metaTextGrad: Learning to learn with language models as optimizers

Guowei Xu
Tsinghua University
xgw23@mails.tsinghua.edu.cn

Mert Yuksekgonul
Stanford University
merty@stanford.edu

Carlos Guestrin
Stanford University
guestrin@stanford.edu

James Zou
Stanford University
jamesz@stanford.edu

Abstract

Large language models (LLMs) are increasingly used in learning algorithms, evaluations, and optimization tasks. Recent studies have shown that incorporating self-criticism into LLMs can significantly enhance model performance, with frameworks such as TextGrad [1] illustrating this approach by iteratively refining model outputs through prompting. However, these frameworks often require extensive hand-crafting and are sensitive to instruction wording. To mitigate these challenges, we propose metaTextGrad, a meta-learning approach for LLM-based optimizers, focusing on learning loss functions and templates for inference-time optimization. Our method significantly improves performance across multiple benchmarks, achieving 5-27% gains on question-answering tasks. These results demonstrate the potential of meta-learning to enhance LLM-based systems, reducing manual tuning and improving generalizability.

1 Introduction

Large language models (LLMs) are increasingly used in learning algorithms, optimization, and evaluation tasks [1, 2, 3, 4, 5]. However, algorithms that incorporate LLMs often face significant challenges. Firstly, many of these algorithms are still hand-crafted to a considerable extent, requiring substantial human expertise and effort to design and implement effectively. This manual approach can be time-consuming and may not always result in optimal performance across diverse tasks and domains. Second, LLMs are notably sensitive to the specific wording and structure of their instructions [6], making it tedious to improve them effectively to be used in learning algorithms.

In this study, we focus on inference-time optimization problems. Specifically, instead of relying solely on the model's output during inference, we iteratively optimize it to progressively enhance the model's performance on the task. There are two key challenges associated with inference-time optimization: the first is identifying an appropriate loss function to ensure that the optimization is moving in the correct direction; the second is determining a good initialization to facilitate the optimization process. Traditionally, these design choices have been made through complex manual efforts, and our work aims to develop an automated approach to address both of these issues.

Given that learning good initializations [7, 8] or loss functions [9, 10] is a well-explored area in meta-learning, we draw inspiration from two families of meta-learning algorithms. Specifically, we propose metaTextGrad, which consists of two approaches. First, we optimize LLM-based loss functions to better align the loss with the task of interest, enabling the loss function to provide more effective guidance for optimization in the correct direction. Second, we explore learning good

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

initializations, akin to methods such as MAML [7] and Reptile [8], which we refer to as inference templates. These templates can include solution strategies for reasoning problems, thereby improving the model's adaptability and performance during inference.

We find that `metaTextGrad` significantly improve the performance of LLM-based optimizers across multiple benchmarks. Our results demonstrate that learned loss functions and inference templates can lead to more accurate and efficient optimization, particularly in complex reasoning tasks. Specifically, `metaTextGrad` achieves improvements between $5 - 27\%$ on question answering tasks, such as BBH [11], MMLU [12], and GPQA [13].

To the best of our knowledge, `metaTextGrad` is **the first approach to employ meta-learning for inference-time optimization** in LLMs. Our findings suggest the benefits of meta learning for LLM-based optimizers. As LLMs are utilized more in evaluation and optimization tasks, we suggest learning to learn will be a highly important part of future optimizers. This approach not only enhances the adaptability of LLM-based systems but also reduces the need for manual tuning, potentially leading to more robust and generalizable AI systems.

2 `metaTextGrad`: Learning to (learn at inference time) in natural language

We are interested in learning to learn at inference time, using LLMs in the loop of learning algorithms. Let $\phi(x, y, \hat{y}) \in \mathbb{R}$ be some success metric for a task, where x is the input for the task, y is the ground truth label (if it exists), and $\hat{y}$ is a prediction. For instance, ϕ can be accuracy for question answering.

In LLM-based learning algorithms, we often use LLMs both as *evaluators* or *critics* to provide the signal for optimization, and *optimizers* update the states of variables. An LLM denoted by f_{LLM} is simply a probability distribution, where we can sample the next token iteratively through: $x_t \sim f_{LLM}(\cdot|x_{1:t-1})$. Sometimes we'll abuse the notation and use $x_{t:T} = \text{LLM}(x_{1:t-1})$ to denote sampling multiple tokens, which will be clear from the context.

