

LADA: Enabling Adaptation of Black-Box LLMs to Dynamic Domain Changes at Test Time

Yihao Hu^{1,2}, Congyu Qiao^{1,2}, Junming Yang^{1,2}, Mofei Song^{1,2}, Xin Geng^{1,2}, Ning Xu^{1,2*}

¹School of Computer Science and Engineering, Southeast University, Nanjing, China

²Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China
{yhhu, qiaocy, jmingyang, songmf, xgeng, xning}@seu.edu.cn

Abstract

Adaptation of black-box large language models (LLMs) at test time seeks to adjust LLMs to target domains in order to handle distribution shifts during testing, without access to model parameters. Most existing approaches rely on adapters trained with substantial target-domain data, which is often scarce in practical scenarios, and these adapters are tightly coupled to their training distribution, leading to degraded performance in dynamic scenarios with changing domains. To solve this, we propose LADA, a framework leveraging meta-training for continuous and rapid adaptation of black-box LLMs to unseen and changing domains. Specifically, LADA meta-trains an adapter on diverse tasks that cover multiple datasets and error combinations to learn transferable adaptation skills. At test time, the adapter needs only a few target-domain examples for lightweight adaptation and guides the LLM stepwise toward domain-appropriate reasoning via adaptive selection of reliable steps, without any updates to the LLM parameters. Experiments on datasets spanning multiple domains validate the effectiveness of the proposed approach.

1 Introduction

Large language models (LLMs), such as GPT series (OpenAI et al., 2023) and Qwen family (Yang et al., 2024a), have demonstrated remarkable capabilities in language understanding, reasoning, and generation through training on vast text corpora. These models are now deployed across a wide range of real-world applications, including software engineering (Fan et al., 2023; Hou et al., 2024), healthcare (Wang and Zhang, 2024; Bedi et al., 2025), and legal assistance (Huang et al., 2023a; Zhou et al., 2024), where they often encounter inputs that deviate significantly from their pretraining distribution (Quiñonero-Candela et al., 2022).

To address this challenge, adapting LLMs to target domains at test time has received increasing attention (Shi et al., 2024; Hu et al., 2025a). A growing body of work focuses on white-box adaptation methods, which update internal model parameters via fine-tuning on retrieved data (Hardt and Sun, 2024; Hübötter et al., 2025) or via perplexity minimization on unlabeled test inputs (Hu et al., 2025a). While effective, these approaches require full access to model weights and incur high computational costs, making them incompatible with black-box, API-served LLMs widely used in practice.

In response, recent studies explore black-box strategies that leverage auxiliary adapters to guide the LLM without modifying its parameters. For example, Liu et al. (2024) use a small trainable model to adjust next-token probabilities, while Shi et al. (2024) and Sun et al. (2024) employ rerankers or scoring models to select among multiple generated candidates. However, these methods share a critical limitation: they rely on substantial target-domain data for adapter training, resulting in domain-tied adapters that fail in dynamic real-world settings, where task requirements and data distributions shift, such as general-purpose assistants operating under abrupt and unpredictable domain shifts across user sessions. Re-collecting domain-specific data and retraining adapters for every new context is impractical, and assuming prior knowledge of future domains contradicts the open-ended nature of dynamic deployment environments.

To overcome this challenge, we propose a novel framework named LADA, i.e., Learning-to-ADapt for black-box LLMs, that enables continuous and rapid adaptation of black-box LLMs to unseen and changing domains at test time, even without prior domain information. Unlike previous methods with static and domain-specific adapters, LADA meta-trains an adapter across diverse tasks spanning multiple publicly available datasets and error combi-

*Corresponding author.

nations, enabling it to learn transferable step-level error patterns and quickly adapt to new domains using a few examples. At test time, the adapter employs a few target-domain positive–negative reasoning pairs for lightweight adaptation, then guides the black-box LLM step-by-step by adaptively selecting reliable reasoning steps and steering generation toward domain-appropriate trajectories without parameter updates or prior domain exposure. Critically, its ability to generalize across domains and error types ensures effectiveness as the target domain changes. Our contributions can be summarized as follows:

- We propose LADA, a meta-learning-based framework for black-box LLMs that enables continuous adaptation to unseen and changing domains at test time, eliminating reliance on prior knowledge of specific target domains.
- LADA avoids the need for substantial training data from a specific target domain. Instead, it meta-trains an adapter on diverse tasks spanning multiple available datasets and error combinations, and requires only a few target-domain examples for lightweight adaptation at test time.
- The few-shot adaptation and stepwise adaptive selection mechanism introduce a controllable overhead at test time, making LADA feasible for real-world deployment.

2 Related Work

2.1 Adapting LLMs at Test Time.

In general, this paradigm can be categorized into white-box and black-box settings, depending on whether LLM parameters are accessible. In the white-box setting, the parameters are fully accessible and can be directly updated. For each test input, [Hardt and Sun \(2024\)](#) retrieve its neighbors and fine-tune the model on their text at test time. Building on this idea, [Hübötter et al. \(2025\)](#) further reduce redundancy among the selected data to optimize the overall information gain of the chosen examples. Along similar lines, [Yu et al. \(2025\)](#) enable LLM to retrieve and learn from related reasoning traces to enhance reasoning capabilities. More recently, [Hu et al. \(2025a\)](#) reveal that more accurate predictions can be obtained by minimizing the input perplexity of unlabeled test data. Yet, these approaches incur substantial computational overhead and are inapplicable to today’s API-served LLMs, where parameters are inaccessible, thereby limiting their practicality in real-world scenarios.

