In-Context Principle Learning with Categorized Mistakes

Jiani ZHONG   Xixi CHEN   Ying CAI   Yiqiao HUANG

Abstract

In-context learning (ICL) adapts Large Language Models (LLMs) to various tasks through few-shot prompting. While recent studies have explored learning from induced mistakes, systematic error categorization remains limited. This study innovates by integrating error typologies into ICL to refine the learning process. We use a dual approach: leveraging few-shot learning and semi-supervised BERT methodologies to classify errors in LLM-generated content. A subset of errors is manually categorized to train models, which then label new instances, validated by human review. These labeled datasets instruct LLMs, like GPT-3.5-turbo and GPT-4, to generate task-specific learning principles, tested on the MATH benchmark. Our research (1) improves error categorization granularity in ICL, (2) offers insights into semi-supervised learning for error analysis, and (3) advances understanding of LLMs' structured learning from mistakes, potentially developing more robust AI systems.

1 Introduction

In-context Learning (ICL), or few-shot learning, has advanced Large Language Models (LLMs) in natural language processing (NLP), enabling efficient task adaptation with minimal examples, reducing the need for extensive datasets (Radford et al., 2019, [9]; Brown et al., 2020 [12]). This allows LLMs to generate outputs by implicitly learning from a few examples (Wei et al., 2022b [12]).

Traditionally, ICL uses input-output pairs with correct answers (Min et al., 2022b [8]). Inspired by human learning processes (Edmondson, 1996 [3]; Chialvo and Bak, 1999 [2]; Edmondson, 1999 [2]), Zhang et al. (2024) [14] introduced Learning Principles (LEAP), encouraging LLMs to learn from mistakes by including incorrect answers. This method has shown promising results, outperforming Chain-of-Thought (CoT; Wei et al., 2022c [13]) in several settings.

However, LEAP's improvements are inconsistent, and CoT outperforms it in some scenarios. Drawing on human error learning principles, emphasizing analysis of reasoning behind mistakes (Metcalfe, 2017 [7]), we propose enhancing LEAP through error categorization. This enhancement progresses through several phases:

a) Few-Shot Learning for Categorization: We manually label a subset of errors and use these examples to train the LLM to categorize the remaining unlabeled data. This enables the LLM to apply learned principles to new errors, identifying similar patterns across the dataset, as illustrated in Figure 1. 1

b) Semi-Supervised Learning with Sentence-BERT: We use BERT (Bidirectional Encoder Representations from Transformers) introduced by Devlin et al. (2018) [1] to categorize text errors. BERT's transformer encoders, pre-trained on diverse language tasks, enhance performance in specific NLP tasks like sentiment analysis and question answering. For our project, we adapt BERT through a semi-supervised approach, using a subset of manually labeled error data. A detailed procedure of this phase is illustrated in Figure 2. 2

c) Manual Analysis of Categorization Quality: We manually analyze the quality of error categorization by sampling from both the few-shot learning and semi-supervised learning sets to determine which method is more accurate.

2 Background

In-context learning for question-answer (QA) tasks involves using a small set of task-specific examples, typically 3 to 8, to guide a pre-trained Large Language Model (LLM) (Liu et al., 2021) [5]. Let $P = (x_i, y_i)i = 1^k$ be concatenated to form a prompt $p = \langle x_1 \cdot y_1 \rangle \oplus \langle x_2 \cdot y_2 \rangle \oplus \ldots \oplus \langle x_k \cdot y_k \rangle$, where x_i is the question and y_i is the answer; $\cdot$ denotes concatenation of input with output, and $\oplus$ denotes sequential combination. Each new test question $x$test is appended to this prompt, resulting in $p \oplus \langle x_{\text{test}} \cdot$, which the LLM completes to generate the answer $\hat{y}_{\text{test}}$.