An LLM evaluator is denoted by $\text{LLM}(x, y, \hat{y}; \theta_{\mathcal{L}})$, where the LLM uses the task input x, the label y, the prediction $\hat{y}$, and a loss parameter $\theta_{\mathcal{L}}$ to produce an evaluation, which can be in natural language.

When running optimization at inference-time, we do not have access to the ground truth label. Thus, an optimizer using LLM evaluators often proceeds through the following iterative process:

$$\text{Evaluation} = \text{LLM}(x, \hat{\mathbf{y}}_{t-1}; \theta_{\mathcal{L}}), \tag{1}$$

$$\hat{\mathbf{y}}_t = \text{TGD.step}(\hat{\mathbf{y}}_{t-1}, \frac{\partial \text{Evaluation}}{\partial \hat{\mathbf{y}}_{t-1}}), \tag{2}$$

where TGD.step is a textual gradient descent optimizer such as TextGrad [1] (check appendix A for details), and $\hat{y}_0$ is an initial response generated using an inference template $y_{\mathcal{T}}$. Table 1 displays a minimal example of this process.

Our goal is to improve the loss prompt $\theta_{\mathcal{L}}$ and inference template $y_{\mathcal{T}}$ using ideas from meta learning [7, 8]. In particular, we want to optimize the average performance of the LLM on the dataset:

$$\theta^*_{\mathcal{L}} = \arg\max_{\theta_{\mathcal{L}}} \mathbb{E}_{(x,y)\sim D_{\text{train}}} \left[\phi(x, y, \hat{y}_T(x, \theta_{\mathcal{L}}, y_{\mathcal{T}}))\right] \tag{3}$$

$$y^*_{\mathcal{T}} = \arg\max_{y_{\mathcal{T}}} \mathbb{E}_{(x,y)\sim D_{\text{train}}} \left[\phi(x, y, \hat{y}_T(x, \theta_{\mathcal{L}}, y_{\mathcal{T}}))\right], \tag{4}$$

where D_{train} is the training dataset, and we obtain $\hat{y}_T$ through the optimization process in Eq 1.

The meta-learning framework contains two main components: an ***outer loop*** that optimizes the learning target (either loss prompt or inference template), and an ***inner loop*** (described in equations 1 and 2) that uses the current loss prompt and inference template to generate optimized solutions. After each inner loop, we evaluate the generated results. If the evaluated accuracy is not better than the best iteration before, we revert the learning target and optimize again. If the evaluated accuracy is better, we proceed to a new outer loop.

In what follows, we describe three meta learning approaches to improve optimization at inference time.

Table 1: A minimal example to illustrate the inference time optimization process

Variable	Value
x (question)	If it takes 1 hour to dry 25 shirts under the sun, how long will it take to dry 30 shirts under the sun?
y (standard answer, not accessible)	1 hour
$y_{\mathcal{T}}$ (inference template)	Let's break this down step by step. First,
$\theta_{\mathcal{L}}$ (loss prompt)	Critically go through reasoning steps and see if the answer is correct.
$\hat{\mathbf{y}}_0$ (generated response)	Let's break this down step by step... it will take 1.2 hours.
Evaluation	Your reasoning contains a critical flaw that drying time is directly proportional to...
$\hat{\mathbf{y}}_1$ (improved response)	Let's break this down step by step... it will take 1 hour.

2.1 Optimizing a loss function

The goal of this approach is to learn a good loss prompt for inference-time optimization, specifically targeting situations where we optimize solutions at test time without access to ground-truth answers. Typically, these loss prompts are manually crafted [14, 5, 1]. Designing such prompts is time-consuming, and each new dataset often requires a new, custom loss prompts. Moreover, human-designed loss prompts tend to be relatively simple, whereas more expressive ones could better identify errors or refine approaches.

To iteratively optimize the loss prompt $\theta_{\mathcal{L}}$, we use the manually designed loss prompt described in TextGrad [1] as the initial prompt, which is listed in Appendix C.

