn to detect rare errors or nuanced mistakes. For example, incorrect answers may vary in type and severity, and the model may be exposed to a higher volume of "obvious" errors, while it may struggle with identifying less obvious, but more subtle, flaws in explanations.

To mitigate the effects of imbalance, techniques like oversampling the minority class (correct answers and explanations), undersampling the majority class (incorrect answers and explanations), or using class-weighting during training can be employed. However, these approaches need to be balanced carefully to avoid overfitting to either class, which could harm the model's ability to generalize across the full range of mathematical problems.

2.4.3 Model Not Following Instructions

Ensuring that the model correctly interprets and follows instructions across multiple tasks—such as evaluating both the answer's correctness and the explanation's relevance—presents a considerable challenge. The model needs to not only classify the answer as correct or incorrect but also verify the mathematical reasoning behind it and determine whether the explanation provided is logically sound to improve its reasoning capabilities and generalize to unseen problems. However, there is the challenge of "following instructions" in the sense of adhering to the precise logic required for the verification process. If the model is trained on general language tasks[3], it may struggle to apply the necessary mathematical reasoning principles to validate both the answer and the explanation in the way a human expert would.

3 Related Work

Fine-tuning LLMs presents significant computational challenges due to their massive parameter counts, often reaching billions of parameters. Indeed, traditional fine-tuning involves updating all parameters of a pre-trained model to optimize performance on a target task. Given a pre-trained model with parameters $\theta = (W_1, \cdots, W_l)$[4], the fine-tuning process minimizes the task-specific loss $\mathcal{L}$:

$$\theta' = \arg\min_{\theta} L(D_{target}; \theta) \tag{2}$$

where D_{target} represents the target task dataset. While effective, this approach requires substantial computational resources and memory, scaling linearly with model size. To address the computational inefficiency associated with full fine-tuning, parameter-efficient fine-tuning (PEFT) techniques (Han et al., 2024) have emerged, focusing on training the model with a reduced number of parameters. PEFT methods can be broadly categorized into the following approaches:

- **Weight Updates:** Modifying a subset of the model's original parameters (Ben Zaken et al., 2022).

[2] LLaMA 3.1 8B

[3] This applies to the standard LLaMA 3.1 8B model.

[4] W_i are the weight matrices of the LLM.

- **Adapter Techniques:** Introducing special adapter layers "bottleneck networks" within the model (Poth et al., 2023), which are trained while keeping the core model parameters frozen.
- **Low-Rank Adaptation (LoRA):** Using low-rank adaptation matrices to update the model's weight matrices (Hu et al., 2021).
- **Prompt Tuning:** Techniques that involve modifying the input prompts to guide the model's responses without changing the model's internal parameters (Petrov et al., 2024).

In this section, we examine key PEFT methods that address these challenges, with a primary focus on low-rank adaptation techniques and weight updates. Adapter techniques typically require updating a large number of parameters to achieve improved performance, whereas prompt tuning methods aim to elicit the model's existing capabilities rather than enabling it to learn new skills (Petrov et al., 2024).

3.1 Low Rank Adaptation

LoRA (Hu et al., 2021) introduces a parameter-efficient approach by representing weight updates through low-rank decomposition. For each weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA adds an adaptation matrix ΔW that mimics the gradient update:

$$\Delta W = BA \tag{3}$$

where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$, with rank $r \ll \min(d, k)$. The updated weight matrix becomes:

$$W' = W + \frac{\alpha}{r}\Delta W$$

where α is a scaling factor. This approach reduces the trainable parameters from $d \times k$ to $r(d + k)$, achieving significant memory efficiency while maintaining performance comparable to full fine-tuning.

3.2 Gradient Low-Rank Projection (GaLore)

GaLore (Zhao et al., 2024) presents an approach to memory-efficient training by operating in gradient space rather than parameter space. It consists of projecting gradients onto a low-dimensional subspace determined through efficient singular value decomposition (SVD) of gradient samples. This projection ensures that parameter updates occur within a memory-efficient subspace while preserving the most important directions of optimization. GaLore achieves comparable performance to full fine-tuning while reducing memory requirements by orders of magnitude.