The answer y usually includes an explanation or reasoning (Wei et al., 2022c [13]). Each y_i contains a thought t_i and a solution a_i, denoted as $y_i = t_i \cdot a_i$, and the LLM's output $\hat{y}$test should have the structure $t$test $\cdot\, a_{\text{test}}$. We focus on this chain-of-thought setting, which has proven effective for reasoning tasks (Wang et al., 2022 [11]; Wei et al., 2022 [13]; Zhou et al., 2022 [15]; Wang et al, 2022 [10]; Zhang et al., 2024 [14]).

3 Methodology

In this study, our goal is to use the mistakes and their categorization labels to make the LLM learn principles that help the it not only against this specific mistake but against the broader category of syntax errors in future problems.

3.1 Dataset

We use the MATH dataset [4], which includes problems from seven categories: algebra, calculus, statistics, geometry, linear algebra, and number theory. We sampled 100 problems from each category at a difficulty level of 3. This dataset was chosen because it performed the least favorably in Zhang (2024)'s work [14]. The dataset's categorization by sub-fields suggests varying methods and thinking approaches, implying different reasons for mistakes. These can be categorized to generate principles to improve the model's ability. A difficulty level of 3 ensures problems are challenging enough to produce mistakes but not too difficult for the model.

3.2 Inducing the model to make mistakes

To enhance the model's learning, we leverage its ability to reflect on its mistakes. We use Zhang (2024)'s method [14], inducing mistakes through zero-shot learning with a non-zero temperature, which increases the likelihood of errors by generating answers with less confidence.

For each question x_i, we input it into the LLM and generate 15 responses $y_1, \ldots, y_{15}$. Using chain-of-thought prompting [13], we identify incorrect responses by comparing the final result a_j of the generated answers y_j (comprising the thought process t_j and final result a_j) to the ground truth solution s_i. This process forms the wrong answer set for each question i.

Mathematically, the wrong answer set which contains the question x_i and all its wrong answers can be expressed as:

$$\mathcal{W}_i = \{(x_i, a_j) \mid a_j \neq s_i, j = 1, \ldots, 15\}$$

This set $\mathcal{W}_i$ contains all the responses whose final answer does not match the ground truth solution s_i.

3.3 Benchmark Principle Generating

The wrong answer set $\mathcal{W}$ is fed to the LLM to generate task-specific "low-level" principles, denoted as P_{LOW}. Each principle P_{LOW_i} is added to its corresponding wrong answer set. We then use the question, wrong answers, and the principle as prompts, asking the LLM to generate answers for similar but unseen problems.

We aggregate the low-level principles to form more general "high-level" principles for each category. Using these high-level principles with the wrong answer set as prompts, we ask the LLM to generate answers for questions from the same category. These steps align with the LEAP method, serving as the benchmark for our approach. The procedure is illustrated in Figure 3. 3

3.4 Generating Principles From Categorized Mistakes

We advanced the LEAP algorithm by categorizing mistakes before generating explicit principles, inspired by human learning processes. Zhang's LEAP draws an analogy to human learning, where mistakes are not only recognized but categorized and used to find general improvements. For example, if someone incorrectly calculates "x - 1 = 2" as "x = 2 - 1," they identify it as an operational error, a common mistake in similar problems.

The quality of categorization is crucial for generating effective principles. To ensure robustness, we used two methods to generate error type labels for each mistake. First, we manually examined mistakes by sampling 20 questions from each category, labeling errors as "Calculation error in numerical value,"

”Operator error,” ”Extra or missing terms,” ”Ignoring specified conditions,” or ”Misunderstanding key concepts.” Errors were one-hot encoded, resulting in 98 labeled samples.

These manual labels were then used in two approaches: leveraging the LLM’s in-context learning to label unlabeled mistakes and using BERT for semi-supervised learning on the unlabeled samples.

