# The Reason behind *Good* or *Bad*: Towards a Better Mathematical Verifier with Natural Language Feedback

**Anonymous ACL submission**

## Abstract

Mathematical verfier achieves success in mathematical reasoning tasks by validating the correctness of solutions. However, existing verifiers are trained with binary classification labels, which are not informative enough for the model to accurately assess the solutions. To mitigate the aforementioned insufficiency of binary labels, we introduce step-wise natural language feedbacks as rationale labels (i.e., the correctness of the current step and the explanations). In this paper, we propose **Math-Minos**, a natural language feedback enhanced verifier by constructing automatically-generated training data and a two-stage training paradigm for effective training and efficient inference. Our experiments reveal that a small set (30k) of natural language feedbacks can significantly boost the performance of the verifier by the accuracy of 1.6% (86.6% $\rightarrow$ 88.2%) on GSM8K and 0.8% (37.8% $\rightarrow$ 38.6%) on MATH.

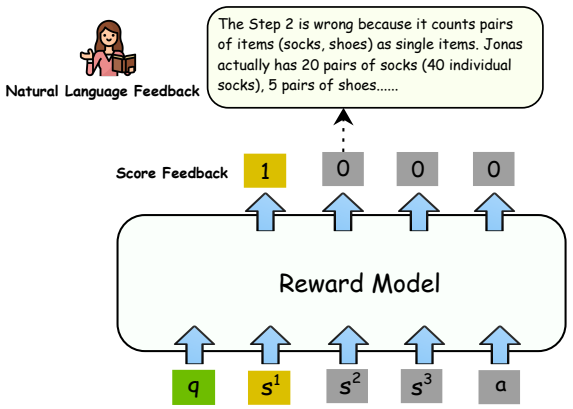


Figure 1: Illustration of **Score Feedback** and **Natural Language Feedback**. $q$ represents the mathematical questions $s_1, s_2, s_3$ represent the intermediate solutions. $a$ represents the final answer. 0 and 1 represent the score feedbacks. Our work aims to mitigate the insufficiency of score feedbacks and enhance verifiers' evaluation capabilities by introducing step-wise natural language feedbacks.

## 1 Introduction

Large Language Models (LLMs) (Bai et al., 2023; Touvron et al., 2023a,b; Jiang et al., 2023; OpenAI et al., 2024) have demonstrated strong capabilities in summarization (Touvron et al., 2023b), coding (Rozière et al., 2024), tool using (Song et al., 2023) and dialogue (Ouyang et al., 2022). However, mathematical reasoning remains a challenge for LLMs (Lightman et al., 2023; Huang et al., 2024). To tackle this problem, recent research has focused on using verifiers to validate the correctness of response generated by models (Wang et al., 2023b; Zhu et al., 2023; Li et al., 2023; Wang et al., 2024). An effective verifier can serve as 1) response reranker in the decoding (Li et al., 2023; Yu et al., 2024a; Wang et al., 2024). 2) reward model in RLHF (Shao et al., 2024); 3) data purifier that filters erroneous responses in the SFT (Rafailov et al., 2023);

However, existing verifiers are all trained as binary classifiers by adding a classification head to an LLM. We argue that the score feedbacks as binary classification labels are not informative in training as they do not contain explanations for the underlying reasons for the errors, which causes inefficient training.

In this work, we aim to enhance the verifier's evaluation ability for mathematical solutions by introducing step-level natural language feedbacks as rationale labels (i.e., the correctness of the current step and the explanations). We propose **MATH-Minos**, a natural language feedback enhanced verifier as section 3. By employing supervised fine tuning on only 30k training data with natural language feedbacks as rationale labels before binary classification training, we can effectively enhance the model's evaluation capabilities. In the first stage, we create high-quality step-level natural language feedback data as subsection 3.2. In order to address the challenge of accurate evaluation generation, we introduce Label-aware Natural Language

Feedback Curation to simplify the task by introducing step-level binary classification labels to enhance GPT-4's evaluation generation. The natural language feedbacks can provide in-depth reasons behind classification feedbacks, which is helpful for training the verifier. In the second stage, we introduce a two-stage training for **MATH-Minos** as subsection 3.3: firstly, we adopt the supervised fine tuning on rationale labels to effectively help improve the model's evaluation capabilities, followed by standard ORM & PRM training on score feedbacks to achieve efficient inference with a single forward step.