3.3 BitFit

BitFit (Zaken et al., 2022) proposes an extreme form of PEFT by updating only the bias terms in a pre-trained model. The approach is formalized as follows:

$$\theta'_{bias} = \arg\min_{\theta_{bias}} L(D_{target}; \theta_{frozen}, \theta_{bias})$$

where θ_{bias} represents the bias parameters and θ_{frozen} denotes the frozen pre-trained weights. Despite its simplicity, BitFit demonstrates competitive performance on various NLP tasks while requiring minimal memory overhead, as it updates only a tiny fraction of the model's parameters.

Each of these methods represents a distinct approach to the challenge of efficient model adaptation. LoRA leverages low-rank decomposition in parameter space, GaLore operates through gradient projection, and BitFit focuses exclusively on bias terms.

4 Methodology

Our primary objective is to develop a mathematical verification system capable of generalizing across a wide range of mathematical problems. However, we face a significant challenge due to resource constraints, as we are unable to train our model on the entire dataset in a single pass. Additionally, sequential fine-tuning on subsets of the dataset is not ideal, as it may result in catastrophic forgetting, where the model loses knowledge of previously encountered samples, thereby limiting its reasoning and generalization abilities. To address this, we propose an alternative approach: parallel fine-tuning on different subsets of the data. This method not only mitigates the risk of catastrophic forgetting but also helps reduce bias stemming from data imbalance, as the number of samples in each subset is much smaller than the total dataset size of one million samples. Lastly, to enhance the model's reasoning capabilities, we employ a chain-of-thought prompting strategy to explicitly guide the model through the key steps required for verifying answers.

Figure 5 illustrates our proposed methodology, termed **"HFAM" (Heterogeneous Fact-Checking**

Agents for Mathematical Reasoning). Our framework is structured into three primary components: **(1) input prompt formatting**, which standardizes and prepares the mathematical queries for evaluation; **(2) verification agents**, which independently evaluate the mathematical claims; and **(3) a weighted consensus voting mechanism**, which aggregates the outputs of the verification agents to derive a final, reliable conclusion.

4.1 Prompt Engineering with Chain of Thought

Chain of Thought (CoT) prompting has emerged as a powerful technique for improving LLMs' reasoning capabilities (Wei et al., 2023). It enables models to break down complex tasks into intermediate steps for more reliable outputs. In the context of mathematical verification, CoT is particularly valuable as it mimics the systematic evaluation process that human mathematicians employ when assessing mathematical solutions.

Figure 6 outlines our verification system's prompt. It employs a carefully structured prompt that guides the model through a systematic evaluation process:

```
You are a highly skilled mathematician tasked with evaluating whether an answer to a
mathematical question is correct. To provide an accurate evaluation, follow these steps:

1. Read the question carefully to understand what is being asked.
2. Review the answer provided and determine if it logically follows from the question.
3. Examine the explanation given to know whether it provides sufficient reasoning to
support the answer, and whether there are any logical flaws or assumptions that need
to be addressed.
4. Based on your assessment of the question, answer, and explanation, decide if the
answer is correct or not.
5. Your Output should exactly be 'True' if the answer is correct, otherwise 'False'.

### Question:
{question}

### Answer:
{answer}

### Explanation:
{explanation}

### Output:
{output}
```

Figure 6: Prompt Template for Mathematical Answer Verification

The prompt structure is guided by three key design principles to ensure clarity, consistency, and effectiveness. **Role Definition** establishes a clear expert persona, framing the model as a **"highly skilled mathematician"** to provide an appropriate context for mathematical evaluation. **Sequential Decomposition** breaks the verification task into distinct, logically ordered steps that mirror natural mathematical reasoning processes. Finally, **Explicit Instruction** offers clear evaluation criteria and specifies the expected output format **"True or False"**, minimizing ambiguity in the model's responses and ensuring alignment with task requirements.