3.4.1 Categorizing Mistakes Using LLM

We treat the labeling task as a QA problem, where mistakes, along with their corresponding questions and solutions, are provided. The answer is one of five categories: ”Calculation error,” ”Operator error,” ”Extra/missing terms,” ”Ignoring conditions,” and ”Misunderstanding key concepts.” We leverage the LLM’s in-context learning by prompting it with five examples from the same category (algebra, calculus, statistics, geometry, linear algebra, and number theory) and asking it to generate labels for the remaining examples in that category.

3.4.2 Categorizing Mistakes Using BERT

We leveraged the BERT model [1] for automated classification of error types, fine-tuning it with a mix of manually labeled and unlabeled datasets to improve accuracy. Configured to predict multiple output labels, the model was optimized using CUDA on a GPU.

The dataset included 98 manually labeled and 3,602 unlabeled examples. Unlabeled data generated pseudo-labels, iteratively augmenting the training dataset. The optimal pseudo-labeling threshold was selected during iterations. The labeled data were split 80-20 for training and validation. Training spanned 200 epochs with varying batch sizes, learning rates, and pseudo-labeling thresholds. We used the AdamW optimizer [6] and Binary Cross-Entropy with Logits Loss. Figure 4 4 shows loss and accuracy over epochs.

We used dynamic adaptation with pseudo-labeling, converting confident predictions on unlabeled data into new training samples. Performance was monitored with early stopping based on loss and accuracy metrics, saving the best model parameters. The optimized model was evaluated on a test set of 600 unseen data points.

After manual examination, we adopted the LLM’s results for further use. The in-context principle learning algorithm from categorized learning is summarized in Algorithm 1. 1

4 Results

We evaluated 700 questions from the level-3 MATH dataset. Initially, we applied the standard few-shot Chain-of-Thought (CoT) approach, with results noted as **Few-shot CoT**. Next, we applied LEAP using the same number of examples, with results noted as $\mathbf{LEAP_{Low\ Principle}}$ and $\mathbf{LEAP_{High\ Principle}}$. Finally, we applied categorized LEAP on these samples, with results noted as $\mathbf{LEAP_{Low\ Principle\ with\ Label}}$ and $\mathbf{LEAP_{High\ Principle\ with\ Label}}$.

We assessed the LLM’s accuracy on unseen examples for these methods using both GPT-3.5-turbo and GPT-4.0. The accuracy for each method and model is presented in Table 1 1. Notably, there is an improvement in performance when using categorized LEAP with both high-level and low-level feedback. Sample low-level and high-level principles learned by GPT-3.5-turbo for the algebra category are shown in Table 2 2 and Table 3, respectively 3.

Additionally, we observed that low-level principles outperform high-level principles, which contrasts with LEAP where the two methods differ little. This aligns intuitively with our hypothesis: generalizing the principles eliminates the categorization of errors, reducing our method to the original LEAP. However, we have only tested our method on a limited number of models and datasets, so this result needs to be further validated through additional trials.

5 Discussion and Novelty

Our approach enhances GPT’s performance and reasoning in solving mathematical problems by aligning its thinking process with human learning methods. We introduce the summarization of principles, creating specific low-level principles for each question and general high-level principles. While low-level principles are detailed, high-level principles, though useful for generalization, may be too broad. The optimal method depends on the task’s complexity and question diversity, as high-level principles might not always outperform low-level ones due to their generality.

We use regular expressions to ensure consistency between GPT's responses and ground-truth answers, and employ BERT for efficient, automated error classification. Despite these advancements, the effectiveness of high-level versus low-level principles varies with task complexity and question diversity, indicating no universally superior method.

Future work could involve integrating LEAP with multimodal systems for enhanced problem-solving across various data types, and adapting LEAP for educational applications in STEM fields. Further refining AI reasoning algorithms will enable deeper understanding and handling of complex logical tasks, enhancing AI's robustness and versatility in solving diverse problems.

6 Conclusion

Our approach demonstrates significant advancements in enhancing GPT's performance and reasoning abilities in solving mathematical problems. LEAP has shown to outperform Chain of Thought benchmarks by improving mathematical problem-solving capabilities through principles that emulate human cognitive processes, thereby boosting the model's intelligence and reasoning skills. These improvements suggest that AI can potentially handle complex, logic-based tasks, approaching human-like problem-solving skills.