This motivates black-box settings, where adaptation must be performed without access to or modification of LLM parameters. Some methods assume access to the output token probabilities of the LLM and perform adaptation by correcting the probability distribution of the next token. For example, [Huang et al. \(2023b\)](#) adaptively interpolate the LLM’s prediction probabilities with retrieval results from a datastore. [Ormazabal et al. \(2023\)](#) and [Liu et al. \(2024\)](#) leverage a smaller model fine-tuned on target-domain data to correct the LLM’s next-token probabilities using its context-dependent predictions during inference. It is worth noting that some close source LLMs, such as Claude models, do not expose token-level probabilities, thereby restricting the applicability of such methods. Another line of work relies only on observable outputs generated by LLM. They typically leverage target-domain training data to train a scoring model that ranks candidate responses through multiple sampling. For instance, [Khalifa et al. \(2023\)](#) use a step-level scoring model to select the best reasoning step, [Sun et al. \(2024\)](#) employ an adapter to guide sentence-level beam search, and [Shi et al. \(2024\)](#) utilize a reranker to rank complete solutions, thereby enabling adaptation to the target domain. However, these approaches assume access to substantial target-domain data, which is not always feasible in practical scenarios.

2.2 Canonical Test-Time Adaptation.

Canonical test-time adaptation (TTA) has established a robust paradigm for mitigating distribution shifts by adapting pre-trained models to unlabeled test data before making predictions ([Liang et al., 2025](#)). Most representative methods, such as those focusing on entropy minimization ([Wang et al., 2021](#); [Niu et al., 2022](#)), pseudo-labeling ([Goyal et al., 2022](#); [Hu et al., 2025b](#)), or consistency ([Döbler et al., 2023](#)) are primarily designed for white-box settings where internal parameters or batch-norm statistics can be updated. While these methods have achieved remarkable success in discriminative tasks like image classification, their direct application to the adaptation of black-box LLMs faces challenges. In this work, we distinguish our setting from the canonical unsupervised TTA. Unlike previous unsupervised signals, LADA leverages a few positive–negative reasoning pairs from the target domain for lightweight adapter tuning at test time. The formal formulation of this setting is provided in Section 3.1.

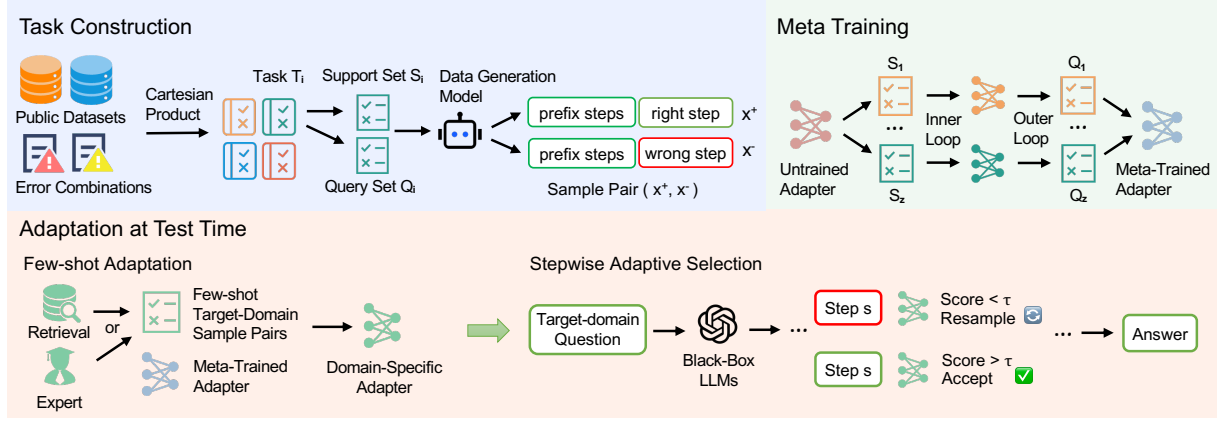


Figure 1: Overview of the LADA framework.

3 Proposed Method

3.1 Preliminaries

We begin by introducing the notations used throughout the paper and outlining the problem setting. In step-by-step reasoning tasks such as question answering or problem solving, given an input question q , a black-box LLM π_{LLM} incrementally generates a sequence of l intermediate reasoning steps $\mathbf{r} = (r_1, \dots, r_l)$. Let $\pi_{\text{LLM}}(\cdot | c)$ denote the next-sentence distribution given context c , and $\text{Sample}(p, t)$ denote the outcome of the t -th sampling trial from distribution p . Given the previously accepted reasoning steps $\hat{\mathbf{r}}_{<s} = (\hat{r}_1, \dots, \hat{r}_{s-1})$, the t -th sampling candidate for the s -th reasoning step is defined as r_s^t by:

$$r_s^t = \text{Sample}(\pi_{\text{LLM}}(\cdot | q, \hat{\mathbf{r}}_{<s}), t). \quad (1)$$

Under the standard decoding process, only a single candidate is sampled at each step, and the accepted step $\hat{r}_s$ is set to this candidate, i.e., $\hat{r}_s = r_s^1$. When π_{LLM} is applied to a new target domain, distribution shift and decoding stochasticity may induce incorrect reasoning steps, manifesting as hallucination, repetition, or missing steps. Once such errors occur, subsequent reasoning typically derails and fails to reach the correct answer. To address this problem, we aim to utilize an adapter that acts as a step-level scoring model for detecting erroneous steps. As reasoning error patterns differ across domains, the meta-trained adapter f_{θ} is quickly adapted using a small set of positive-negative reasoning pairs $\mathcal{B}_T$ from the target domain at test time, yielding a domain-specific adapter $f_{\hat{\theta}}$. At each step, candidates are sampled sequentially and evaluated on the fly by the adapter, with resampling performed until an acceptable candidate is found.