The experiments in section 4 demonstrate that infusing the model with evaluation capabilities via natural language feedback sas rationale labels is more efficient and effective than score feedbacks. We show that only 30k training data with natural language feedbacks can significantly boost the performance of mathematical verifiers. For Outcome Reward Model (ORM) setting, **MATH-Minos** improves the accuracy of MetaMath-Mistral (Yu et al., 2024b) by 1.6% (86.6% $\rightarrow$ 88.2%) on GSM8K and 0.7% on MATH % (37.6% $\rightarrow$ 38.3%) for ORM. For Process Reward Model (PRM) setting, **MATH-Minos** improves the accuracy by 0.7% (87.1% $\rightarrow$ 87.8%) on GSM8K and 0.8% (37.8% $\rightarrow$ 38.6%) on MATH.

In summary, our contributions are threefold:

1. We are the first to conduct in-depth analyses on the reasons behind incorrect evaluations generated by verifiers, revealing the shortcomings of current verifier's training paradigm and inspiring future research.

2. We propose and demonstrate that training verifiers with natural language feedbacks can complement the non-informative score feedbacks thus enhancing the model's evaluation ability.

3. We propose *MATH-Minos* by proposing label-aware natural language feedback curation and two-stage training paradigm.

4. We demonstrate the effectiveness of MATH-Minos across both ORM and PRM task settings. Extensive analysis demonstrates the superiority of the proposed method.

## 2 Related Works

**Enhancing the mathematical reasoning ability of LLM** Previous works focus on improving the mathematical reasoning ability of LLMs on three ways: (1) Pre-training: LLMs (Azerbayev et al., 2023; OpenAI et al., 2024; Touvron et al., 2023a,b) are pre-trained on a large set of corpus related to mathematical questions with next-token prediction objective. (2) Supervised fine-tuning: Supervised fine-tuning can also improve the mathematical reasoning ability of LLMs by training LLMs with mathematical questions with detailed solutions (Yu et al., 2023b; Luo et al., 2023; Wang et al., 2023a). (3) Inference: (Wei et al., 2022; Fu et al., 2022; Zhang et al., 2023; Bi et al., 2023) design prompting strategies to improve the reasoning ability of LLMs.

**Verifier for mathematical reasoning** Previous mathematical verifiers can be mainly categorized into two categories: Outcome Reward Model (ORM) gives an evaluation score to the whole solution; Process Reward Model (PRM) gives an evaluation score to each intermediate step of the solution. Previous works (Yu et al., 2023a; Ying et al., 2024; Wang et al., 2024) use question-solution pair data with a score to train a ORM or a PRM, which is inefficient to help models understand the errors. Therefore, in this work, we aim to train a verifiers with error types and detailed explanations about the errors.

## 3 Methodology

In this section, we introduce the background of our proposed method (§3.1), then delve into our proposed **MATH-Minos**, which contains label-aware natural language feedback curation (§3.2) and the two-stage model training (§3.3).

### 3.1 Background

**Outcome Reward Model** For a given problem $p$, the Outcome Reward Model (ORM) assigns a reward $r \in \mathbb{R}$ based on the whole completion $s$. The common approach for training an ORM involves implementing a binary sequence classification, which adds a classifier at the end of the LLM. The training loss is represented as follows:

$$\mathcal{L}_{orm} = y_s \cdot log(\hat{y}_s) + (1 - y_s) \cdot log(1 - \hat{y}_s), \quad (1)$$

where $y_s$ is the golden label of the solution and $\hat{y}_s$ is the sigmoid score of $s$ predicted by ORM. For mathematical reasoning tasks, the quality of a sample can be directly determined by judging the correctness of the result. Therefore, the general approach to train a ORM involves using a generator to provide completions. Subsequently, rule-based