In each inner loop, the algorithm generates responses using the current loss prompt θ_t. Then the outer loop optimizes the loss prompt $\theta_{\mathcal{L}}$ based on responses and the meta learning loss prompt θ_{meta}:

$$\text{Meta Evaluation} = \text{LLM}(x, y, \hat{y}, \theta_{\mathcal{L}}; \theta_{meta}) \tag{5}$$

$$\theta_{\mathcal{L}} = \text{TGD.step}(\theta_{\mathcal{L}}, \frac{\partial \text{Meta Evaluation}}{\partial \theta_{\mathcal{L}}}), \tag{6}$$

To ensure the generality of the outer loop, we keep the following meta loss prompt below fixed across all experiments:

Meta Loss Prompt θ_{meta}

LLM("You are a language model that improves prompts for answering reasoning questions.
You are given some wrong answers generated using the prompt and the standard answers:
Question 1: {question}
Generated Wrong Answer 1: {response}
Standard Answer 1: {answer}
...
(more Q&A pairs)
...
Prompts for evaluation of answers of a reasoning question that must be improved:
{**loss prompt**}
Provide concise feedback on how to improve the prompt so it better addresses reasoning questions. Your suggestions should focus on refining key points related to answering reasoning questions. Ensure your suggestions are applicable to these types of reasoning questions as general aspossible, not just to the specific examples provided.")

2.2 Optimizing an inference template

In this section, we aim to design meta learning algorithms to learn a good inference template $y_{\mathcal{T}}$ for the inner loop. In our settings, an LLM uses the task input x, the prediction $\hat{y}$, and a loss prompt $\theta_{\mathcal{L}}$ to iteratively learn a good $y_{\mathcal{T}}$ that can help solve problems more accurately. We use a simple initial inference template to start with: *"Let's break this down step by step. First, ... "* [15]

The algorithms generate responses using the current inference template $y_{\mathcal{T}}$ in each inner loop. We present two methods to update the outer loop:

MAML-style optimization with textual gradients. Inspired by MAML algorithm [7], the outer loop optimizes the inference template $y_{\mathcal{T}}$ based on the gradients of the generated responses in the inner loop, which are directly obtained from the last gradient from the optimizer in the inner loop:

$$y_{\mathcal{T}} = \text{TGD.step}(y_{\mathcal{T}}, \frac{\partial \text{Evaluation}}{\partial \hat{y}}) \tag{7}$$

Reptile-style optimization with a meta loss. Inspired by the Reptile algorithm [8], the outer loop optimizes $y_{\mathcal{T}}$ based on these generated responses and the Reptile loss prompt $\theta_{\mathcal{R}}$:

$$\text{Meta Evaluation} = \text{LLM}(x, y, \hat{y}, y_{\mathcal{T}}; \theta_{\mathcal{R}}) \tag{8}$$

$$y_{\mathcal{T}} = \text{TGD.step}(y_{\mathcal{T}}, \frac{\partial \text{Meta Evaluation}}{\partial y_{\mathcal{T}}}) \tag{9}$$

To clarify how the Reptile loss prompt is applied to improve the inference template $y_{\mathcal{T}}$, we present the function $\theta_{\mathcal{R}}$ used in our experiments, which remains consistent across different benchmarks:

Reptile Loss Prompt $\theta_{\mathcal{R}}$

LLM("You are a language model that improves inference templates for answering reasoning questions. Here are some wrong responses:
Question 1: {question}
Generated Wrong Answer 1: {response}
Standard Answer 1: {answer}
...
(more Q&A pairs)
...
Here is the inference template to improve:
{**inference template**}
Your task is to improve the inference template (the first few sentences of the response) so that it can generate better responses. Please keep the inference template concise and general, and end it with a colon so that it can be used as a prefix for generating responses.")

3 Experiments

In this section, we introduce our experiments conducted with GPT-4o-mini on three representative and challenging benchmarks: BBH Word Sorting [16], MMLU Machine Learning [12], and GPQA Diamond [17]. For the BBH Word Sorting task, we used the same train/val/test splits as TextGrad [1]. For the other two benchmarks, since there are no default training splits, we split the data with a train-to-test ratio of 1:3. Due to the non-determinism in the APIs serving the LLMs [18], the test accuracy is averaged over 6 random seeds.