4.2 Verification Agents Architecture

Our verification framework employs an ensemble of 34 classifiers, designed to capture different aspects of mathematical reasoning verification. The system consists of two main groups of agents: **primary verification agents (n=20)** and **diversity-enhanced agents (n=14)**, each built upon the LLaMA-3 8B (Dubey et al., 2024) model using different training configurations.

4.2.1 Primary Verification Agents

Each primary agent is trained on an independent subset of the original dataset, which is carefully constructed to preserve the statistical properties of the original training set, particularly the proportions of correct and incorrect labels.

The main characteristics of the training subsets used to train the primary agents are summarized in Table 1.

Table 1: Training Data Characteristics per Agent

Characteristic	Value
Dataset Size	50,000 samples
Correct Labels	40%
Incorrect Labels	60%

The primary agents in our model are fine-tuned using LoRA. In LoRA, selecting the appropriate rank for the weight matrices is a crucial task, as it involves balancing expressivity with computational efficiency. Given the resource constraints in our setup, we begin by setting the rank "r" to 16. This choice ensures sufficient expressivity for complex reasoning tasks, compared to using lower ranks. The alpha scaling factor "α" is set to 32 ($\alpha = 2r$), which controls the influence of the low-rank weight matrices on the original parameters, promoting stable gradient flow and efficient convergence. Additionally, a dropout rate of 0.1 is employed to mitigate overfitting to the training data. To further stabilize training, we use RSLoRA (Rank Stabilizing LoRA) (Kalajdzievski, 2023),

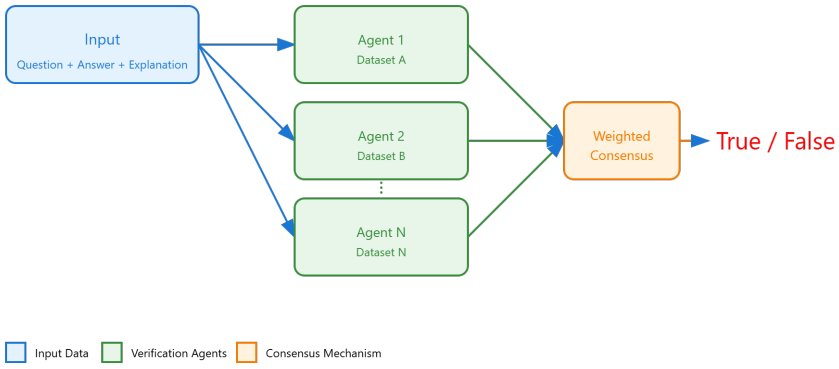


Figure 5: Proposed multi-agent architecture for mathematical answer verification. The system employs N heterogeneous verification agents, each trained on distinct datasets, to independently assess input consisting of a question, answer, and explanation. A weighted consensus mechanism aggregates individual agent verdicts to produce a final true/false verification decision.

which adjusts the scaling factor by dividing it by the square root of the rank, ensuring stability when larger ranks are used without sacrificing performance. The fine-tuning process focuses on optimizing the attention mechanisms and feed-forward networks, mainly `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`, to enhance the model's performance on mathematical reasoning tasks.

The main LoRA parameters used to train the primary agents are summarized in Table 2.

Table 2: Training LoRA Parameters for Primary Agents

Parameter	**Value**
Rank	16
Alpha	32
Target Modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
Dropout	0.1
RS LoRA	True
Bias	None

4.3 Diversity-enhanced Agents

To improve the expressiveness of our model panel, we add extra classifiers to capture patterns that the primary models may have missed. These new agents are created using either shard-diversity, LoRA parameterization, or a combination of both.

4.3.1 Shard-diversity Agents

Shard-diversity agents are created through the generation of multiple training data subsets, each initialized with different random seeds. These derivative training shards differ from the original dataset in three key aspects:

1. **Class Distribution:** For some agents, we maintain balanced representation between positive and negative samples to prevent class imbalance effects.