Formally, given the question q from the target distribution $p_T(q)$ and $\hat{\mathbf{r}}_{<s}$, we define the accepted step $\hat{r}_s$ as the first candidate whose score exceeds the threshold τ :

$$\begin{aligned} \hat{r}_s &= r_s^{t^*}, \\ t^* &= \min \{t \mid \tanh(f_{\hat{\theta}}(q, \hat{\mathbf{r}}_{<s}, r_s^t)) > \tau\}. \end{aligned} \quad (2)$$

By iteratively applying this selection process, the adapter $f_{\hat{\theta}}$ guides π_{LLM} step by step, accepting reliable reasoning steps and avoiding potential reasoning errors, thereby steering the reasoning toward domain-appropriate trajectories.

3.2 Overview

LADA achieves adaptation of black-box LLMs at test time using an adapter that is meta-trained in advance. To meta-train the adapter, we construct tasks via the Cartesian product of different datasets and error combinations, with each task containing step-level positive-negative reasoning pairs with the same dataset and error combination. Then these tasks are utilized to meta-train the adapter as a step-level scorer in an inner-outer loop: the inner loop adapts to each task with a few samples, while the outer loop optimizes meta-parameters to capture transferable patterns across diverse tasks, thereby enabling effective adaptation to dynamic domain changes. At test time, the adapter is rapidly optimized with a few paired examples from the target domain, which can be obtained from expert annotations or retrieval, and guides the black-box LLM stepwise by adaptively retaining reliable steps while resampling unreliable ones. The overall framework of LADA is illustrated in Figure 1.

3.3 The LADA Framework

3.3.1 Task Construction

We aim to construct diverse meta-training tasks to simulate potential adaptation scenarios and enable the meta-trained adapter to capture transferable patterns across domains. π_{LLM} may produce various errors during reasoning, such as hallucination, repetition, or missing steps. To train the adapter to recognize different types of reasoning errors, following Golovneva et al. (2023), we formalize these errors into m categories, denoted as $\{e_g\}_{g=1}^m$. At test time, we aim to quickly adapt the meta-trained adapter using n positive-negative sample pairs from the target domain, where each pair is associated with one error type, allowing the adapter to adapt to error patterns specific to the target domain. To simulate such adaptation scenarios, we construct meta-training tasks by considering combinations of error types with repetition. Specifically, from the m error types, we form combinations with repetition of size n , yielding a total of $c = \binom{m+n-1}{n}$ distinct error-type combinations, each corresponding to an error-type configuration of the n pairs. The resulting set of combinations $\mathcal{E}$ is formally defined as:

$$\mathcal{E} = \left\{ (e_{g_1}, \dots, e_{g_n}) \mid (g_1, \dots, g_n) \in \{1, \dots, m\}^n, g_1 \leq \dots \leq g_n \right\}. \quad (3)$$

To enable meta-learning of cross-domain reasoning patterns, we use multiple publicly available datasets $\{\mathcal{D}_i\}_{i=1}^a$. Each dataset comes from a different domain and is defined as $\mathcal{D}_i = \{(q_j, \mathbf{r}_j)\}_{j=1}^{b_i}$, where for each example $\mathbf{r}_j = (r_{j,s})_{s=1}^{l_j}$ represents a sequence of reasoning steps. Taking the Cartesian product of the $\{\mathcal{D}_i\}_{i=1}^a$ and $\mathcal{E}$ yields ac tasks, where each task $\mathcal{T}_i$ is specified by a specific dataset and an error combination, and is intended to simulate an adaptation scenario that may arise in real-world deployment. Formally, the meta-learning task set is $\mathcal{M} = \{\mathcal{T}_i\}_{i=1}^{ac}$.

With the meta-training tasks defined, we next describe the internal organization of each task. Each task $\mathcal{T}_i$ consists of a support set $\mathcal{S}_i$ and a query set $\mathcal{Q}_i$. The support set $\mathcal{S}_i$ contains n reasoning pairs, where each negative sample corresponds to one error type in the corresponding error combination. The query set $\mathcal{Q}_i$ is designed to serve two purposes, to evaluate how well the adapter learns the error types present in $\mathcal{S}_i$ and to assess whether it can

generalize to other error types not observed in $\mathcal{S}_i$. In practice, $\mathcal{Q}_i$ contains $m + n$ reasoning pairs: n task-aligned pairs with the same error-type combination as $\mathcal{S}_i$, and m pairs, one for each predefined error type, all using examples distinct from those in $\mathcal{S}_i$. This design intentionally biases the evaluation towards the current task’s error combination by making the corresponding error types appear more frequently in $\mathcal{Q}_i$, while encouraging generalization to other error types.