Future work will focus on integrating LEAP with multimodal systems to enhance problem-solving across various data types, such as text, images, and sounds. Additionally, there is potential to adapt LEAP for educational applications, particularly in providing adaptive learning support and feedback in STEM fields. Further refining AI reasoning algorithms will enable a deeper understanding and handling of complex logical and abstract reasoning tasks, enhancing the robustness and versatility of AI systems in solving diverse problems.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[2] Amy Edmondson. Psychological safety and learning behavior in work teams. *Administrative science quarterly*, 44(2):350–383, 1999.

[3] Amy C Edmondson. Learning from mistakes is easier said than done: Group and organizational influences on the detection and correction of human error. *The journal of applied behavioral science*, 32(1):5–28, 1996.

[4] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021.

[5] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.

[7] Janet Metcalfe. Learning from errors. *Annual review of psychology*, 68:465–489, 2017.

[8] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. *arXiv preprint arXiv:2110.15943*, 2021.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[10] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*, 2022.

[11] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[12] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

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

[14] Tianjun Zhang, Aman Madaan, Luyu Gao, Steven Zheng, Swaroop Mishra, Yiming Yang, Niket Tandon, and Uri Alon. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*, 2024.

[15] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

A Figures, Tables, and Algorithm

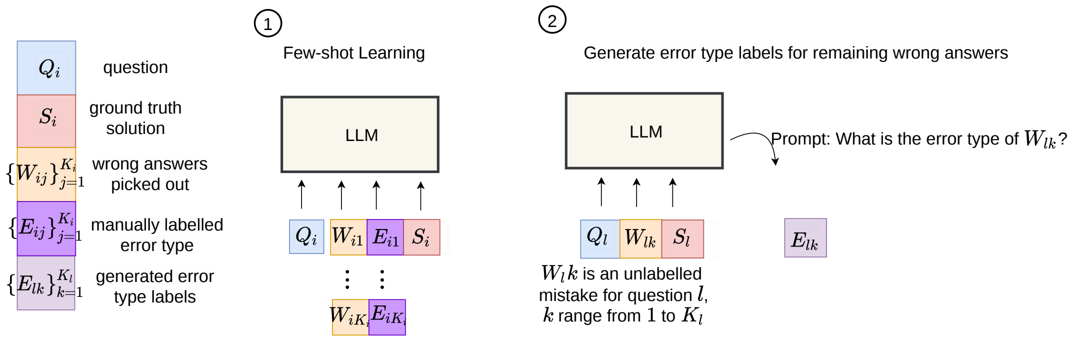
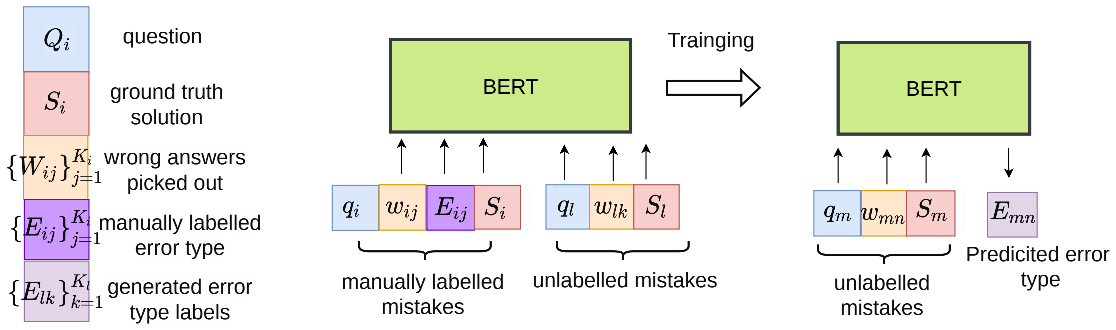