2. **Variable Dataset Sizes:** Multiple training shards are created with distinct sample counts (20,000, 30,000, 40,000, 51,000, and 100,000 samples) to assess the impact of training data volume.

3. **Calibration Bias:** One shard incorporate an intentional class ratio skew (e.g., 6,000 positive to 5,000 negative samples) to calibrate the model's overall prediction confidence.

4.4 LoRA-diversity Agents

LoRA-diversity agents are generated by creating multiple LoRA models, each initialized with distinct parameters [5]. These models differ from the primary agents in four key aspects:

1. **Dropout Ratio:** For certain agents, we apply a lower dropout ratio to enable the model to capture more patterns from the training data.

[5] It is worth mentioning that Model 31 has been trained using GaLore.

Dropout ratios of 0.05, 0.01, and 0 are tested to observe their effect on model performance.

2. **Rank:** The ranks of the low-rank adaptation layers are adjusted to both higher and lower values. This variation allows for greater expressivity in some models, while others use lower ranks for faster convergence, depending on the dataset size. (Example: r = 8, r = 64)

3. **Alpha Rank Ratio:** The alpha rank ratio is modified to enable faster adaptation of the model's weights, allowing the model to capture patterns more quickly during training. Values tested include alpha rank ratios of 1, 2, and 3.

4. **Using RS LoRA**: The RS LoRA approach is employed, where rank stabilization is omitted in certain cases to capture different patterns from other models.

These parameter configurations, as well as the dataset shard details, are available in the publicly hosted configuration files for the fine-tuned adapters on Hugging Face, allowing for easy replication and further experimentation.

4.5 Weighted Consensus Voting Mechanism

In Figure 7, we observe that the agents exhibit a bias towards predicting "False" labels. This bias persists even after mitigating dataset imbalance by splitting the data into smaller shards.

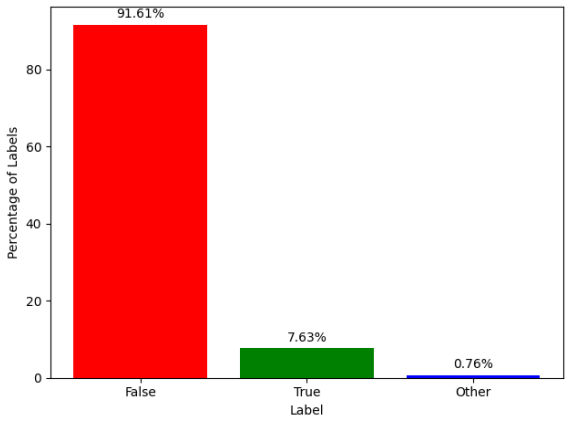


Figure 7: Distribution of models predictions across the test dataset, showing the percentage of "True", "False", and "Other" labels.

Moreover, Figure 8 shows that some agents exhibit a stronger bias than others, depending on the data seeds used for training. This suggests that not all data samples contribute equally to model performance. A more effective approach would involve evaluating the relevance of these samples and retaining only the most useful ones. One possible method for this is Shapley value-based techniques, which assign a score to each data sample based on its contribution to the model's performance during training (Choe et al., 2024).

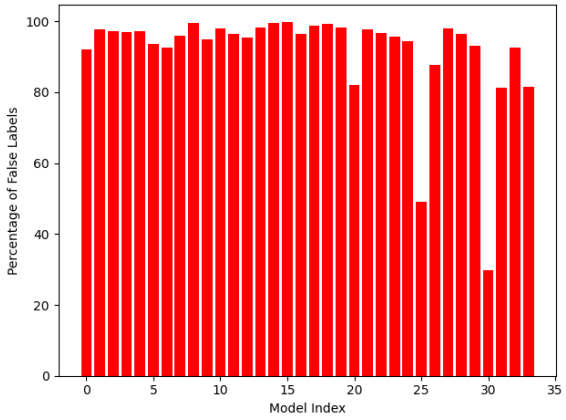


Figure 8: Distribution of "False" label percentages across 34 agents' predictions.