Finally, at the finest granularity, we describe the construction of the basic training units within each task, namely positive-negative reasoning pairs. These paired examples allow the adapter to explicitly contrast correct and erroneous reasoning steps, enabling it to learn discriminative step-level error patterns. For the construction of these pairs, we aim to ensure balanced coverage across error types, but error types in real-world data typically follow a long-tailed distribution (Golovneva et al., 2023). Using real-world data alone often results in insufficient samples for certain long-tail error types. In practice, we employ a data synthesis approach (Tan et al., 2024; Yang et al., 2024b). We employ an oracle LLM π_{oracle} to synthesize positive-negative reasoning pairs for each error type e_g in three steps. Given an example $(q_j, \mathbf{r}_j)$, π_{oracle} first selects a step r_{j,s_g} from $\mathbf{r}_j$ that is susceptible to the error type e_g . Conditioned on $(q_j, \mathbf{r}_j)$ and the chosen step r_{j,s_g} , π_{oracle} then produces two corresponding variants: a correct argumentation r_{j,s_g}^o that augment the step, and an erroneous counterpart $r_{j,s_g}^{e_g}$ that distorts it. This process can be formalized as:

$$\begin{cases} r_{j,s_g}^o \sim \pi_{\text{oracle}}(\cdot \mid (q_j, \mathbf{r}_j), r_{j,s_g}, \langle o \rangle) \\ r_{j,s_g}^{e_g} \sim \pi_{\text{oracle}}(\cdot \mid (q_j, \mathbf{r}_j), r_{j,s_g}, \langle e_g \rangle) \end{cases}, \quad (4)$$

where $\langle o \rangle$ and $\langle e_g \rangle$ denote the prompts for correct argumentation and error generation of type e_g , respectively. Finally, by truncating the reasoning sequence after r_{j,s_g} , we construct a positive-negative reasoning pair $(\mathbf{x}_j^+, \mathbf{x}_j^-)$, where $\mathbf{x}_j^+ = (q_j, r_{j,1}, \dots, r_{j,s_g-1}, r_{j,s_g}^o)$ and $\mathbf{x}_j^- = (q_j, r_{j,1}, \dots, r_{j,s_g-1}, r_{j,s_g}^{e_g})$. The two reasoning sequences differ only in their final step, and continuing reasoning from $\mathbf{x}_j^-$ typically leads π_{LLM} to fail in producing the correct answer. This design isolates the impact of a single step correctness, allowing the adapter to focus on step-level scoring while avoiding interference from later steps (Khalifa et al., 2023). In addition, we use a judge model π_{judge} to evaluate and filter the synthetic samples

for each error type, ensuring the quality of the synthesized data.

3.3.2 Meta-Training

Building on the constructed task set, we employ a meta-training (Finn et al., 2017) stage to enable f_θ to be quickly adapted when only a few target-domain examples are available. In each meta-training iteration, we first sample z tasks $\{\mathcal{T}_i\}_{i=1}^z$ from the task set $\mathcal{M}$. In the inner loop, the goal is to obtain the task-specific adapter for $\mathcal{T}_i$. To this end, the base adapter f_θ is adapted by minimizing the inner loss $\mathcal{L}^{\text{in}}$ on the corresponding support set $\mathcal{S}_i$:

$$\min_{\theta} \mathcal{L}^{\text{in}}(f_\theta, \mathcal{S}_i), \quad (5)$$

which simulates the adaptation to an unseen target task with a few examples. The resulting task-specific parameters are obtained through the update:

$$\theta'_i \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}^{\text{in}}(f_\theta, \mathcal{S}_i), \quad (6)$$

where α is the adaptation step size. Subsequently, the outer loop evaluates the generalization of the adapted parameters θ'_i on the corresponding query set $\mathcal{Q}_i$ through the outer loss $\mathcal{L}^{\text{out}}$. By minimizing the aggregated $\mathcal{L}^{\text{out}}$ across all query sets $\{\mathcal{Q}_i\}_{i=1}^z$, we obtain the meta-objective:

$$\min_{\theta} \sum_{i=1}^z \mathcal{L}^{\text{out}}(f_{\theta'_i}, \mathcal{Q}_i). \quad (7)$$

Since each θ'_i is derived from θ through the inner update, the meta-objective's dependence on θ'_i implicitly links the optimization to θ . Thus, the meta-parameters are updated as:

$$\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{i=1}^z \mathcal{L}^{\text{out}}(f_{\theta'_i}, \mathcal{Q}_i), \quad (8)$$

where β is the meta step size. Specifically, the meta-update relies on a meta-gradient calculated over the performance on a variety of query sets. This meta-gradient averages information from multiple tasks, thereby preventing the model from overfitting to any single task's data. Consequently, the meta-parameters are steered towards a general-purpose initialization that captures the shared structure across different tasks, which is key for acquiring transferable knowledge.