Baselines. We used Zero-shot CoT [15], COPRO [4], MIPROv2 [19], and TextGrad's Prompt Optimization and Solution Optimization algorithms [1] as baselines. Zero-shot CoT leverages chain-of-thought reasoning without task-specific training, while COPRO, MIPRO and TextGrad refine responses based on different methods. To facilitate reproducibility, we have included the learned loss prompts and inference templates in Appendix B.

Table 2: Test Accuracy (%) of GPT-4o-mini on Different Benchmarks

Method	Word Sorting	MMLU Machine Learning	GPQA
Zero-Shot CoT	45.7	74.4	38.6
Prompt optimizers			
COPRO	55.0	73.8	34.2
MIPROv2	57.0	75.0	42.3
Prompt Optimization (TextGrad)	58.3	76.8	40.0
Inference time optimizers			
Solution Optimization (TextGrad)	60.0	76.6	38.4
Meta-learned inference time optimizers			
`metaTextGrad` (Learned Loss)	**73.2**	75.6	39.1
`metaTextGrad` (MAML-style)	66.0	**78.0**	**43.0**
`metaTextGrad` (Reptile-style)	65.8	77.4	40.9

BBH Word Sorting. The BBH Word Sorting task [16] is a challenging benchmark that requires accurate reasoning about word order. Our method, particularly `metaTextGrad` (Learned Loss) , outperformes all baselines with a test accuracy of 73.2%. This result demonstrates the effectiveness of our learned loss function in guiding the model toward better solutions by penalizing errors in the sorting process and refining word ordering criteria, such as those outlined in the prompt in Appendix B (e.g., "clearly explain the rules for sorting words alphabetically" and "list the original words and demonstrate the sorting process step-by-step").

MMLU Machine Learning. The MMLU Machine Learning benchmark [12] evaluates the model's ability to answer questions about machine learning concepts, which requires a deep understanding of the subject matter. `metaTextGrad` (MAML-style) achieves the best performance with an accuracy of 78.0%, outperforming other methods and demonstrating the strength of our meta-learned inference templates for this domain.

GPQA Diamond. The GPQA Diamond benchmark [17] tests the model's ability to solve graduate-level, Google-proof questions, a task that demands advanced reasoning capabilities. `metaTextGrad` (MAML-style) achieves the highest accuracy at 43.0%, proving that inference-time optimization through meta-learning can significantly improve the model's ability to tackle highly complex reasoning tasks.

All experimental results are summarized in Table 2. These results demonstrate the effectiveness of `metaTextGrad`, highlighting the great potential of meta-learning techniques in LLM inference-time optimization.

4 Conclusion

In this paper, we propose `metaTextGrad`, a meta-learning-based approach for optimizing large language models (LLMs) during inference. Our method tackles two key challenges in inference-time optimization: learning effective loss functions and good initializations. By leveraging meta-learning techniques, `metaTextGrad` significantly enhances performance across multiple benchmarks, demonstrating the robustness and generalizability of our approach.

Looking ahead, there are several promising directions for future work. One avenue is to explore whether jointly optimizing loss functions and initializations, rather than treating them separately, could lead to even better results. Another direction is to investigate whether it is possible to develop a unified loss function or initialization that can generalize across different tasks, rather than being tailored to specific domains. We firmly believe that meta-learning offers a powerful approach for future advances in LLM-based inference-time optimization, with the potential to deliver more adaptable and universally applicable solutions.

References

[1] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[2] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

[3] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

[4] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024.

[5] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Steffi Chern, Ethan Chern, Graham Neubig, and Pengfei Liu. Can large language models be trusted for evaluation? scalable meta-evaluation of llms as evaluators via agent debate. *arXiv preprint arXiv:2401.16788*, 2024.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[8] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms, 2018.

[9] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta learning via learned loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4161–4168. IEEE, 2021.

[10] Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Bautista Martin, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In *International conference on machine learning*, pages 2891–2900. PMLR, 2019.

[11] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, Toronto, Canada, July 2023. Association for Computational Linguistics.