Given the time constraints, one direct approach to mitigate this bias is to employ a weighted voting mechanism, which assigns greater importance to "True" predictions than to "False" predictions. Specifically, we evaluate three main variants of this mechanism: the first is an extreme version, where a "True" output from any single agent results in an overall "True" output; the second variant outputs "True" if two agents predict "True"; and the third variant requires that at least three agents output "True" for the final decision to be "True".

5 Experimental Results

In this section, we present our experimental design and results to assess the performance and efficiency of our method against the baseline. We include the numerical results and discuss these findings, highlighting the advantages and limitations of our approach.

5.1 Experimental Setting

We implemented all algorithms using PyTorch, based on the publicly available HuggingFace Transformers (Wolf et al., 2020) code-base. For optimization, we used the AdamW optimizer (Loshchilov and Hutter, 2019), which features parameters set to $\epsilon = 10^{-6}$, $\beta_1 = 0.9$, and

$\beta_2 = 0.999$. The batch size was set to 2, with 16 gradient accumulation steps. Additionally, we applied a warm-up ratio of 0.1, weight decay of 0.01, and a maximum gradient norm of 0.3. A cosine learning rate scheduler was used, and learning rates were varied between $1e^{-5}$ and $5e^{-5}$. Finally, we set the number of training epochs to 1, and we use a quantized version of of the base model (4 bits). All experiments were conducted on 8 `NVIDIA A100-SXM4 GPUS`.

5.2 Evaluation Metric

Both the training set and the test subset exhibit a significant imbalance, with a much higher proportion of False labels compared to True labels. Given the large size of the datasets, this imbalance becomes even more pronounced. Consequently, using accuracy as a performance metric is not an appropriate choice in this context. This is because accuracy tends to favor the majority class, making it less reliable for evaluating model performance on imbalanced datasets (Menon et al., 2013; Holzmann and Klar, 2024).

For example, if we use a deterministic model that always outputs False, the test accuracy would still be approximately 68%. This result is misleading, as it does not reflect the model's ability to correctly classify both True and False labels. Instead, it merely highlights the imbalance in the dataset, where the majority class "False" dominates the distribution. Such a high accuracy is unrealistic and does not provide meaningful insights into the model's actual performance.

Indeed, in the context of fact-checking systems using LLMs, two primary metrics commonly used to evaluate model performance on inherently imbalanced fact-checking benchmarks are F1-score and balanced accuracy. These metrics are preferred because they account for the class imbalance present in such datasets.

F1-score. It is the harmonic mean of precision and recall, offering a balance between the two. It is especially useful in imbalanced settings because it provides a more informative evaluation than accuracy by considering both false positives and false negatives, making it a reliable metric for assessing performance across both the minority and majority classes.

Balanced accuracy. It adjusts the traditional accuracy metric by calculating the average of recall for each class, ensuring that performance on the minority class is given equal weight. This metric mitigates the bias that can occur when the majority class dominates the dataset, providing a more balanced and fair evaluation of the model's performance across all classes.

5.3 Numerical Results

5.3.1 Training Dynamics

In this section, we begin by discussing the training losses of our models. Since all models exhibit similar training curves, we focus on analyzing the training curve of the first model. The training curves of the remaining models are provided in GitHub.

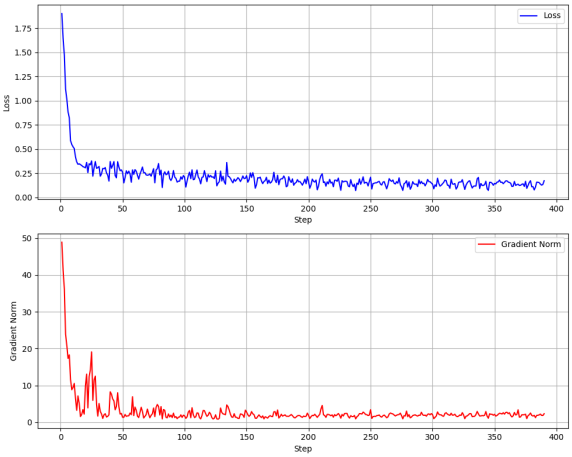


Figure 9: Loss and Gradient Norm vs. Training Steps: Plots showing the evolution of the loss (top) and gradient norm (bottom) over the course of training, with respect to the number of steps.