In practice, the inner objective is defined using

Algorithm 1 LADA Algorithm

Require: Black-box LLM π_{LLM} , pretrained adapter f_θ , publicly available datasets $\{\mathcal{D}_i\}_{i=1}^a$, reasoning error types $\{e_g\}_{g=1}^m$, meta batch size z , a small set of target-domain reasoning pairs $\mathcal{B}_T$;

Task Construction:

1: Construct meta-training task set $\mathcal{M}$ based on $\{\mathcal{D}_i\}_{i=1}^a$ and $\{e_g\}_{g=1}^m$ via data synthesis and filtering;

Meta Training:

2: **while** not converged **do**

3: Sample z tasks $\{\mathcal{T}_i\}_{i=1}^z$ from $\mathcal{M}$;

4: **for all** $\mathcal{T}_i$ **do**

5: Compute θ'_i based on $\mathcal{S}_i$ using Eq. (6);

6: **end for**

7: Update θ based on $\{\mathcal{Q}_i\}_{i=1}^z$ using Eq. (8);

8: **end while**

Adaptation at Test Time:

9: Obtain the adapted parameters $\hat{\theta}$ based on $\mathcal{B}_T$ using Eq. (11);

10: **for** $q \sim p_T(q)$ **do**

11: Obtain adapted response $\hat{r}$ step by step using Eq. (1) and Eq. (13);

12: **end for**

Ensure: Adapted response $\hat{r}$ for $q \sim p_T(q)$.

the max-margin loss:

$$\mathcal{L}^{\text{in}}(f_\theta, \mathcal{S}_i) = \frac{1}{n} \sum_{j=1}^n \max\left(0, \zeta - \left[\tanh\left(f_\theta\left(x_j^+\right)\right) - \tanh\left(f_\theta\left(x_j^-\right)\right) \right] \right), \quad (9)$$

where ζ is the margin hyperparameter. For the outer objective, we adopt the same form of max-margin loss:

$$\mathcal{L}^{\text{out}}(f_{\theta'_i}, \mathcal{Q}_i) = \frac{1}{m+n} \sum_{j=1}^{m+n} \max\left(0, \zeta - \left[\tanh\left(f_{\theta'_i}\left(x_j^+\right)\right) - \tanh\left(f_{\theta'_i}\left(x_j^-\right)\right) \right] \right). \quad (10)$$

Through meta-learning across diverse tasks, the adapter acquires an initialization θ that encodes cross-task patterns, allowing it to adapt efficiently to unseen domains with limited data and remain effective in dynamic real-world scenarios.

3.3.3 Adaptation at Test Time

Building on the meta-trained adapter, we proceed to describe the adaptation at test time process, con-

sisting of the few-shot adaptation of the adapter and stepwise adaptive selection. For few-shot adaptation, we exploit a small set of positive–negative reasoning pairs $\mathcal{B}_T = \{(x_j^+, x_j^-)\}_{j=1}^n$ from the target domain, which can be obtained through expert annotations or retrieval, to quickly adapt the meta-trained adapter f_θ by minimizing the inner loss:

$$\hat{\theta} \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{L}^{\text{in}}(f_\theta, \mathcal{B}_T), \quad (11)$$

where γ is the adaptation step size. Through this process, $f_{\hat{\theta}}$ acquires the ability to score reasoning steps in accordance with the target domain, then it steers π_{LLM} stepwise toward domain-appropriate reasoning trajectories. For a question $q \sim p_T(q)$, the basic stepwise selection process is described in Eq. (2), which accepts the first candidate reasoning step whose score exceeds a threshold τ . In practice, to prevent the selection process from entering an infinite loop, we set a maximum number of resampling trials $t_{\max}$. When no candidate exceeds the threshold within $t_{\max}$ trials, we fall back to selecting the candidate with the highest score, whose index t' is given by:

$$t' = \arg \max_{1 \leq t \leq t_{\max}} f_{\hat{\theta}}(q, \hat{r}_{<s}, r_s^t). \quad (12)$$

Overall, the acceptance of step s can be formalized as $\hat{r}_s = r_s^{\hat{t}}$, where $\hat{t}$ denotes the accepted index:

$$\hat{t} = \begin{cases} t^*, & t^* \leq t_{\max}, \\ t', & \text{otherwise.} \end{cases} \quad (13)$$

By iteratively applying this adaptive selection strategy, LADA effectively constructs a complete and adapted reasoning path $\hat{r} = (\hat{r}_1, \dots, \hat{r}_l)$ for a given question $q \sim p_T(q)$ using the domain-specific adapter $f_{\hat{\theta}}$. The algorithmic description of LADA is presented in Algorithm 1.

4 Experiments

4.1 Experimental Configurations

4.1.1 Datasets

We evaluate LADA on ten subsets of MMLU (Hendrycks et al., 2021), including high school physics (Phys.), high school statistics (Stats.), high school microeconomics (Microecon.), high school computer science (CS), machine learning (ML), formal logic (Form. Log.), logical fallacies (Log. Fall.), professional accounting (Prof. Acc.), professional medicine (Prof. Med.), and

electrical engineering (EE). Each selected subset is treated as a distinct target domain, and dynamic domain changes are simulated by switching between subsets. Our meta-training data are constructed from three datasets: GSM8K (Cobbe et al., 2021), StrategyQA (Geva et al., 2021), and ReClor (Yu et al., 2020). These datasets differ from the selected MMLU subsets, encouraging the adapter to learn domain-general error patterns rather than domain-specific knowledge. Complete dataset details are given in Appendix A.1.

4.1.2 Baselines

We compare LADA with four black-box baselines:

- ZERO-SHOT-COT (Kojima et al., 2022): A prompt-based approach that instructs the model to “think step by step” at test time to derive the final answer.
- COT-PROMPTING (Wei et al., 2022): A prompt-based approach that augments the prompt with chain-of-thought examples to guide multi-step reasoning.
- SELF-CONSISTENCY (Wang et al., 2023): A decoding-based approach that samples multiple solutions and aggregates them by majority voting to obtain the final answer.
- BBOX-ADAPTER (Sun et al., 2024): A training-based approach that learns a scoring model on the target-domain training set and leverages it to guide beam search for the final prediction.