Q_i question
S_i ground truth solution
$\{W_{ij}\}_{j=1}^{K_i}$ wrong answers picked out
$\{E_{ij}\}_{j=1}^{K_i}$ manually labelled error type
$\{E_{lk}\}_{k=1}^{K_l}$ generated error type labels
1 Few-shot Learning
LLM
Q_i W_{i1} E_{i1} S_i
W_{iK_i} E_{iK_i}
2 Generate error type labels for remaining wrong answers
LLM
Prompt: What is the error type of W_{lk}?
Q_l W_{lk} S_l
E_{lk}
W_lk is an unlabelled mistake for question l, k range from 1 to K_l

Figure 1: The procedure of using LLM to categorize the mistakes.

Few-shot Learning: Feed five examples to LLM, each consists of the question, the wrong answers, the manually labelled error type of each mistake, and the ground truth solution. Along in the prompt ask the LLM to predict the label of the error type for an unseen mistake, providing its corresponding question and the solution.

	GPT-3.5-turbo	**GPT-4**
Few-shot CoT	42.54	60.94
LEAP$_{\text{Low Principle}}$	42.76	62.70
LEAP$_{\text{High Principle}}$	43.51	61.54
LEAP$_{\text{Low Principle with Label}}$	**43.92**	**63.04**
LEAP$_{\text{High Principle with Label}}$	42.37	62.88

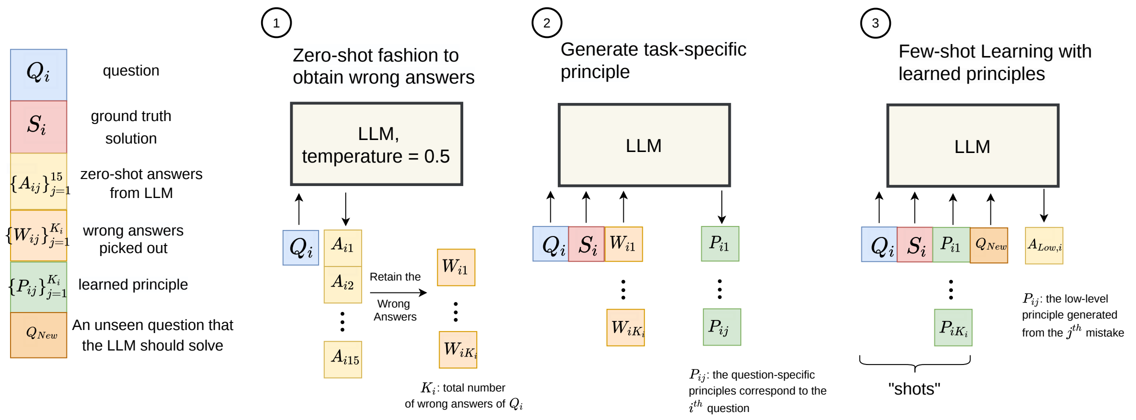
Table 1: Accuracy results on 700 questions from the level-3 MATH dataset for GPT-3.5-turbo and GPT-4.The best approach for each base LLM in each dataset is in **bold**



Q_i question
S_i ground truth solution
$\{W_{ij}\}_{j=1}^{K_i}$ wrong answers picked out
$\{E_{ij}\}_{j=1}^{K_i}$ manually labelled error type
$\{E_{lk}\}_{k=1}^{K_l}$ generated error type labels
BERT
q_i w_{ij} E_{ij} S_i
q_l w_{lk} S_l
manually labelled mistakes
unlabelled mistakes
Trainging
BERT
q_m w_{mn} S_m
E_{mn}
unlabelled mistakes
Predicited error type

Figure 2: Error Categorization Using BERT

The subset of mistakes, accompanied by their manually labeled error categorization, along with the corresponding questions and solutions, constitutes the cornerstone of our semi-supervised learning approach. In addition to this labeled subset, we also incorporate a portion of unlabeled data into the training process. Following validation, we anticipate that BERT will leverage its learned functions to annotate the remaining unlabeled data.