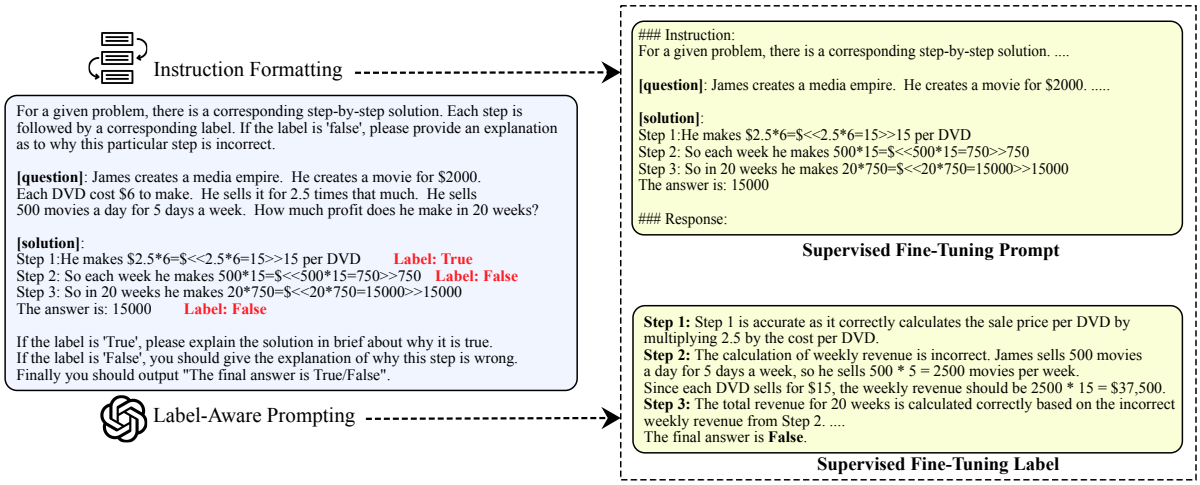


Figure 2: The illustration of the label-aware natural language feedback curation of MATH-Minos on GSM8K (Cobbe et al., 2021) dataset. We introduce step-level classification label to achieve step-level natural language feedback curation.

| Task | GSM8K | MATH |
|---|---|---|
| Outcome Evaluation | 95.1 | 62.0 |
| Process Evaluation | 85.1 | 59.7 |

Table 1: The step-level Acc. score of prompting GPT-4 to generate natural language feedback.

matching is employed to determine the correctness of the current completion, and this outcome is used as the label for training. For the sake of simplicity and comparability, we directly modified the open-sourced dataset provided by Wang et al. (2024) as the training set of ORM.

**Process Reward Model** For a given question $q$, the Process Reward Model (PRM) assigns a reward $r \in \mathbb{R}$ to each step $s_i$ of the completion $s$. The training of PRM is through the task of token classification, the training loss can be represented as follows:

$$\mathcal{L}_{prm} = \sum_{i=1}^{K} y_{s_i} \cdot log(\hat{y}_{s_i}) + (1 - y_{s_i}) \cdot log(1 - \hat{y}_{s_i}), \quad (2)$$

where $K$ is the number of reasoning steps of the completion, $y_{s_i}$ is the golden label of the solution and $\hat{y}_{s_i}$ is the sigmoid score of $s_i$ predicted by PRM. Compare to ORM, PRM can provide fine-grained supervision which is more detailed and reliable.

### 3.2 Label-aware Natural Language Feedback Curation

In this section, we introduce the label-aware natural language feedback curation of MATH-Minos as shown in Figure 2. Since the natural language feedback can be understood by both humans and large models, it is suitable to stimulate the evaluation capabilities of LLMs. Unlike the binary label, natural language feedback provides detailed explanations for right or wrong completions, which also brings complexity to data collection. The best way for generating the natural language feedback data is through manual annotation. Considering the costs associated with human annotation, we obtain natural language feedback by leveraging the capabilities of the most advanced LLM, GPT-4-turbo (OpenAI et al., 2024).

To verify the quality of data, we sample data from the Math-Shepherd (Wang et al., 2024) and PRM800K (Lightman et al., 2023) to create an evaluation dataset including 500 question-solution samples with step-level and outcome-level binary classification label for GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021). We then check the accuracy of outcome evaluation and step-level evaluation, with results presented in Table 1. Experimental results show that this prompting manner doesn't yield high-quality data with only 85.1 step-level accuracy for GSM8K and 59.7 step-level accuracy for MATH. This also indicates that one of the factors limiting the performance of the reward model is the base model's evaluation capability.

To facilitate GPT-4 in generating higher quality data, we propose a label-aware prompting method, which simplify the evaluation task by introducing the binary classification label within the prompt. As illustrated in Figure 2, GPT-4's task shifts from