4.1.3 Implementation Details

For the black-box LLMs, we simulate API-style access via vLLM (Kwon et al., 2023) and use two representative open-source models: Qwen2-7B-Instruct (Yang et al., 2024a) and Mixtral-8x7B-Instruct-v0.1 (Jiang et al., 2024). As the adapter, we use Qwen3-0.6B (Yang et al., 2025), followed by a single linear layer.

For task construction, we synthesize $m = 7$ types of commonly observed reasoning errors (Golovneva et al., 2023). This results in 252 meta-training tasks. We use DeepSeek-V3 (DeepSeek-AI et al., 2024) as π_{oracle} and Gemini 2.5 Flash as π_{judge} . Additional details on task construction, including the selected error types, illustrative examples, and the prompts used for data synthesis and post-processing, are provided in Appendix A.2.

For meta-training, we use first-order MAML and train for 2 epochs. In each epoch, we iterate over all tasks with meta-batch size $z = 8$. The inner update uses 3 gradient steps with $\alpha = 1 \times 10^{-4}$ and

Table 1: Reasoning accuracy (%) of different methods, reported as mean $\pm$ standard deviation across changing target-domains.

Methods	Model	Phys.	Stats.	Microecon.	CS	ML
ZERO-SHOT-COT	Qwen2-7B	46.36 $\pm$ 3.90	61.42 $\pm$ 1.09	76.47 $\pm$ 2.40	75.67 $\pm$ 1.25	44.94 $\pm$ 3.67
COT-PROMPTING		52.98 $\pm$ 1.87	60.34 $\pm$ 0.79	81.09 $\pm$ 0.34	77.33 $\pm$ 2.49	47.92 $\pm$ 0.42
SELF-CONSISTENCY		54.97 $\pm$ 1.32	62.19 $\pm$ 1.34	82.63 $\pm$ 0.87	78.67 $\pm$ 1.15	49.70 $\pm$ 0.52
BBOX-ADAPTER		53.64 $\pm$ 1.99	62.04 $\pm$ 1.60	82.21 $\pm$ 1.75	79.33 $\pm$ 2.52	48.51 $\pm$ 2.58
LADA		57.17 $\pm$ 1.38	64.35 $\pm$ 1.39	83.19 $\pm$ 1.11	79.67 $\pm$ 1.53	50.60 $\pm$ 0.52
ZERO-SHOT-COT	Mixtral-8x7B	45.58 $\pm$ 3.04	56.21 $\pm$ 2.74	69.47 $\pm$ 1.70	73.67 $\pm$ 0.47	47.32 $\pm$ 1.26
COT-PROMPTING		53.98 $\pm$ 2.75	62.44 $\pm$ 1.51	78.01 $\pm$ 1.10	77.26 $\pm$ 0.90	47.92 $\pm$ 2.34
SELF-CONSISTENCY		55.19 $\pm$ 1.67	63.58 $\pm$ 0.71	78.99 $\pm$ 0.42	78.67 $\pm$ 1.53	49.11 $\pm$ 0.89
BBOX-ADAPTER		54.53 $\pm$ 2.68	63.27 $\pm$ 0.96	79.13 $\pm$ 1.75	78.00 $\pm$ 2.65	48.51 $\pm$ 3.38
LADA		56.95 $\pm$ 1.99	65.28 $\pm$ 1.67	79.69 $\pm$ 0.49	79.33 $\pm$ 2.08	49.40 $\pm$ 1.03
Methods	Model	Form. Log.	Log. Fall.	Prof. Acc.	Prof. Med.	EE
ZERO-SHOT-COT	Qwen2-7B	46.03 $\pm$ 1.37	73.42 $\pm$ 0.77	49.65 $\pm$ 0.29	68.14 $\pm$ 0.76	65.29 $\pm$ 2.60
COT-PROMPTING		46.30 $\pm$ 1.87	72.60 $\pm$ 0.94	54.26 $\pm$ 0.77	68.63 $\pm$ 1.05	67.36 $\pm$ 0.65
SELF-CONSISTENCY		48.41 $\pm$ 0.79	74.23 $\pm$ 0.61	55.79 $\pm$ 0.41	69.24 $\pm$ 0.77	69.20 $\pm$ 0.80
BBOX-ADAPTER		47.35 $\pm$ 2.55	73.01 $\pm$ 1.62	54.85 $\pm$ 1.08	68.87 $\pm$ 1.39	68.74 $\pm$ 2.11
LADA		49.47 $\pm$ 1.21	75.26 $\pm$ 0.29	56.03 $\pm$ 0.71	69.85 $\pm$ 1.27	69.89 $\pm$ 1.44
ZERO-SHOT-COT	Mixtral-8x7B	45.62 $\pm$ 2.26	70.49 $\pm$ 3.06	45.37 $\pm$ 0.64	70.80 $\pm$ 2.25	66.21 $\pm$ 0.69
COT-PROMPTING		51.85 $\pm$ 1.65	75.87 $\pm$ 1.61	54.79 $\pm$ 1.39	79.02 $\pm$ 1.41	66.90 $\pm$ 3.38
SELF-CONSISTENCY		52.38 $\pm$ 0.79	77.10 $\pm$ 0.71	55.91 $\pm$ 0.89	79.78 $\pm$ 0.37	67.36 $\pm$ 1.59
BBOX-ADAPTER		51.06 $\pm$ 2.29	76.48 $\pm$ 1.54	55.20 $\pm$ 1.43	79.41 $\pm$ 1.33	67.52 $\pm$ 2.07
LADA		53.97 $\pm$ 1.37	77.71 $\pm$ 0.94	56.26 $\pm$ 0.82	80.39 $\pm$ 0.56	68.28 $\pm$ 1.38

is performed with SGD. The outer update uses a single gradient step with $\beta = 5 \times 10^{-6}$, optimized using AdamW (Loshchilov and Hutter, 2019) with a weight decay of 0.01, with a cosine decay learning rate. We set the margin ζ in both inner and outer losses to 1.