Figure 9 illustrates the training dynamics of the first primary model, which exhibit steady convergence of both the loss and gradient norm, without significant fluctuations, suggesting a stable and consistent optimization process. The loss decreases gradually from an initial value of approximately 2 to around 0.2, indicating effective learning and a substantial reduction in the model's error over the course of training. Similarly, the gradient norm shows a smooth decline from values exceeding 50 to approximately 1, suggesting that the magnitude of the gradients is progressively diminishing as the model approaches a more optimal parameter configuration. The lack of high fluctuations in both metrics further supports the notion of stable convergence, indicative of efficient training with minimal overfitting or instability.

5.3.2 Assessment of Overfitting

Initially, given the size of the dataset used for fine-tuning the base model and the effective batch size of 256 ($2 \times 16 \times 8$), we hypothesized that overfitting would not occur, especially since we are only training for one epoch.

To validate this hypothesis, we trained an additional model with double the accumulation steps (32) to expedite the training process. We compared the training and validation loss curves to evaluate the presence of overfitting. A separate validation set of 2,000 samples, distinct from the training data, was used for this purpose. To reduce computational overhead, validation loss was calculated every 20 steps instead of after each step. The results are presented in Figure 10.

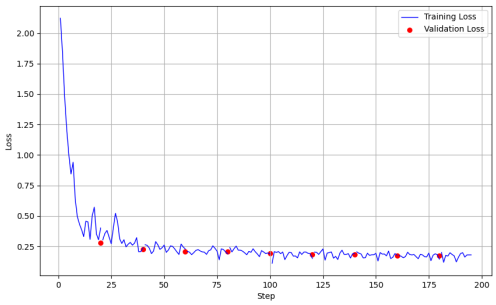


Figure 10: Training and validation loss across training steps: The training loss (blue line) is plotted continuously for all steps, while the validation loss (red points) is displayed only for the steps where validation was performed.

Figure 10 demonstrates the convergence of the validation loss without any upward trend toward the end of training confirms the absence of overfitting, thereby providing evidence in support of our hypothesis.

5.4 Validation Results

To comply with the competition guidelines, we will report the following metrics: accuracy, balanced accuracy, and F1-score. These metrics are calculated on a subset of 500 samples, randomly selected from the shuffled training set using a random seed of 42. The first 500 samples are then considered for evaluation. The results are presented in Table 3.

Table 3 presents a key observation: despite the relatively high accuracy of the weak models (approximately 66%), their F1-score is notably low (below 0.4). This discrepancy is due to data imbalance, where the model's accuracy is influenced by the dominant class "False", but its ability to correctly classify the minority class is poor. In the following section, we explore how our weighted consensus mechanism can address this issue and improve the performance of these weak models.

Table 3: Validation metrics for primary agents.

Model	Accuracy	Balanced Accuracy	F1 Score
Model 1	0.68	0.58	0.30
Model 2	0.64	0.52	0.07
Model 3	0.64	0.53	0.13
Model 4	0.64	0.52	0.10
Model 5	0.63	0.51	0.08
Model 6	0.67	0.56	0.22
Model 7	0.69	0.59	0.32
Model 8	0.64	0.52	0.07
Model 9	0.62	0.50	0.00
Model 10	0.67	0.56	0.22
Model 11	0.62	0.50	0.04
Model 12	0.65	0.53	0.12
Model 13	0.66	0.54	0.17
Model 14	0.64	0.52	0.07
Model 15	0.63	0.50	0.03
Model 16	0.62	0.50	0.01
Model 17	0.66	0.55	0.19
Model 18	0.62	0.50	0.04
Model 19	0.63	0.50	0.03
Model 20	0.65	0.53	0.13