Q_i question
S_i ground truth solution
$\{A_{ij}\}_{j=1}^{15}$ zero-shot answers from LLM
$\{W_{ij}\}_{j=1}^{K_i}$ wrong answers picked out
$\{P_{ij}\}_{j=1}^{K_i}$ learned principle
Q_{New} An unseen question that the LLM should solve
1
Zero-shot fashion to obtain wrong answers
LLM, temperature = 0.5
Q_i
A_{i1}
A_{i2}
A_{i15}
Retain the Wrong Answers
W_{i1}
W_{iK_i}
K_i: total number of wrong answers of Q_i
2
Generate task-specific principle
LLM
Q_i S_i W_{i1}
W_{iK_i}
P_{i1}
P_{ij}
P_{ij}: the question-specific principles correspond to the i^{th} question
3
Few-shot Learning with learned principles
LLM
Q_i S_i P_{i1} Q_{New}
P_{iK_i}
$A_{Low,i}$
P_{ij}: the low-level principle generated from the j^{th} mistake
"shots"

Figure 3: Procedure of the benchmark

We initially set the LLM's temperature to 0.5 and use a zero-shot approach to generate 15 different answers for each question. Incorrect answers are identified and selected. These incorrect answers, along with the corresponding question and ground-truth solution, are fed back into the LLM with the temperature set to zero to generate question-specific "low-level" principles. These low-level principles are then used to derive more general "high-level" principles for each category (algebra, calculus, statistics, geometry, linear algebra, and number theory). Finally, the LLM is prompted with the original question, the solution, the principles (both low-level and high-level), and a new, unseen question to assess if the LLM's answer accuracy has improved.

Algorithm 1 Categorized LEAP Algorithm

Require: Few-shot examples $\mathcal{P} = \{(x_i, y_i)\}_{i=1}^{k}$, a pretrained LLM, number of outputs per input n, high-temperature setting T, question categories:

- $\mathcal{Q} = \{$algebra, calculus, statistics, geometry, linear algebra, number theory$\}$

error type categories:

- $\mathcal{E} = \left\{ \begin{array}{l} \text{calculation error in numerical value,} \\ \text{operator error,} \\ \text{extra or missing terms in the equation,} \\ \text{ignoring the specified conditions of the problem,} \\ \text{misunderstanding the key concept in the problem} \end{array} \right\}$

```
 1: for each input-output pair $(x_i, y_i)$ in $\mathcal{P}$ do
 2:     $S_i \leftarrow$ ZeroShotCoT(LLM, $x_i, n, T$) {Generate solutions using zero-shot chain-of-thought prompting}
 3:     $\mathcal{M}_i \leftarrow \{(x_i, y_i, \hat{y}_i^j) \in S_i : \hat{a}_i^j \neq a_i\}$ {Identify incorrect solutions for each question}
 4: end for
 5: for each 20 random samples from the question category $\mathcal{Q}$ do
 6:     $L_{manual} \leftarrow$ A manual label from the error type category $\mathcal{E}$
 7: end for
 8: for each remaining samples do
 9:     $L_{LLM} \leftarrow$ Generate Error type labels by LLM from the category $\mathcal{E}$
10: end for
11: for each $\mathcal{M}_i$ in $\mathcal{M}$ do
12:     for each $(x_i, y_i, \hat{y}_i^j)$ in $\mathcal{M}_i$ do
13:         $(x_i, y_i, \hat{y}_i^j, l_i^j) \leftarrow$ Assign error type label for each mistake
14:         $\mathcal{L}_{\text{LOW-LEVEL},i} \leftarrow$ Generate Low-Level Principles(LLM, $x_i, \hat{y}_i^j, y_i$) {Generate principles for each mistake}
15:     end for
16: end for
17: $\mathcal{L}_{\text{LOW-LEVEL}} \leftarrow \bigcup_{i=1}^{k} \mathcal{L}_{\text{LOW-LEVEL},i}$ {Aggregate low-level principles}
18: $\mathcal{L}_{\text{HIGH-LEVEL}} \leftarrow$ Generate High-Level Principles(LLM, $\mathcal{L}_{\text{LOW-LEVEL}}$) {Generate high-level principles}
19: $p_{\text{LOW-LEVEL}} \leftarrow$ Concatenate($\mathcal{L}_{\text{LOW-LEVEL}}, \mathcal{P}$) {Create enhanced prompt with low-level principles}
20: $p_{\text{HIGH-LEVEL}} \leftarrow$ Concatenate($\mathcal{L}_{\text{HIGH-LEVEL}}, \mathcal{P}$) {Create enhanced prompt with high-level principles}

21: return $p_{\text{LOW-LEVEL}}, p_{\text{HIGH-LEVEL}}$
```