For adaptation at test time, we perform 3 update steps using SGD, with the adaptation step size $\gamma = 1 \times 10^{-4}$. We set the acceptance threshold $\tau = 0$, and the maximum number of sampling attempts $t_{\max} = 3$. The decoding parameters were set to a temperature of 0.7 and a top-p value of 0.95. In principle, $\mathcal{B}_T$ could be obtained via expert annotation or retrieval, but in this work, we focus on the synthesis approach in order to isolate the effect of the proposed adaptation mechanism. In practice, we use $n = 3$ sample pairs for adaptation, which are obtained via the data synthesis procedure in Section 3.3.1.

For the baselines, SELF-CONSISTENCY samples 8 answers and BBOX-ADAPTER also uses Qwen3-0.6B as the adapter, which is initialized by training

on GSM8K, StrategyQA, and ReClor, then adapted on 3 target-domain examples with a beam size of 3, to ensure a fair comparison. All methods use the same 3 target-domain examples for in-context learning, which are updated as the target domain changes, except ZERO-SHOT-COT.

The MAML and test-time adaptation hyperparameters are selected using a held-out validation split from the meta-training task set. Specifically, we split the constructed meta-training tasks into training and validation tasks. The training tasks are used to meta-train the adapter with MAML, while the validation tasks are used to simulate test-time adaptation on unseen domains. We then choose the hyperparameters based on the validation performance after this simulated adaptation process.

4.2 Experimental Results

We conduct 3 trials and report the mean and standard deviation of reasoning accuracy. Results are summarized in Table 1, and the best performance is shown in boldface. The results show that:

- Across all evaluated domains and black-box

Table 2: Reasoning accuracy (%) of LADA and its variants across different target domains, reported as mean $\pm$ standard deviation.

Methods	Phys.	Stats.	Log. Fall.
LADA-ND	53.20 $\pm$ 1.67	60.80 $\pm$ 1.93	73.01 $\pm$ 2.81
LADA-NR	56.51 $\pm$ 0.76	63.43 $\pm$ 1.67	74.64 $\pm$ 0.94
LADA-NC	55.85 $\pm$ 2.02	63.73 $\pm$ 0.71	74.44 $\pm$ 1.54
LADA-NO	55.41 $\pm$ 1.01	63.12 $\pm$ 1.87	74.23 $\pm$ 1.23
LADA-NM	53.86 $\pm$ 2.68	61.57 $\pm$ 2.12	73.21 $\pm$ 1.28
LADA-NS	52.98 $\pm$ 1.87	60.34 $\pm$ 0.79	72.60 $\pm$ 0.94
LADA-NA	54.30 $\pm$ 1.99	62.50 $\pm$ 1.67	74.03 $\pm$ 2.32
LADA	57.17 $\pm$1.38	64.35 $\pm$1.39	75.26 $\pm$0.29

Table 3: Adaptation and 10-sample inference running time (s) and GPU memory consumption (GiB) of different methods on target domain. "-" denotes that the metric is not applicable.

Methods	Adaptation		Inference	
	Time	Mem.	Time	Mem.
COT-PROMPTING	-	-	25.48	-
SELF-CONSISTENCY	-	-	203.17	-
BBOX-ADAPTER	7.39	15.97	122.79	6.85
LADA	4.62	11.06	46.72	6.81

LLMs, LADA consistently achieves the best performance, outperforming all baselines, including prompt-based, decoding-based, and training-based methods.

- LADA achieves performance improvements to varying degrees across all target domains. In domains such as physics and statistics, the improvements exceed 3.5% compared to COT-PROMPTING, which demonstrates the effectiveness of LADA under distribution shifts.
- The performance gains of LADA are robust across different black-box LLMs. Performance improvements are observed on both Qwen2-7B-Instruct and Mixtral-8x7B-Instruct, suggesting that LADA does not rely on model-specific behaviors and can generalize to different backbone architectures.