[12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

[13] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[14] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] 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.

[16] Aarohi Srivastava, Abhinav Rastogi, and Abhishek Rao et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[17] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[18] 152334H. Non-determinism in gpt-4 is caused by sparse moe. `https://152334h.github.io/blog/non-determinism-in-gpt-4/`, 8 2023.

[19] Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs, 2024.

[20] Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, et al. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*, 2023.

[21] Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models, 2024.

[22] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online, November 2020. Association for Computational Linguistics.

[23] Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web conference 2022*, pages 2778–2788, 2022.

[24] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

[25] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*, 2023.

[26] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore, December 2023. Association for Computational Linguistics.

A Implementation Details

TextGrad [1] is a robust framework that enables automatic "differentiation" using text. It leverages the concept of backpropagation through textual feedback provided by large language models. In our experiments, we utilized the TextGrad engine to perform backpropagation and textual gradient descent. For readers unfamiliar with the TextGrad framework, we provide a few examples to demonstrate how it functions.

A.1 How do we perform backpropagation?

After evaluating the loss using the loss prompt, we also need to determine how to improve each relevant variable. To achieve this, TextGrad implements a "backpropagation" method, which provides suggestions (textual gradients) for the variables. This process recursively computes the gradients for each variable.

For instance, assuming $\mathcal{L} = \text{LLM}(y), y = \text{LLM}(x)$, the value of $\frac{\partial \mathcal{L}}{\partial x}$ can be computed using:

Example implementation for the gradient operator [1]

$$\frac{\partial \mathcal{L}}{\partial y} \triangleq \text{LLM}(\texttt{Below are the criticisms: } \{\mathcal{L}\} \quad \texttt{Explain how to improve \{y\}.}) \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial x} = \nabla_{\text{LLM}}(x, y, \frac{\partial \mathcal{L}}{\partial y}) \triangleq \texttt{"Here is a conversation with an LLM: \{x|y\}."} \tag{11}$$
$$+$$
$$\text{LLM}(\texttt{Here is a conversation with an LLM: \{x|y\}. Below are the criticisms on \{y\}:} \left\{\frac{\partial \mathcal{L}}{\partial y}\right\} \texttt{Explain how to improve \{x\}.})$$

A.2 How does TGD.step work?

Similar to traditional gradient descent, the goal of textual gradient descent is to refine a variable based on feedback (textual gradients) received for that variable.

Let's assume x is the variable we want to improve, and $\frac{\partial \mathcal{L}}{\partial x}$ represents the textual gradient (feedback) we obtained for x during the backward pass. Below is an example of one iteration of Textual Gradient Descent (TGD):

Example implementation of one TGD iteration [1]

$$\text{TGD.step}(x, \frac{\partial \mathcal{L}}{\partial x}) \triangleq \text{LLM}(\texttt{Below are the criticisms on \{x\}:} \left\{\frac{\partial \mathcal{L}}{\partial x}\right\} \texttt{Produce a new variable based on the criticisms.}) \tag{12}$$

B Learned Loss Prompt / Inference Template

To facilitate the reproducibility of our work, we present the Loss Prompts and Response Templates learned through three different meta-learning methods.

B.1 BBH Word Sorting Benchmark

The following results pertain to the BBH Word Sorting Benchmark:

COPRO

Analyze the provided ‘question‘ with careful attention to its context, nuances, and underlying assumptions. Construct a well-reasoned, clear, and concise ‘answer‘ that not only directly responds to the query but also anticipates potential follow-up questio ns or concerns. Aim to enrich your response with relevant examples, analogies, or explanations that enhance understanding, ensuring that th e tone is engaging and appropriate for the audience’s level of familiarity with the topic.
—
Follow the following format.
Question: The question to answer.
Reasoning: Let’s think step by step in order to $produce the answer. We ...
Here’s a detailed answer that addresses your question and provides further insights: The answer to the question. The last line of your resp onse should be of the following format: ’Answer:’
—

MIPROv2

Given the fields ‘question‘, produce the fields ‘answer‘.
—
Follow the following format.
Question: The question to answer.
Reasoning: Let’s think step by step in order to $produce the answer. We ...
Answer: The answer to the question. The last line of your response should be of the following format: ’Answer:’
—