Table 4 illustrates the performance of the weighted consensus-based models, which aggregate the predictions of the primary agents using a voting mechanism that prioritizes "True" predictions. The "True" threshold represents the minimum number of predicted true values required to output a "True" prediction. As shown, the performance of our model has significantly improved across all metrics, including the F1-score, which has increased from 0.3 (the maximum performance of the primary agents) to 0.55.

Table 4: Validation metrics for weighted-consensus based models.

"True" Threshold	Accuracy	Balanced Accuracy	F1 Score
1	0.74	0.68	0.55
2	0.70	0.61	0.38
3	0.70	0.61	0.38

Next, we will investigate the impact of the diversity agents, whose performance on the validation set is summarized in Table 5.

Table 5: Validation metrics for diversity agents.

Model	Accuracy	Balanced Accuracy	F1 Score
Model 1	0.71	0.63	0.46
Model 2	0.64	0.53	0.14
Model 3	0.64	0.52	0.11
Model 4	0.66	0.55	0.20
Model 5	0.66	0.55	0.20
Model 6	0.73	0.74	0.68
Model 7	0.70	0.61	0.38
Model 8	0.64	0.52	0.08
Model 9	0.64	0.53	0.12
Model 10	0.67	0.56	0.22
Model 11	0.75	0.79	0.74
Model 12	0.63	0.52	0.13
Model 13	0.65	0.54	0.18
Model 14	0.71	0.62	0.39

Table 5 presents the performance of the diversity agents on the selected validation set. Most models demonstrate poor performance in terms of F1-score, despite achieving high accuracy. However, it is important to note that further investigation revealed that the models showing high F1-scores likely benefited from cross-contamination between the validation set and their training data.

Finally, we evaluate the performance of the weighted consensus-based models and summarize the results in Table 6.

Table 6: Validation metrics for weighted-consensus based models including diversity agents.

"True" Threshold	Accuracy	Balanced Accuracy	F1 Score
1	0.73	0.77	0.72
2	0.80	0.77	0.71
3	0.77	0.71	0.62

Table 6 demonstrates that the weighted consensus-based models exhibit better performance on the validation set, with the threshold of 2 yielding the best performance, achieving approximately 80% accuracy. This threshold also provides a good balanced accuracy (77%) and F1-score (71%).

5.5 Test Results

By adopting our weighted consensus-based models, we vary the "True" threshold and explore different combinations of primary and diversity agents to construct our final submission file.

Table 7: Test metrics for weighted-consensus based model.

Split	Accuracy
Public	0.82
Private	0.81

Table 7 demonstrates that our model is both reliable and robust, as its accuracy remained consistent across different splits, indicating strong generalization capabilities. However, it is still essential to compute the F1-score to make a final assessment of the model's performance given the imbalance of the dataset.

6 Conclusion

In this work, we propose a novel approach for fact-checking mathematical answers that emphasizes improved robustness and generalization. Our method involves training 20 primary verification agents on distinct, independent subsets of the data. Additionally, we train 13 agents to enhance model expressiveness, utilizing differently sampled datasets that vary in label distribution or LoRA fine-tuning parameters. This approach ensures that our models capture diverse patterns in the training set, allowing them to generalize effectively to unseen data. Finally, we aggregate the results of these models using a weighted consensus mechanism, which mitigates bias inherent in the training data by prioritizing true predictions over false ones. Our experimental results demonstrate that the performance of our model remains consistent across both public and private test sets, showing a significant improvement over the baseline model.