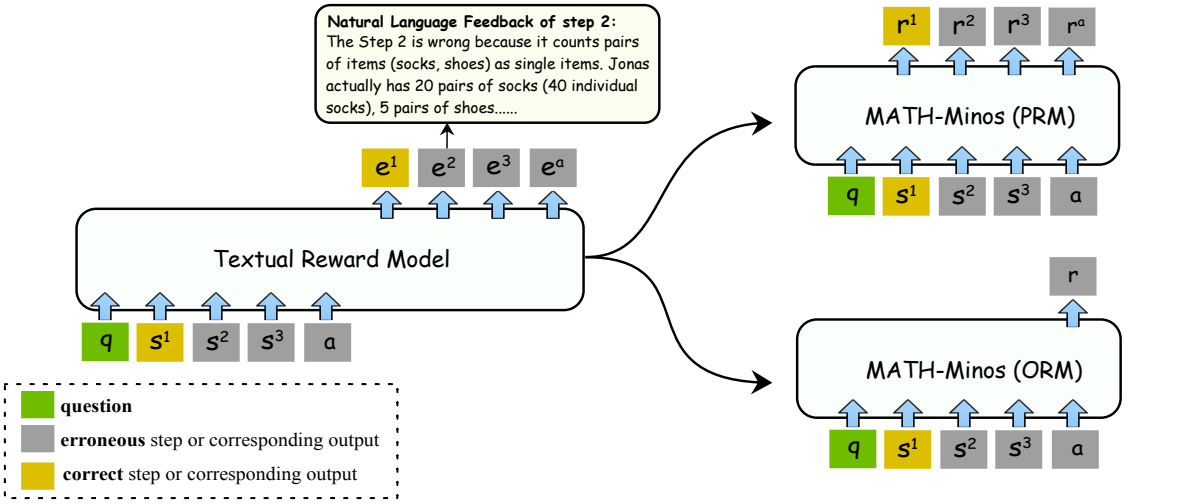


Figure 3: The overview of the two-stage training process of MATH-Minos. In Stage 1, training reward model (RM) with natural language feedbacks helps RM learn to evaluate effectively and efficiently. In Stage 2, training RM as binary classification helps RM inference efficiently by outputing a reward with one single forward pass.

determining correctness and generating explanations to generating explanations based on the given label. Extensive analysis in Section 5 have also validate the effectiveness of our approach.

### 3.3 Two-Stage Training of MATH-Minos

Based on the aforementioned data generated in Section 3.2, we introduce a novel two-stage training paradigm including (1) Stage 1: Training with Step-Wise Natural Language Feedback and (2) Stage 2: Training with Binary Classification to synergistically combine the strengths of evaluation generator and discriminator, which is shown in figure 3. This training paradigm enjoys two potential benefits: Firstly, natural language feedback contains rich information, especially for complex reasoning tasks such as mathematics. Therefore training with natural language feedback can significantly improve the models' evaluation ability with just a small set of data. Secondly, the inference of binary classification discriminator is more efficient compared with natural language feedback generation. This approach not only allows model to generate a score but also enables the model to produce evaluation results with just a single forward pass, thereby enhancing the efficiency.

**Reward Modeling with Natural Language Feedback** In the first stage, we employ supervised fine-tuning to enhance the evaluation capabilities of the model. We utilize the *Supervised Fine-Tuning Prompt* shown in Figure 2 as the input $x_{q,s}$ for the model, with the *Supervised Fine-Tuning Label* generated by GPT-4 serving as the model's output $y$. The training loss for a sample can be defined as follows:

$$L_{textrm} = \sum_{t=1}^{M} logP(y_t|y_{<t}, x_{q,s}), \qquad (3)$$

where $M$ is the total length of $y$ and $y_{<t}$ is the previous tokens.

**Reward Modeling with Binary Classification** After the first stage, the evaluation capability of the model is improved. However, natural language feedback cannot provide a score and thus can't be used as a reward for further optimizations like PPO. (Schulman et al., 2017) Additionally, when the model generates feedback, it produces a complete evaluation with rationales, making it significantly less efficient than using a classification-based verifier. Therefore, we further train the verifier with binary classification labels as Equation 1 and Equation 2.

Benefiting the proposed two-stage training, we can enhance the verifier's evaluation ability with natural language feedbacks and efficiently apply the verifier to PRM or ORM with one single forward pass.

## 4 Experiment

### 4.1 Experiment Setup

**Dataset** We conduct our experiment on two widely used mathematical datasets GSM8K (Cobbe

| Models | Verifier | GSM8K | MATH500 |
|---|---|---|---|
| Mistral-7B: MetaMATH | Self-Consistency (Li et al., 2023) | 84.1 | 34.6 |
| | ORM (Wang et al., 2024) | 86.2 | 35.9 |
| | PRM (Wang et al., 2024) | 87.1 | 36.7 |
| | Self-Consistency + ORM (Wang et al., 2024) | 86.6 | 37.6 |
| | Self-Consistency + PRM (Wang et al., 2024) | 86.8 | 37.8 |
| | MATH-Minos (ORM) † | 87.3 | 37.4 |
| | MATH-Minos (PRM) † | 87.6 | 37.8 |
| | Self-Consistency + MATH-Minos (ORM) † | **88.2** | 38.3 |
| | Self-Consistency + MATH-Minos (PRM) † | 87.8 | **38.6** |

Table 2: Main results of MATH-Minos in verification. The verification is based on 256 outputs. † denotes the method is proposed in this paper. Our **MATH-Minos** significantly outperforms baselines in both ORM and PRM settings.

et al., 2021) and MATH (Hendrycks et al., 2021). GSM8K(Cobbe et al., 2021) comprises a variety of word problems that are typically found in grade school mathematics curricula, which contains 7473 samples in the training set and 1319 samples in the test set. MATH (Hendrycks et al., 2021)is a diverse collection of mathematical problems that cover a broad range of topics and skill levels, from elementary to advanced mathematics, which contains 7500 samples in the training set and 5000 samples in the test set. In the setting of verification, we sample the test set of MATH to 500 samples which is identical to Lightman et al. (2023).

**Verification** Following Lightman et al. (2023) and Wang et al. (2024), we adopt the best-of-N selection to evaluate the capability of our verifier. Specifically, given a question $q$ and a generator, we let the generator sample $N$ times for the question $q$. Then, the verifier is used to evaluate the quality of each completion. The final answer $a$ is determined as the one with the highest reward according to the verifier's output $RM(q, a_i)$, formally expressed as follows:

$$a_{\text{rm}} = \mathcal{F}(\arg\max_{s_i} RM(q, s_i)), \quad (4)$$

where $s_i$ is the i-th solution generated by generator and $\mathcal{F}(\cdot)$ denotes extracting the final answer from the solution.

Following Li et al. (2023) and Wang et al. (2024), we also explore the ensemble of self-consistency (majority voting) and the verifier. Specifically, we classify the results output by the model into different groups and calculate the cumulative reward for each group, which can be calculate as follows:

$$a_{\text{sc+rm}} = \arg\max_{a} \sum_{i=1}^{N} \mathbb{I}(\mathcal{F}(s_i) = a) \cdot RM(q, s_i), \quad (5)$$

where $N$ is the number of solutions, $s_i$ is the solution generated by generator and $\mathcal{F}(\cdot)$ denotes extracting the final answer from the solution.

**Experimental Setting** For the training of the verifiers, to ensure comparability and convenience, we utilize the open-source MATH-shepherd dataset (Wang et al., 2024) for both baseline reward models and MATH-Minos. We curate total 30K samples of natural language feedback using the data from phase-one of PRM800K (Lightman et al., 2023) and the subset of MATH-Shepherd. Our main experiment conducts the verification on the test set of GSM8K and MATH. We use the MetaMATH-Mistral as the generator for the questions in the test set. In order to ensure the model has the ability of solving mathematical problem before learning to evaluate, we also use the MetaMATH-Mistral as the base model for MATH-Minos and all other reward models. For the training of natural language feedback, we use 30k training data generated as subsection 3.2 with learning rate of 5e-6 with total batch size of 256. For the training of score feedback, we use 440k training data (i.e., 30k data with the binary classification labels from the training data in the training of natural language feedbacks and 410k data sampled from MATH-Shepherd. For the training of baseline, we use the total 440k training data from MATH-Shepherd. For the training of ORM, we adopt the learning rate of 3e-6 with the batch size of 512. For PRM, the learning rate is 2e-6 with the batch size of 512.

### 4.2 Main Result

We present the performance of various methods in Table 2. Compared to traditional ORM, MATH-Minos (ORM) achieves an improvement of 1.1% in accuracy on the GSM8K and 0.7% in on the MATH.

For PRM, MATH-Minos (PRM) achieves an accuracy improvement of 0.7% on GSM8K and 0.8% on the MATH. Ensembling with self-consistency and MATH-Minos, the MetaMATH-Mistral generator achieves optimal accuracy of 88.2% on GSM8K and 38.6% on MATH500.

Beyond the improvement of the performance, we find that MATH-Minos has a more pronounced effect in the setting of ORM. We believe this phenomenon could be attributed to the sparser supervision in ORM compared to PRM, implying that information-rich textual explanations can offer more substantial benefits to ORM.

## 5 Analysis

### 5.1 Error Distributions of the Math Solvers

To further illustrate the shortcomings of training the verifier solely relying on binary classification, we conduct an in-depth investigation into the errors produced by the generator at the step level. Specifically, we take the natural language feedback generated by GPT-4 as a reference and heuristically and categorized the causes of errors in responses into five distinct types: ***Unrelated***: This indicates that the step is irrelevant and does not contribute towards deducing the final answer. ***Accumulation***: This denotes that the step is incorrect due to a mistake in the preceding step, leading to subsequent errors. ***Calculation***: This categorization is reserved for errors arising from incorrect calculations, which is one of the most common errors in mathematical reasoning. ***Logic***: This applies to steps that are logically flawed in the context of solving the given problem. ***Other***: This category encompasses steps that are erroneous for reasons not covered by the aforementioned categories.

We use the same way as the label-aware prompting introduced in Section 2 to automatically analyze the cause of errors. We obtain the step-level labels from the subset of MATH-Shepherd (Wang et al., 2024), which contains 500 samples for both GSM8K and MATH. Given the question, solution and the natural language feedbacks, we employ GPT-4 for the classification of error causes. The experimental result is shown in Figure 4.

From our statistical analysis, it is evident that the model produces errors across all types. For the MATH dataset, given it higher difficulty level and the necessity for more steps, a greater total number of errors occur within the same number of samples of GSM8K. Furthermore, we discovere that

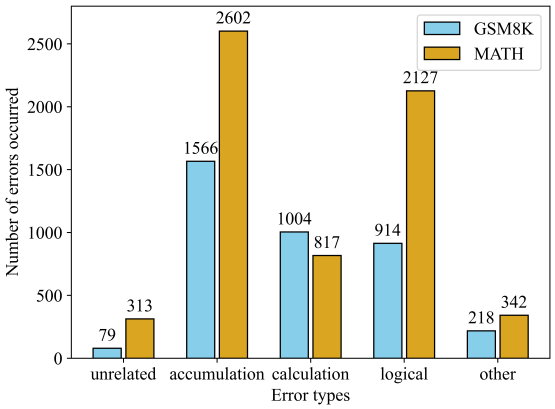


Figure 4: Statistics on the types of reasoning errors of MetaMath-Mistral on the GSM8K and MATH.

the most common cause of errors in both datasets is accumulation, which is consistent with our intuition. In multi-step reasoning, a mistake in one step is likely to directly cause errors in all subsequent steps. Furthermore, we observe distinct patterns of errors in the GSM8K and MATH datasets. For the GSM8K dataset, the occurrences of calculation errors and logical errors were approximately the same. Instead, in the MATH dataset, logical errors significantly outnumber calculation errors. This also indirectly demonstrates that models are vulnerable in more complex reasoning tasks.

These findings further illustrate that using binary labels to supervise the learning of reasoning evaluation tasks is insufficient and therefore highlighting the proposal for using natural language feedback to supplement the training of vanilla ORM or PRM.

### 5.2 Meta-Evaluation and Convergence Curves

To measure the verifier's capabilities in a more convenient and direct manner instead of verification, a intuitive approach is to assess whether the verifier can accurately determine the correctness of the final answer. Without the influence of the generator, this method purely relies on the capability of the verifier. We construct a meta-evaluation set based on whether the final answer provided in the generator's output is correct, serving as the ground truth label (despite the potential for false positives). By sampling several responses from the generator on the test set and deriving labels through rules, we create a meta-evaluation set for GSM8K containing 20,000 samples. We conduct tests on the meta-evaluation set using the checkpoints of each epoch after completing the model training and verification. The results of the meta-evaluation are

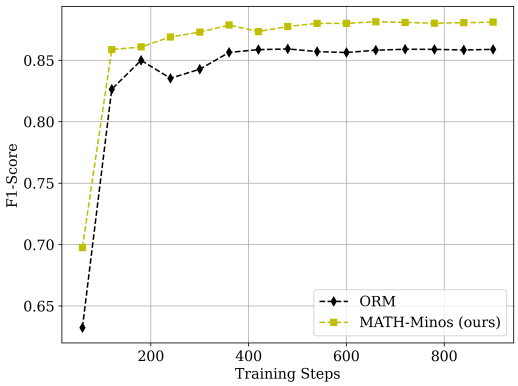


Figure 5: The convergence curve of vanilla ORM trained with score feedbacks (i.e., 440k quesiton-solution data with classification labels) and our **Math-Minos** with natural language feedbacks (i.e., 30k quesiton-solution data with rationale labels) on GSM8K.

| GSM8K | **Meta-Eval** | **Verification** |
|---|---|---|
| ORM | 85.9 | 86.2 |
| + curation w/o label | 86.2 | 85.8 |
| + curation w/ label | 88.1 | 88.2 |

Table 3: Experimental result of the ORM, ORM with vanilla natural language feedback curation and our Math–Minos (ORM with label-aware natural language feedback curation). Label-aware natural language feedback curation significantly enhances ORM's evaluation ability.

presented in Figure 5.

We observe that MATH-Minos consistently outperforms Vanilla ORM in meta-evaluation. Additionally, MATH-Minos exhibit a faster convergence rate, surpassing the baseline at only approximately 120 steps. Given that we trained for only 1 epoch, this means that in the actual secondary phase of binary classification, only about 60K data are required to exceed the baseline. Hence, the experiment demonstrates that natural language feedback can significantly reduce the amount of data needed to train a verifier.

### 5.3 Influence of the Data Quality

To validate the effectiveness of Label-Aware Natural Language Feedback Curation, we conduct a comparative experiment against directly prompting. We use the 30K direct GPT-4 evaluation which is the same number of MATH-Minos to compare. We use both meta-eval and verification to measure the capability of the verifier. The experimental result

| GSM8K | **Meta-Eval** | **Verification** |
|---|---|---|
| ORM w/o stage 1 | 85.7 | 86.2 |
| RM w/o stage 2 | 82.8 | 84.7 |
| MATH-Minos | 88.0 | 88.2 |

Table 4: Ablation study of the two-stage training paradigm. RM w/o stage 1 denotes that we only train the verifier with the score feedback. RM w/o stage 2 denotes that we only train the verifier the natural language feedback generated.

is shown in Table 3.

The experimental results indicate that directly prompting GPT does not significantly enhance the performance of the verifier. This is possibly due to the quality of the data shown in Table 1. The hallucinations produced by GPT-4 can further accumulate in the verifier, thus affecting the final performance of the model.

### 5.4 Ablation Study

Table 4 presents the results of our ablation study, wherein we delve into the effect of each stage.

Removing stage 1 essentially reverts our method to a vanilla ORM, as shown in the table. Without training on natural language feedback, the model is unaware of the reasons behind what makes an answer correct or incorrect. Hence, the performance of the binary classification in the second stage noticeably declines compared to MATH-Minos.

When eliminating stage 2, binary classification, an intuitive approach is to directly utilize the natural language feedback generated by the text reward model for the generator's verification. Given that the model outputs a binary discrete value ('True' or 'False'), we cannot employ a best-of-N verification but instead use it as a filter. Specifically, we apply self-consistency in filtering out cases where the model output is 'True'. Unfortunately, we observe that relying solely on natural language feedback from text reward model leads to a significant decline in performance. The probable reasons may include: 1) Upon closer inspection, we notice inconsistencies in the model's feedback. This is characterized by samples where a step is recognized as incorrect yet the overall outcome is deemed correct, and vice versa. Such inconsistencies are even found in the strongest models such as GPT-4, despite their ability to provide accurate explanations. 2) The performance of evaluation might be constrained by the model's scale. Influenced by computational resources, we do not further explore larger mod-

| | Recall | Avg. Reward |
|---|---|---|
| ORM | 0.74 | 0.234 |
| MATH-Minos (ORM) | 0.92 | 0.105 |

Table 5: The recall and average reward of the false positive examples (i.e., the final answer of the solution is true while the intermediate steps are false) of ORM and **MATH-Minos**. **MATH-Minos** significantly improves the evaluation towards false positive examples.

els. Evaluation generation tasks could be more challenging for models of smaller scale. 3) The binary discrete output of the model is relatively coarse-grained. For instance, two examples judged as correct cannot be compared with each other.

In summary, in this section, our experiments demonstrate that both stage one and stage two are essential, where natural language feedback and binary classification play complementary roles.

### 5.5 Performance on False Positive Samples

To further investigate the efficacy of Math-Minos, we analyze the performance on false positive samples within the training dataset. False positive samples refer to those instances that have a correct final outcome but contain errors in the intermediate steps. Ideally, a robust verifier should assign lower rewards to these samples. We extract such examples from the training set of the verifier, amounting to a total of 600 samples, which includes data from both GSM8K and MATH datasets. We test the performance of both ORM and Math-Minos, with the experimental results presented in Table 5.

According to our experimental findings, it turns out that vanilla ORM can correctly discriminate a majority of the false-positive samples from the training set but with an accuracy significantly lower than MATH-Minos. It achieves a recall of 74% with an average reward of 0.234. While MATH-Minos reach a recall rate of 92% with an average reward of 0.105. This performance is significantly better than that of ORM not trained on the natural language feedback. Delving into the data, we discover that in the context of false positives, a substantial portion of the natural language feedback generated by GPT-4 are contradicted to the "True" labels we assigned. We believe that these data endows MATH-Minos with a stronger capability to discern false-positives, thereby enhancing the model's performance.

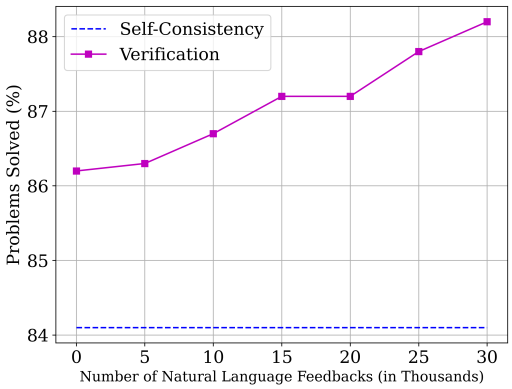


Figure 6: The impact of different amount of natural language feedback on the performance of the verifier in GSM8K. This shows the scalable potential of our **MATH-Minos**.

### 5.6 Influence of the Data Amount

Figure 6 illustrates how different amounts of natural language feedback affect the model during the first stage. We use SFT on the model in the first stage using different scales of natural language feedback. In the second stage, we adopt the setup of ORM setting and use the verifier to select the best-of-N of GSM8K test set. We observe a positive correlation between the model's performance and the quantity of natural language feedback provided in stage one, which implicitly evidences the benefit of natural language feedback for the model.

## 6 Conclusion

We analyze the current training paradigm of verifiers, demonstrating that score feedback from binary classification labels is suboptimal for teaching LLMs to accurately evaluate mathematical solutions. By introducing rationale labels that provide detailed explanations of error types, our training paradigm significantly enhances the verifier's evaluation ability. The experimental results show that models trained on a small dataset with natural language feedback (30k instances) significantly outperform the baselines that rely solely on classification labels. This highlights the critical role of rich and informative labels in training data in crafting more nuanced and effective training strategies for the development of large language models (LLMs) that are capable of complex reasoning tasks. Finally, the findings of this work pave the way for the potential integration of natural language feedback with classification verifiers.

## 7 Limitations

Following the scaling laws, the evaluation ability of a model, especially in terms of generating natural language evaluations, may vary across different sizes. However, due to computational resource limitations, experiments were conducted solely on a model with 7 billion parameters, thereby unable to explore the impact of model's scaling on the evaluation ability. We leave it into our future work.

## References


Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2023. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *Preprint*, arXiv:2309.16609.

Zhen Bi, Ningyu Zhang, Yinuo Jiang, Shumin Deng, Guozhou Zheng, and Huajun Chen. 2023. When do program-of-thoughts work for reasoning? *arXiv preprint arXiv:2308.15452*.

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. *Preprint*, arXiv:2110.14168.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

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. *Preprint*, arXiv:2310.01798.

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. *Preprint*, arXiv:2310.06825.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

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. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Yifan Song, Weimin Xiong, Dawei Zhu, Wenhao Wu, Han Qian, Mingbo Song, Hailiang Huang, Cheng Li, Ke Wang, Rong Yao, Ye Tian, and Sujian Li. 2023. Restgpt: Connecting large language models with real-world restful apis. *Preprint*, arXiv:2306.06624.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Peiyi Wang, Lei Li, Liang Chen, Feifan Song, Binghuai Lin, Yunbo Cao, Tianyu Liu, and Zhifang Sui. 2023a. Making large language models better reasoners with alignment. *arXiv preprint arXiv:2309.02144*.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *Preprint*, arXiv:2312.08935.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Huaiyuan Ying, Shuo Zhang, Linyang Li, Zhejian Zhou, Yunfan Shao, Zhaoye Fei, Yichuan Ma, Jiawei Hong, Kuikun Liu, Ziyi Wang, Yudong Wang, Zijian Wu, Shuaibin Li, Fengzhe Zhou, Hongwei Liu, Songyang Zhang, Wenwei Zhang, Hang Yan, Xipeng Qiu, Jiayu Wang, Kai Chen, and Dahua Lin. 2024. Internlm-math: Open math large language models toward verifiable reasoning. *Preprint*, arXiv:2402.06332.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2023a. Outcome-supervised verifiers for planning in mathematical reasoning. *arXiv preprint arXiv:2311.09724*.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024a. Ovm, outcome-supervised value models for planning in mathematical reasoning. *Preprint*, arXiv:2311.09724.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024b. Metamath: Bootstrap your own mathematical questions for large language models. *Preprint*, arXiv:2309.12284.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Xinyu Zhu, Junjie Wang, Lin Zhang, Yuxiang Zhang, Yongfeng Huang, Ruyi Gan, Jiaxing Zhang, and Yujiu Yang. 2023. Solving math word problems via cooperative reasoning induced language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.