Meta Learning via Learned Loss Prompt

Prompt for Evaluating Word Sorting Answers:
Understanding the significance of sorting in reasoning tasks is crucial. This prompt guides you through evaluating the provided answer to a word sorting problem. Follow these steps to enhance clarity and understanding of the sorting process:
1. **Sorting Criteria**: Clearly explain the rules for sorting words alphabetically. What principles should guide the process?
2. **Verification**: List the original words and demonstrate the sorting process step-by-step. Identify any discrepancies between the original and sorted lists.
3. **Identify Errors**: Point out mistakes in the sorting order. Reflect on the reasoning behind these errors and explain why they are incorrect. Provide the correct order with examples.
4. **Correct Order**: Present the accurate alphabetical order of the words clearly.
5. **Learning Application**: Summarize what you learned about sorting and suggest practical ways to reinforce this knowledge, such as sorting different word sets or engaging in related activities.
6. **Real-World Implications**: Discuss how incorrect sorting can impact data organization in fields like databases or libraries. Provide specific examples to illustrate your points.
7. **Encourage Engagement**: Consider discussing your findings with peers or engaging in group evaluations to enhance learning and understanding of sorting principles.

Meta Learning via Learned Loss Prompt (Cont.)

By following these steps, you will not only evaluate the sorting process effectively but also contribute to a deeper understanding of its significance in various contexts.

Meta Learning via Learned Response Template (MAML-style)

We will sort the words alphabetically. Here are the words in the list:

Meta Learning via Learned Response Template (Reptile-style)

To sort the provided list of words alphabetically, we will compare the first letters of each word. If the first letters are the same, we will then compare the subsequent letters until we find a difference. Here's the list of words to be sorted:

B.2 MMLU Machine Learning Benchmark

Below are the results obtained from the MMLU Machine Learning Benchmark:

COPRO

Analyze the provided question thoroughly and generate a comprehensive answer that addresses all aspects of the inquiry. Ensure that your re sponse is clear, concise, and informative, providing relevant details and examples where applicable. Aim to enhance the understanding of th e topic for the reader.
—
Follow the following format.
Question: The question to answer.
Reasoning: Let's think step by step in order to $produce the answer. We ...
Here is a detailed answer to the question: The answer to the question. The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of ABCD.
—

MIPROv2

Given the multiple-choice question provided in the 'question' field, please analyze the statements step by step to determine the correct an answer. Your response should include a clear rationale explaining your reasoning process, followed by the final answer formatted as 'Answer: $LETTER', where LETTER is one of A, B, C, or D.
—
Follow the following format.
Question: The question to answer.
Reasoning: Let's think step by step in order to $produce the answer. We ...
Answer: The answer to the question. The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) wh ere LETTER is one of ABCD.
—
Question: Answer the following multiple choice question. The last line of your response should be of the following format: 'Answer: $LETTER$ ' (without quotes) where LETTER is one of ABCD. Think step by step before answering. Statement 1| Linear regression estimator has the small est variance among all unbiased estimators. Statement 2| The coefficients α assigned to the classifiers assembled by AdaBoost are always no n-negative. A) True, True B) False, False C) True, False D) False, True
Reasoning: Let's think step by step in order to evaluate the truth of the statements.

MIPROv2 (cont.)

For Statement 1, it is known that while linear regres sion is a popular estimator, it does not necessarily have the smallest variance among all unbiased estimators; this is a property of the Ga uss-Markov theorem under certain conditions. Therefore, Statement 1 is False. For Statement 2, in AdaBoost, the coefficients assigned to th e classifiers are indeed always non-negative, as they represent the weight of each classifier in the ensemble. Thus, Statement 2 is True.
Answer: D
—
(6 more Q&A pairs)
—

Meta Learning via Learned Loss Prompt

Revised Prompt for Evaluating Answers to Reasoning Questions:
Context: Below is a multiple-choice question from the field of [insert subject area, e.g., statistics, machine learning, etc.], along with an answer. Your task is to evaluate the answer based on the following criteria:
1. **Does the answer logically follow from the statements?** 2. **Are the statements relevant to the question being asked?** 3. **Do the statements align with established knowledge in the field?**
Instructions: - Identify specific elements in the statements that support or contradict the chosen answer. - Summarize the main points of each statement and explore their implications and interrelations. - Reflect on any prior knowledge that influenced your evaluation.
Justification: Provide a brief justification for your answer in the format: 'Answer: $LETTER' (without quotes), where LETTER is one of ABCD. Use the following template: "I chose answer X because [reasoning based on statements]."
Feedback: Identify one strong point and one area for improvement in the statements.
Common Pitfalls: Be aware of typical mistakes in evaluating reasoning questions, such as misinterpreting the statements or overlooking key concepts.
Collaborative Evaluation: Consider discussing your evaluations with peers to gain different perspectives and enhance your understanding.
Iterative Feedback: After your evaluation, reflect on your process and consider how you might improve your approach in future assessments.
This revised prompt aims to enhance the evaluation process by providing clear guidelines, encouraging deeper analysis, and making it applicable to a wide range of reasoning questions.

Meta Learning via Learned Response Template (MAML-style)

To begin our analysis, we will methodically evaluate each statement to determine their accuracy based on established principles in machine learning and statistical theory.

Meta Learning via Learned Response Template (Reptile-style)

Let's analyze the question and the provided statements or options in detail. We will evaluate the truth of each statement step by step, considering relevant concepts and definitions to ensure a clear understanding. This methodical approach will guide us to the correct answer. Here's the breakdown:

B.3 GPQA Diamond Benchmark

The results for the GPQA Diamond Benchmark are provided below:

COPRO

Analyze the provided ‘question‘ in depth, synthesizing relevant information and context to generate a comprehensive a nd insightful ‘answer‘. Ensure your response is not only clear and concise but also addresses potential follow-up questions, providing exam ples or explanations that enhance understanding and demonstrate the broader implications of the topic. — Follow the following format. Question: The question to answer. Reasoning: Let’s think step by step in order to $produce the answer. We ... Here is a thorough and insightful answer to your question: The answer to the question. The last line of your response should be of the foll owing format: ’Answer: $LETTER’ (without quotes) where LETTER is one of ABCD. —

MIPROv2

Given the multiple-choice question provided in the ‘question‘ field, please analyze the statements step by step to determine the correct an swer. Your response should include a clear rationale explaining your reasoning process, followed by the final answer formatted as ’Answer: $LETTER’, where LETTER is one of A, B, C, or D.
—
Follow the following format.
Question: The question to answer.
Reasoning: Let’s think step by step in order to $produce the answer. We ...
Answer: The answer to the question. The last line of your response should be of the following format: ’Answer: $LETTER’ (without quotes) wh ere LETTER is one of ABCD.
—
Question: Answer the following multiple choice question. The last line of your response should be of the following format: ’Answer: $LETTER ’ (without quotes) where LETTER is one of ABCD. Think step by step before answering. Statement 1| Linear regression estimator has the small est variance among all unbiased estimators. Statement 2| The coefficients α assigned to the classifiers assembled by AdaBoost are always no n-negative. A) True, True B) False, False C) True, False D) False, True Reasoning: Let’s think step by step in order to evaluate the truth of the statements. For Statement 1, it is known that while linear regres sion is a popular estimator, it does not necessarily have the smallest variance among all unbiased estimators; this is a property of the Ga uss-Markov theorem under certain conditions. Therefore, Statement 1 is False. For Statement 2, in AdaBoost, the coefficients assigned to th e classifiers are indeed always non-negative, as they represent the weight of each classifier in the ensemble. Thus, Statement 2 is True.
Answer: D
—
(5 more Q&A pairs)
—

Meta Learning via Learned Loss Prompt

Below is a multiple-choice question along with the selected answer. Your task is to evaluate the reasoning behind the chosen answer. 1. **Clarify the Context**: Understand that the "prediction" refers to the selected answer to the question.
2. **Analyze the Reasoning**: Break down the reasoning steps that led to the selected answer. Identify the key concepts or principles involved in the question. 3. **Consider Alternatives**: Think about whether other answers could also be valid. What reasoning could support these alternative answers? 4. **Identify Assumptions**: What assumptions underlie the selected answer? Are there any logical fallacies present in the reasoning?
5. **Provide Justifications**: For each objection raised, provide a rationale that explains why the reasoning may be flawed.
6. **Compare Answers**: How does the reasoning for the selected answer differ from that of the correct answer?
7. **Summarize Your Findings**: Conclude with a brief summary of your evaluation, highlighting the strengths and weaknesses of the reasoning.

Meta Learning via Learned Loss Prompt (cont.)

8. **Reflect on the Process**: Consider how your own reasoning process could be improved for future evaluations.
9. **Holistic View**: What does this evaluation reveal about common misconceptions in the subject area?
This prompt is designed to guide you through a structured and comprehensive analysis of the reasoning question, ultimately leading to better insights and understanding.

Meta Learning via Learned Response Template (MAML-style)

To begin our analysis, let's clearly outline the assumptions we are making regarding the context of the problem. First, we will break down the reasoning step by step, ensuring that we address key factors such as the specific conditions, potential alternatives, and the implications of our findings. This structured approach will help us navigate through the complexities of the question and arrive at a well-supported conclusion.

Meta Learning via Learned Response Template (Reptile-style)

Let's analyze the question carefully and approach it step by step. We will consider the key details and relevant concepts before arriving at a conclusion. This methodical approach will help ensure that we arrive at the correct answer. Now, let's proceed with the analysis:

C Initial Loss Prompts

The following are the initial loss prompts used for each benchmark. These prompts serve as the starting point for the iterative optimization of the loss prompt $\theta_{\mathcal{L}}$, as described in TextGrad [1]. Each prompt is manually designed to guide the model in critically evaluating the problem or prediction in a specific domain. We use them as the first-step loss prompt in our experiments.

BBH Word Sorting Benchmark

Below is a word sorting problem and an answer. You are an expert scientist. Your job is to investigate the answer. Critically go through reasoning steps, consider your knowledge, and see if the answer is correct or if there are any critical mistakes.

MMLU Machine Learning Benchmark

Below is a multi-choice question and an answer. You are an expert scientist. Your job is to investigate the answer. Critically go through reasoning steps, consider your knowledge, and see if the answer is correct or if there are any critical mistakes.

GPQA Diamond Benchmark

Below is a multi-choice question and a prediction. You are an expert scientist. Your job is to investigate the prediction. Critically go through reasoning steps, and see if there is a reason why the prediction could be incorrect. Use the Janusian Process. Think about whether alternative answers could be true. Raise creative and critical objections to the solution, when needed.

D Related Work

D.1 Prompt Optimization in LLMs

Prompt optimization has become a critical area of research in improving the performance of large language models (LLMs). Initial strategies, such as few-shot learning and in-context learning, demonstrated that careful prompt design could significantly boost LLM performance across various tasks [20]. Techniques like Chain-of-Thought (CoT) reasoning [15] and ensemble methods [21] also emerged as popular ways to structure prompts for more effective problem-solving.

Efforts to automate prompt optimization have led to the development of gradient-based approaches [22, 23]. These methods, however, require access to the internal model parameters, which limits their application to open-source models. To address these limitations, approaches using LLMs themselves as prompt optimizers were introduced [24, 25, 4, 26, 1]. Our work draws inspiration from these frameworks but goes beyond prompt optimization to include inference-time optimization of diverse reasoning tasks.

D.2 Meta-Learning Algorithms

Meta-learning focuses on learning models that generalize better across tasks by optimizing either the loss functions or initializations. In terms of loss functions, the idea of learning loss functions through meta-learning has shown promise in improving model performance across various domains. For example, "Meta-Learning via Learned Loss"[9] presents a method for training parametric loss functions that can generalize across tasks. In the area of good initializations, Model-Agnostic Meta-Learning (MAML)[7] and its first-order variant, Reptile[8], have become foundational. MAML optimizes model parameters so that the model can adapt quickly to new tasks with minimal gradient updates. Reptile simplifies this by avoiding second-order derivatives, making it more computationally efficient. Both algorithms have inspired numerous advancements in meta-learning for fast task adaptation, which we incorporate in our approach to inference-time optimization by learning better initialization strategies for LLMs.