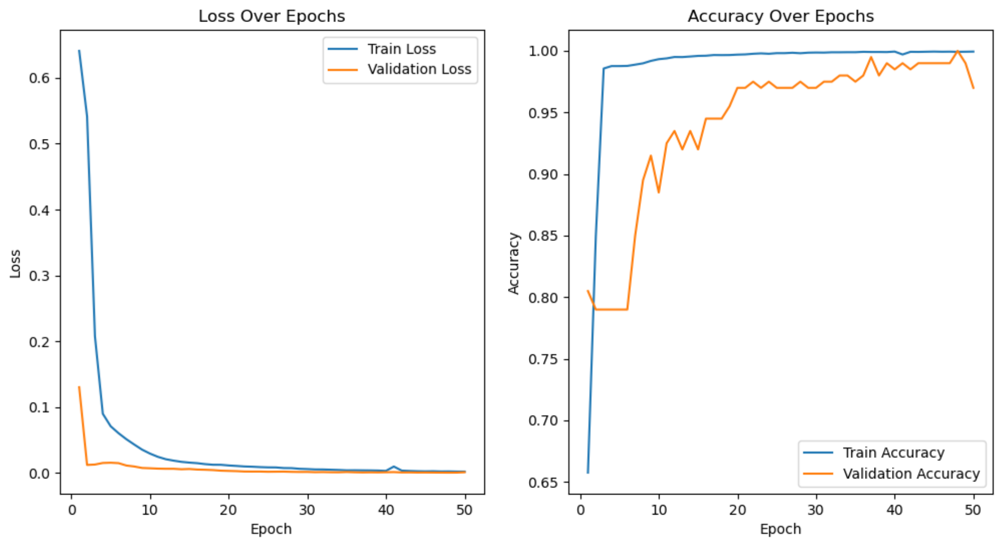


Figure 4: Loss and accuracy of BERT training

The left side of Figure 4 displays the loss over epochs, and the right side shows the accuracy over epochs.

Low-level principles learned for algebra using GPT-3.5-turbo
1. Always verify that the given points are opposite vertices of the square before applying any formulas. 2. Using the diagonal length instead of the side length to calculate the area. 3. The quadratic formula was incorrectly applied in some solutions, leading to incorrect roots for the equation. 4. Ensuring consistency in interpreting the intervals where the expression is positive or negative and in determining the valid interval for satisfying the inequality. 5. The distance between two points is the length of the line segment connecting them.

Table 2: low-level principles learned by GPT3.5-Turbo for algebra category

High-level principles learned for algebra using GPT-3.5-turbo
1. Before attempting to solve counting and probability problems, carefully analyze the problem statement, identify all relevant constraints, and apply systematic and logical approaches. 2. Always check your work and make sure you have the correct signs and values for the coefficients and constants in the quadratic equation. 3. For algebra problems, use logical and systematic methods to manipulate equations and expressions, while following established rules and principles, to solve problems and find solutions.

Table 3: high-level principles learned by GPT3.5-Turbo for algebra category