Limitations

While our panel-based mathematical fact-checking system demonstrates promising results, several crit-

ical limitations must be acknowledged. Firstly, significant challenges arise in scalability and generalization when increasing the number of agents to capture broad areas of mathematical concepts, as we need to perform inference for each agent independently before aggregating the results. The interpretability of the system remains a notable concern, as the consensus mechanism between different fact checkers lacks transparency, making it challenging to understand and validate the reasoning behind verification decisions. Moreover, the system's performance is inherently limited by its training paradigm, which relies on vanilla LLaMA3.1 8B, affecting its ability to follow the necessary instructions passed to the prompt to verify the correctness of the answer. Finally, we still need to perform necessary verification for questions requiring very long explanations and assess whether our model can effectively follow through these explanations and multi-step reasoning processes. These limitations suggest important directions for future research, particularly in developing more transparent consensus mechanisms and improving the system's ability to handle diverse mathematical reasoning patterns.

Other Approaches

It is worth mentioning that we attempted to sequentially fine-tune a single model with different prompt variations and data seeds. However, these models struggled to generalize well to unseen data. One primary reason for this is the inability of the vanilla LLaMA model to accurately follow prompt instructions. Using the instruct version of LLaMA as the base model could significantly improve the classifier's performance. Additionally, not all data seeds are equally useful for model training. Therefore, implementing a filtering step before training, using techniques such as Shapley value analysis and influence functions, could enhance the model's effectiveness.

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. 2024. What is your data worth to gpt? llm-scale data valuation with influence functions.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

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.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2018. The Lean Theorem Prover (system description).

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev,

Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg

Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. 2024. The llama 3 herd of models.

Vedant Gaur and Nikunj Saunshi. 2023. Reasoning in large language models through symbolic math word problems. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5889–5903, Toronto, Canada. Association for Computational Linguistics.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey.

John Harrison. 2009. Hol light: An overview. In *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics*, TPHOLs '09, page 60–66, Berlin, Heidelberg. Springer-Verlag.

Hajo Holzmann and Bernhard Klar. 2024. Robust performance metrics for imbalanced classification problems.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. 2024. Large language models cannot self-correct reasoning yet.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora.

Amardeep Kumar. 2024. Nyu dl fall 24 competition. `https://kaggle.com/competitions/nyu-dl-fall-24-competition`. Kaggle.

Zhenwen Liang, Dian Yu, Xiaoman Pan, Wenlin Yao, Qingkai Zeng, Xiangliang Zhang, and Dong Yu. 2024. MinT: Boosting generalization in mathematical reasoning via multi-view fine-tuning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11307–11318, Torino, Italia. ELRA and ICCL.

Yixin Liu, Avi Singh, C. Daniel Freeman, John D. Co-Reyes, and Peter J. Liu. 2023. Improving large language model fine-tuning for solving math problems.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Aditya Menon, Harikrishna Narasimhan, Shivani Agarwal, and Sanjay Chawla. 2013. On the statistical consistency of algorithms for binary classification under class imbalance. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 603–611, Atlanta, Georgia, USA. PMLR.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris

Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. 2023. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning.

L.C. Paulson. 1994. *Isabelle: A Generic Theorem Prover*. Lecture Notes in Computer Science. Springer.

Aleksandar Petrov, Philip H. S. Torr, and Adel Bibi. 2024. When do prompting and prefix-tuning work? a theory of capabilities and limitations.

Adam Poliak. 2020. A survey on recognizing textual entailment as an nlp evaluation.

Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A unified library for parameter-efficient and modular transfer learning.

Andreas Stephan, Dawei Zhu, Matthias Aßenmacher, Xiaoyu Shen, and Benjamin Roth. 2024. From calculation to adjudication: Examining llm judges on mathematical reasoning tasks.

Ivan Vykopal, Matúš Pikuliak, Simon Ostermann, and Marián Šimko. 2024. Generative large language models in automated fact-checking: A survey.

Ruida Wang, Jipeng Zhang, Yizhen Jia, Rui Pan, Shizhe Diao, Renjie Pi, and Tong Zhang. 2024. Theoreml-lama: Transforming general-purpose llms into lean4 experts.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection.