Rule Compliance in Language Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

In our previous research observations, we noticed that large language models, which do not rely on external auxiliary tools, have accuracy issues when performing logical operations. Language models typically rely on the surface patterns of logical problems rather than deeply understanding their underlying logic. This limitation might stem from the model's inadequate learning methods for logical operations. Taking mathematical calculations as an example, considering the infinity of mathematical operations, models struggle to accurately complete calculations when trained only on a limited set of example data sets. Inspired by meta-learning, we expect models to not only learn the data and solutions for specific tasks but, more importantly, to learn strategies and methods of learning.

In this paper, we propose a new architecture for language models. This model can precisely complete numerical calculations and other complex logical operations by learning and combining different rules. The model is based on the Transformer architecture and is trained through a dynamic task composition process. By designing a dataset that encompasses basic rules, compound rules, and iterative rules for mathematical calculations, these rules are textualized and embedded into the model for pre-training, resulting in the MetaRuleGPT pre-trained model capable of handling complex numerical operations. Experimental results show that, especially in high-digit calculations, our model's accuracy surpasses that of existing well-known large language models, such as Google's Palm, Llama2, and Alibaba's QWen, and can compete with ChatGPT-3.5 and ChatGPT-4. Moreover, for more complex mathematical logic reasoning problems, MetaRuleGPT can mimic human rule-following capabilities, simplify complexity, and gradually deduce accurate calculation results.

1 Quotation

In the world of Natural Language Processing (NLP), large language models such as GPT-4 have made remarkable progress, and they have demonstrated amazing understanding and processing capabilities in a variety of tasks. However, these models still face considerable challenges when they encounter mathematical problems and other areas that require specialized knowledge. Taking mathematics problems as an example, they cover a wide range of content, including but not limited to basic addition, subtraction, multiplication and division, derivation, integration and equation solving. Despite their powerful language understanding capabilities, these models are still unable to solve basic mathematical addition, subtraction and numerical calculation problems. For example, for a simple high-digit addition problem:

2 Research Methodology

To enhance the accuracy and generalization of language models in solving complex logical reasoning and numerical calculation tasks, we introduce the MetaRuleGPT model. This model aims to bolster the reasoning capabilities and generalization potential of language models, inspired by the concept of meta-learning. MetaRuleGPT focuses on mastering general learning strategies to precisely complete complex logical deduction tasks by applying learned rules. The model dynamically integrates basic mathematical computation rules with higher-order operation rules, enhancing its ability to process rule combinations. Such a design allows the MetaRuleGPT model to exhibit superior accuracy and generalization capabilities when faced with complex logical reasoning challenges, such as mathematical reasoning problems.

The MetaRuleGPT model adopts a novel approach to handling complex arithmetic expressions. By incorporating iterative strategies into the model architecture, the model automatically matches arithmetic expressions to the most applicable rules for computation in each iteration. The computation process is not directly completed in one step but involves gradually approximating the final calculation result by parsing the expression step by step and applying composite rules, mimicking the human thought process in solving mathematical problems.

Furthermore, by adopting this strategy, the MetaRuleGPT model is not limited to handling a single task but is capable of learning and executing various different tasks. When dealing with multitasking, the rules across different tasks might intersect; our model can flexibly learn these intersecting rules and dynamically apply them to complete multiple tasks simultaneously, while keeping the tasks independent of each other without interference.

2.1 Specific Rule Learning for Arithmetic Tasks

Table 1: Summary Table of Learning Rules for Various Tasks

Train Rule Type	Numerical Addition	Numerical Subtraction	Vector Cross Product
Vector Table	-	-	√
Nine Addition Table	√	-	√
Nine Subtraction Table	-	√	√
Nine Multiplication Table	-	-	√
Mapping Rule	√	√	√
Carrying Rule	√	-	√
Borrowing Rule	-	√	√
Vector Product Rule	-	-	√
Compute Rule	√	√	√

In our research, the model demonstrated outstanding logical reasoning and generalization capabilities in performing three complex tasks: high-digit addition and subtraction calculations, and vector cross-product computations. For these tasks, we designed specific rule datasets for training. For example, during the training for addition calculations, the model was guided to learn key knowledge including single-digit addition rules, carry rules, digit mapping rules, and basic computation rules. By mastering these basic rules, after meticulous pre-training, our model could flexibly apply and combine these rules to accurately complete complex mathematical operations, including high-digit addition and subtraction.

After our model successfully mastered basic addition and subtraction operations, we planned to further extend its capabilities to perform vector cross-product computations. To achieve this goal, we introduced rules for vector representation and cross-product computation into the model to realize vector cross-product calculations. This means that once the model learns these new rules, it could combine the newly acquired rules with existing numerical computation rules to perform vector cross-product calculations. During the process of vector cross-product computation, the model needs to handle a large amount of complex derivation. Through gradual derivation, combining the right-hand rule for cross-products with basic numerical computation rules, the model will be endowed with the ability to compute vector cross-products. This strategy showcases the model's deep logical reasoning and strong generalization ability to solve more complex mathematical operations by learning and integrating various rules.

2.2 Arithmetic Training Dataset

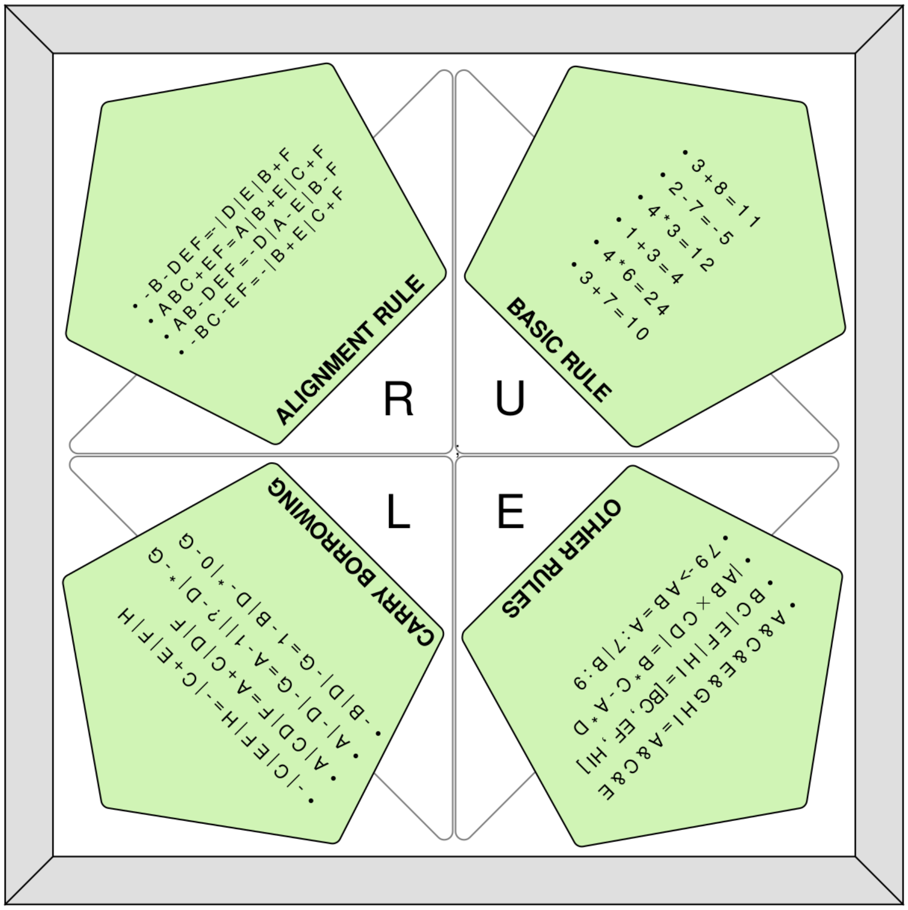

Figure 1: Partial Training Dataset Display

We meticulously designed the computation rule dataset for training, covering a wide range of arithmetic operations from the most basic single-digit arithmetic tasks to various complex arithmetic rules. This dataset, carefully planned, encompasses various arithmetic operations, including alignment rules, carry rules, borrow rules, basic computation rules, and composite rules. Our constructed dataset contains approximately 20,000 records.

In this dataset, each arithmetic expression involves 2 to 10 operational steps, involving a series of mathematical computation operations, such as addition (+), subtraction (-), and vector cross-product operations (Œ). This design aims to provide a comprehensive and diverse mathematical operation learning environment for the model.

In these datasets, the arithmetic expressions we trained only contain the most basic single-digit operations, such as the addition and subtraction tables for single digits, and other simple calculations. Additionally, the dataset includes a series of meticulous mathematical computation rules, including digit alignment rules, carry rules, borrow rules, cross-product rules, and digit mapping rules. This design aims to provide a solid foundation for the model to master the combination of key rules required for basic to complex mathematical operations.

2.3 MetaRuleGPT Model Structure

To closely mimic the natural process of humans solving mathematical problems, we did not directly solve each complex arithmetic expression but adopted an iterative and stepwise strategy. Through this method, our model breaks down complex expressions into a series of simpler and basic computational steps, reasoning the final answer step by step. This approach enables the language model

Table 2: Model Parameter Size Comparison Table

MetaRuleGPT Model	Dimension	Batch size	Heads	Layers	Parameters	Trainning Steps
MetaRuleGPT-10M	256	10	16	5	10M	3000
MetaRuleGPT-30M	256	30	16	15	30M	3000
MetaRuleGPT-100M	256	100	32	20	100M	3000

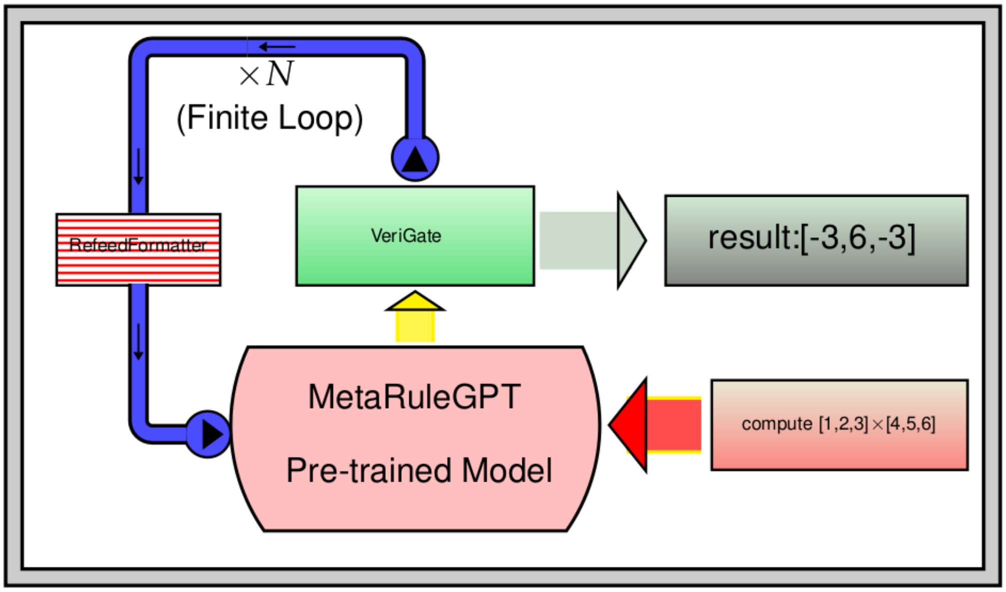

Figure 2: MetaRuleGPT Model Architecture Diagram

to have a deeper understanding and more effective application of specific rules during the learning process, allowing for flexible combination and application of these rules in problem-solving. Our model excels in mathematical calculation tasks, mainly due to its mastery of core computation rules rather than merely relying on memorization of specific cases.

Focusing on arithmetic tasks, we developed a language model based on the Transformer architecture, aimed at solving mathematical problems, which we refer to as the MetaRuleGPT language model. The model architecture, as shown in the figure, includes several key components: the MetaRuleGPT pre-trained model, the RefeedFormatter (formatting tool), and VeriGate (verification gate). We designed a self-iterative method that allows

2.4 MetaRuleGPT Pre-trained Model

As shown in Figure 3.3, we have trained the dataset using a language model based on the Transformer architecture. To flexibly adjust the model's parameter size and internal structure, we designed and implemented a custom Transformer model. In the design phase of the model, we selected a configuration with 16 attention heads and 15 layers for both the encoder and decoder, and we specifically chose carefully selected position encoding and word embedding strategies. Given that the problems we face do not involve a complex vocabulary, we adopted a single-byte-based training method. This training strategy has clear advantages and significance compared to traditional word-based or character-based methods.

Byte-based language models provide a flexible and effective means for handling multilingual text and unknown characters. As shown in Figure 3.3, this is an example of using the Transformer model to train vector cross product calculation rules. By processing each character individually, the model can ensure more accurate learning of the rules, laying a solid foundation for solving complex logical tasks.

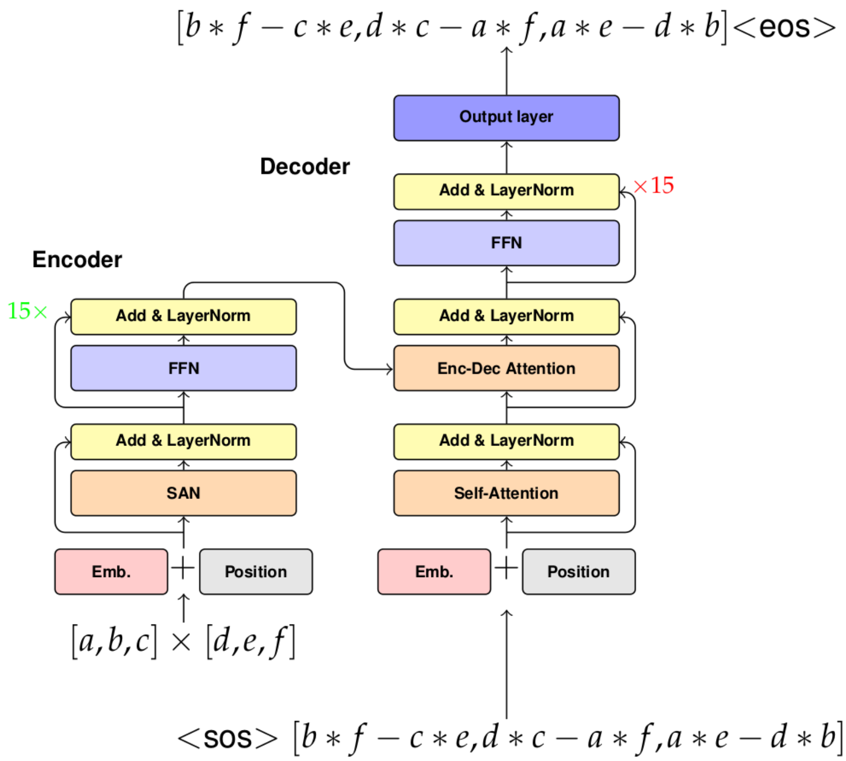

Figure 3: MetaRuleGPT Pre-trained Model

2.5 MetaRuleGPT Model Calculation Example

Figure 3.4 illustrates the internal workings of our model. To clearly demonstrate how the model operates, we use a simple addition example. When the input “Input: 78 + 263” is provided, it is processed sequentially through the Mapping Rule, Compute Rule, Align Rule, and Carry/Borrow Rule to derive the computation result. Figure 3.4 explains how the initial input is transformed into the final result.

1. First, the model structurally processes our input question, where “78 + 263” under the Mapping Rule becomes:

   $$a_1 : 7, b_1 : 8, c_1 : 2, d_1 : 6, e_1 : 3.$$

   The expression “$a_1b_1 + c_1d_1e_1$” through the Align Rule becomes:

   $$c_1|a_1 + d_1|b_1 + e_1.$$

   Through alignment, a combination of mapping rules produces the intermediate output: “2|7 + 6|8 + 3”.
2. Similarly, for “2|7 + 6|8 + 3”, a combination of the Mapping Rule and single-digit addition rule (Add Sub-rule) produces the intermediate output: “2|13|11”.
3. When “2|13|11” is input, the model invokes the Carry Rule and the Mapping Rule to perform digit carry operations, producing an intermediate output: “2 + 1|3 + 1|1”.
4. “2 + 1|3 + 1|1” as a new input, again applying the Mapping Rule and Compute Rule, leads to the final computation result: “3|4|1”.
5. “3|4|1” as the final input stage, our model invokes the formatting rules and uses special symbols for marking. Ultimately, the result is formatted using VeriGate to output: “Output: 341”.

In summary, in the computation process, the MetaRuleGPT model utilizes a combination of rules to align, carry, and output the final result, which is:

$$
\begin{array}{rl}
\text{Input: } & 78 + 263 \\
& \downarrow \\
& 2|7+6|8+3 \\
& \downarrow \\
& 2|13|11 \\
& \downarrow \\
& 2+1|3+1|1 \\
& \downarrow \\
& 3|4|1 \\
& \downarrow \\
\text{Output: } & 341
\end{array}
$$

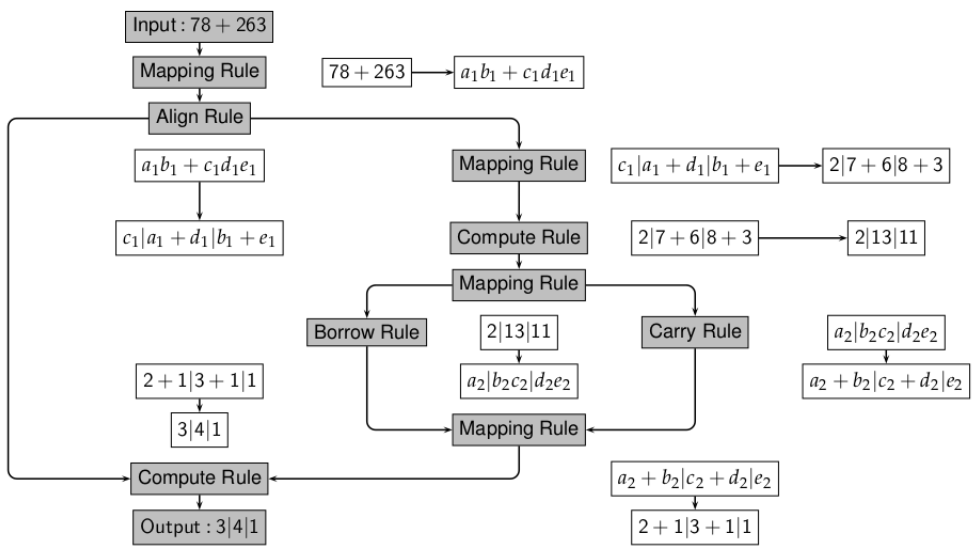

Figure 4: MetaRuleGPT

2.6 Compute Rule

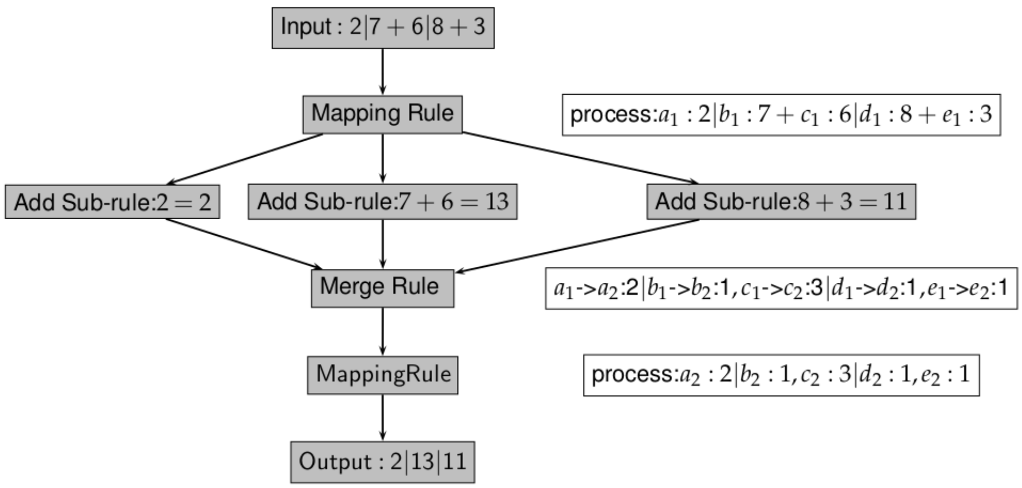

Figure 5: Compute Rule

As shown in Figure 3.5, the Compute Rule is a composite rule that operates as follows for the model input $2|7+6|8+3 a_1|b_1+c_1|d_1+e_1$: First, we need to identify and separate each part: 2, $7+6$, and $8+3$, and compute them individually. Specifically, our model recognizes each position and labels them in order:

$$a_1 : 2, b_1 : 7, c_1 : 6, d_1 : 8, e_1 : 3. \tag{3.9}$$

By separating and invoking the single-digit computation rule (Add Sub-rule) and the Mapping Rule on

$$a_1|b_1+c_1|d_1+e_1$$

we have

$$a_1 : 2 = a_2 : 2, \tag{3.10}$$

$$b_1 : 7 + c_1 : 6 = b_2 : 1, c_2 : 3, \tag{3.11}$$

$$d_1 : 8 + e_1 : 3 = d_2 : 1, e_2 : 1. \tag{3.12}$$

The intermediate output is processed using the Merge Rule:

$$a_2 : 2|b_2 : 1, c_2 : 3|d_2 : 1, e_2 : 1. \tag{3.13}$$

After processing with the Mapping Rule, the model completes the computation on the string and outputs:

Output: $2|13|11$

In summary, the model uses tagging techniques to identify and separate each part, applies learned basic addition sub-rules to perform numerical operations on each part, then merges them to produce and output the intermediate computational results.

3 Experiments

3.1 Experimental Setup

To demonstrate the exceptional accuracy and generalization capability of our MetaRuleGPT model in reasoning tasks, we meticulously designed two experiments: numerical arithmetic tasks and vector cross-product computation tasks. These experiments not only tested the model's basic computational ability but also its ability to solve complex problems, providing a solid foundation for comprehensively evaluating the model's performance in logical reasoning. Furthermore, to further prove the advantages of MetaRuleGPT, we compared it with several well-known large language models, including Alibaba's QWen, Google's Palm, Llama2, and the powerful and recent ChatGPT-3.5 and ChatGPT-4.0, to corroborate the superior performance of the MetaRuleGPT model.

3.2 Test Dataset

Current large language models exhibit certain limitations in handling mathematically rigorous problems, partly due to a lack of deep understanding of mathematical logic. In contrast, our model, built from the ground up on the fundamental principles of mathematics, demonstrates higher precision in solving math challenges. To validate this advantage, we meticulously designed various types of test data and prepared a detailed computation dataset. This was done to highlight the significant advantage of our model in mathematical reasoning. Through this series of validations, not only did we prove that our model could precisely grasp and apply the basic logic of mathematics, but it also showcased its powerful generalization ability in the problem-solving process.

In the domain of arithmetic tasks, we constructed a diverse training dataset containing a wide range of arithmetic operations. To comprehensively evaluate our model's computational accuracy and generalization ability, we designed an evaluation dataset containing 8,000 test cases, entirely non-overlapping with the training set. This dataset covers various types of numerical operations, including but not limited to perfect decimal addition, reverse magnitude subtraction, misplaced subtraction, and addition and subtraction operations based on randomly generated numbers.

Table 3: Partial Test Data Display Table

Data Type	Test Dataset Examples
Randomized Procedure	$6729132856 + 1854307391, \ldots, 1554887316 - 817095695$
Perfect Decadic Addition	$6659891948 + 340108052, \ldots, 4376628072 + 623371928$
Reverse Magnitude Subtraction	$62103 - 2386797965, \ldots, 53006 - 7764286617$
Interleaved Subtraction	$1824453209 - 482835016, \ldots, 8858241744 - 261714262$
Vector Cross Product	$(6, 5, 7) \times (9, 3, 1), \ldots, (8, 2, 0) \times (6, 4, 9)$

3.3 Evaluation Metrics

In evaluating the final computation results, we considered not only the model's calculation results matching the true answers, i.e., accuracy, but also the gap between the calculation results and the correct answers, that is, the difference ratio. Theoretically, the smaller the absolute value of the difference ratio, the closer the model's computation result is to the real value, indicating a greater possibility to improve the model's accuracy through appropriate parameter tuning. Conversely, when the difference ratio is greater than 1, it suggests the model lacks the capability to solve such problems, or it faces significant challenges in dealing with these types of issues.

Assuming the number of correctly predicted quantities is TP and the total number of predictions is N, then accuracy can be defined as:

$$Accuracy = \frac{TP}{N} \times 100\% \tag{1}$$

Suppose our model's computation result is y, the actual computation result N numbers in total, then our final overall difference ratio can be defined as:

$$DifferenceRatio = \frac{1}{N} \sum_{i=0}^{N} \left| \frac{y_i - \hat{y}_i}{\max(y_i, \hat{y}_i)} \right| \tag{2}$$

3.4 Deep Numerical Optimization Experiments on Language Models

To test our model's mathematical reasoning and generalization capabilities, we conducted comparisons using well-known language models such as Alibaba's QWen, Google's Palm, Llama2, and the currently very powerful ChatGPT-3.5 and ChatGPT-4.0. Through such comparisons, we could comprehensively understand the performance differences between different models and assess our model's performance on mathematical reasoning tasks.

We used the various test datasets we previously organized to invoke and test with the aforementioned large language models, preserving and comparing the computational results of each model. We conducted a series of detailed experiments and evaluations, and the results of the test datasets for the models we chose can be found in the appendix.

3.5 Language Model-Driven Vector Cross Product Calculation Experiment

To demonstrate our model's capability in handling complex logical problems, we have carefully selected the calculation of vector cross products, a more complex mathematical task, as a test case. Through this test, we not only verify the model's accuracy in computation but also compare it with the leading large language models in the current field. Table 4.7 details the comparison results of different models' accuracy on the dataset for vector cross product calculations.

Table 4: Vector Cross Product Table

Vector Compute	Cross Product
GPT-4	17%
GPT-3.5	5.5%
llama2-7b	-
llama2-13b	-
llama2-70b	0%
Google-PaLM	0%
Qwen-72b-Chat	23%
MetaRuleGPT	98.5%

4 Results and Discussion

Table 5: Language Models' Performance in Numerical Tasks

Model	Model Parameter	5-digit	10-digit
GPT-4	100000B	99.22%	90.9%
GPT-3.5	175B+	97.26%	83.9%
Llama2-7b	7B	22.3%	1.7%
Llama2-13b	13B	17.8%	1.6%
Llama2-70b	70B	57.76%	6.4%
Google-PaLM	110B	73.32%	26.6%
Qwen-72b-Chat	72B	91.32%	60.4%
MetaRuleGPT	30M	100%	100%

4.1 Test Data Results Analysis

4.1.1 Test Results

As demonstrated in Tables 4.2 - 4.6, to assess our model's performance in solving general numerical problems, we generated a large amount of experimental data with random numbers using Python. Preliminary results show that in low-digit addition and subtraction operations, our model and other tested language models achieved an accuracy rate exceeding 75%, demonstrating high computational precision. However, as the number of digits increased, the performance of most language models significantly declined. Except for ChatGPT, other models often made mistakes in handling high-digit calculations due to their inability to deeply grasp computational rules, nearly losing their computational capability.

It is particularly worth mentioning that even when facing high-digit random addition tasks, our model still maintained a 100% accuracy rate. Although it faced certain challenges in high-digit random subtraction tasks, our model still showed the highest accuracy among all tested language models, approximately 10% higher than ChatGPT. This achievement not only highlights our models good performance in solving complex numerical problems but also proves its generalization ability.

4.1.2 Standardized Maximum Error Analysis

The standardized maximum error metric is used to measure the relative size of a model's computational deviations. A value greater than 1 indicates that the model's calculations completely deviate from the true values. From Tables 4.2, 4.3, 4.4, 4.5, and 4.6, it can be seen that in five-digit numerical calculations, all types of models managed to keep the average standardized error below 1, meaning that even if the calculations were not completely accurate, the errors remained within a controllable range. However, when the complexity of the calculations was increased to ten digits, the average standardized errors of the Llama2-7b model exceeded 1, indicating that this model deviated significantly from true values and was unstable in handling high-digit calculations. The Llama2-13b model had accuracy comparable to the Llama2-7b, but its standardized error was smaller, making the calculations closer to true values. The Qwen-72b-Chat model had the smallest standardized error among all tasks, indicating that its results were relatively close to true values and more stable. Although ChatGPT-3.5 had high accuracy in some test tasks, its standardized error was larger. In contrast, ChatGPT-4.0 showed good performance both in accuracy and standardized error. The MetaRuleGPT model had high accuracy in multiple test tasks, and in Table 4.5, even in cases of rule execution errors, the standardized error on the calculation results was relatively small, making the results stable.

4.1.3 Vector Cross Product Results

From the data in Table 4.7, it is evident that the Llama2 models with 7b and 13b parameter sizes were even unable to perform vector cross product calculations, while the Llama2-70b, the largest parameter model of the Llama2 series, could perform cross product calculations but with an accuracy rate of 0%. Even the currently most powerful language model, ChatGPT, achieved an accuracy rate below 50% without the aid of external tools. In contrast, our model was able to accurately calculate vector cross products in three-dimensional space with an accuracy rate of 98.5%, further confirming that the method of enhancing model capabilities by combining different rules is effective. By comprehensively learning basic operational rules such as addition, subtraction, multiplication, and cross product, our model achieved precise invocation of these rules and successfully outputted accurate calculation results. More importantly, by training rules for two different types of tasks within the same pre-trained model, our model demonstrated multi-task generalization ability. This indicates that our model is not only adaptable to a variety of different task scenarios but can also identify and apply common rules among these tasks, significantly enhancing learning efficiency. This further showcases our models good performance and flexibility.

4.2 Discussion

4.2.1 Controllability

Although existing language models have demonstrated powerful capabilities, they still face challenges in terms of controllability. Particularly, most models struggle to precisely answer questions within a controlled range, often resulting in significant deviations. This is an important issue that current language models need to address. In contrast, our model strictly performs tasks according to rules, thus displaying relatively better controllability. Although this controllability may vary with the increase in tasks required to be performed, the rule-based execution generally ensures relative reliability. We also plan to add more task rules in subsequent model training to further evaluate the controllability of the model.

Furthermore, we hope our model also possesses strong controllability in handling tasks outside its capability range. For example, our model performs well in numerical computation tasks, but when attempting to solve function integration problems, it encounters unpredictable results, leading to significant errors. This is one of the current challenges we face. We plan to introduce more rule data in future optimization efforts to improve the overall controllability of the model, making it more stable and accurate in a wider range of application scenarios.

5 Summary and Outlook

5.1 Summary

Inspired by meta-learning, this study aims to explore the rule-following capabilities of language models, that is, the combinatorial skills and generalization abilities that humans display in problem-solving. Using numerical calculations as an example, we adopted a Transformer-based approach to construct a language model, MetaRuleGPT, utilizing an iterative strategy. Through in-depth training of a series of compound rules and their sub-rules, we successfully developed a pre-trained language model, MetaRuleGPT, with 3 million parameters. After learning basic single-digit arithmetic operations and some core computational rules, our model was able to handle high-digit calculations and more complex vector cross-product operations it had not previously encountered, demonstrating high accuracy and even surpassing current mainstream large language models in accuracy. Experiments have shown that without the aid of external computational tools, existing mainstream language models often struggle with high-digit calculations and other complex computational issues due to limitations in understanding. Our model not only accurately completed these challenging computational tasks but also demonstrated the generalization potential of language models in numerical calculations. This finding strongly supports our hypothesis: through rule-following, language models can exhibit certain generalization abilities.

5.2 Outlook

This research raises several issues worthy of further exploration, including:

1. Although our model has shown certain generalization and understanding abilities after rule learning, it is limited by computational resources, and the variety of problems it can handle is relatively limited. We look forward to expanding the model's parameter size and training with more rule datasets to enable the model to handle a broader range of logical tasks.
2. Our model has room for improvement in capturing the details of problems. MetaRuleGPT generally can mimic human solutions to numerical issues, but it still cannot reach human-like precision in some details. For example, the model often treats zero mechanically, unable to regard it as a special existence that is neither negative nor even, as humans do. This indicates a significant gap between our model's fine-grained problem-solving and human capabilities. Therefore, we will optimize learning strategies and refine issues to enhance the model's performance.
3. Our model currently cannot automatically handle untrained generalization forms or new concepts beyond the meta-learning distribution, which greatly limits its ability to tackle entirely new problems. Therefore, whether the model can utilize real-world training experiences in all aspects to achieve human-like systematic generalization remains an open question.