4.3 Further Analysis

4.3.1 Ablation Study

We perform ablation studies on six design choices of LADA across three stages: task construction, meta-training, and adaptation at test time. Experiments are conducted on three subsets of MMLU: high

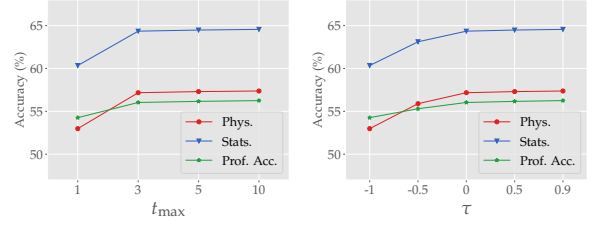


Figure 2: Sensitivity analysis of $t_{\max}$ and τ .

school physics, high school statistics, and logical fallacies, using Qwen2-7B-Instruct. Results are summarized in Table 2. In the task construction stage, we design four variants: LADA-ND, LADA-NR, LADA-NC, and LADA-NO. In LADA-ND, we construct an equal number of tasks solely from GSM8K, rather than from multiple datasets. This leads to a significant performance drop because meta-training is unable to capture cross-domain reasoning patterns. For LADA-NR, the error types of samples in the support set within each task are all different, with no repetition. While for LADA-NC, all samples in the support set within each task share the same error type, with no combination of different error types. Neither variant reflects the practical scenario in which a small number of examples may involve diverse error types, which leads to inferior performance compared to LADA. Finally, in LADA-NO, the query set contains only the error types that appear in the support set. This leads to performance degradation, as the adapter does not learn to generalize to unseen error types.

In the meta-training stage, we introduce one variant, LADA-NM, where the adapter is trained with standard supervised learning instead of meta-learning. Without meta-training, the adapter cannot quickly adjust to a target domain when only a small number of target samples are available, which leads to degraded performance. In the adaptation of LLMs at test time stage, we consider two variants: LADA-NS and LADA-NA. In LADA-NS, we accept the model’s first sample at each reasoning step without error-driven selection. As a result, it loses the ability to correct erroneous reasoning steps and degenerates to standard COT-PROMPTING. In LADA-NA, we directly apply the meta-trained adapter to the target domain without further adaptation. Although meta-training enables the adapter to learn transferable adaptation skills, the lack of target-domain adaptation leads to worse performance than LADA.

Table 4: Reasoning accuracy (%) of different methods across different target domains evaluate on GPT-4o-mini, reported as mean $\pm$ standard deviation.

Methods	CS	Phil.	Law
ZERO-SHOT-COT	63.09 $\pm$ 1.59	45.96 $\pm$ 1.61	34.97 $\pm$ 0.91
COT-PROMPTING	65.85 $\pm$ 1.22	49.37 $\pm$ 1.30	35.33 $\pm$ 0.82
SELF-CONSISTENCY	66.99 $\pm$ 0.86	49.97 $\pm$ 0.90	36.21 $\pm$ 0.50
BBOX-ADAPTER	66.59 $\pm$ 1.95	49.43 $\pm$ 2.12	35.60 $\pm$ 0.96
LADA	68.13 $\pm$1.10	50.50 $\pm$1.20	36.63 $\pm$0.59

4.3.2 Running Time and Memory Analysis

Computational efficiency is essential for making adaptation of models at test time practical (Niu et al., 2022; Song et al., 2023). To evaluate the computational efficiency of LADA, we measured the adaptation and inference running time and GPU memory consumption of LADA and the baselines on the high school physics subset of MMLU using Qwen2-7B-Instruct. The stepwise adaptation time was evaluated on a subset of 10 samples to simulate a single conversation with thematically related questions, as is typically the case in real-world scenarios (Deng et al., 2023). We measure peak GPU memory during both the few-shot adaptation and stepwise adaptive selection phases. The results are presented in Table 3. It can be observed that, compared with inference time, the few-shot adaptation time of LADA accounts for only a small fraction. Moreover, thanks to its stepwise adaptive selection mechanism, LADA requires less than one-quarter of the inference time used by SELF-CONSISTENCY for full-answer resampling, as it avoids redundant resampling of correct reasoning steps and instead focuses on resampling incorrect ones. In addition, LADA fine-tunes only a 0.6B-parameter model using a small number of samples, which helps keep the memory overhead manageable.

4.3.3 Parameter Sensitivity Analysis

Lastly, we study the sensitivity of two hyperparameters in the adaptation at test time stage of LADA, $t_{\max}$ and τ , on three domains with Qwen2-7B-Instruct. The results are presented in Figure 2. We observe that when $t_{\max} \geq 3$, the performance continues to improve with larger values of $t_{\max}$ but remains overall stable, indicating that LADA can correct faulty reasoning steps with only a small number of resampling attempts. Since the adapter is trained with a max-margin objective that separates correct and incorrect reasoning steps by a

positive score gap, setting $\tau = 0$ naturally aligns with the learned decision boundary. Lower values of τ tend to admit erroneous steps, while higher thresholds provide only marginal gains and trigger additional resampling of correct steps, leading to unnecessary inference overhead.

4.3.4 Evaluation on Closed-Source LLMs

We also evaluate LADA on closed-source API-based LLMs, specifically the GPT-4o-mini model, on three domain-specific subsets of the challenging MMLU-Pro (Wang et al., 2024) benchmark, including computer science (CS), philosophy (Phil.), and law. Results in Table 4 show that LADA consistently achieves the best performance across three different domains, which further demonstrates the robustness and practical effectiveness of LADA in real-world scenarios.

4.3.5 Case Study

Figure 4 presents case studies of LADA under different domains. For example, in the first case, the reasoning proceeds normally in the first five steps, with the adapter assigning scores above the acceptance threshold to the sampled candidate at each step. As a result, each of these steps is accepted after a single sampling. At the sixth step, however, the model’s first sampled candidate exhibits a coherency error. The adapter detects this issue and assigns a score below the threshold, triggering a resampling of the step. A subsequent candidate yields a correct reasoning step, which is then scored above the threshold and accepted by the adapter. The model subsequently continues its reasoning and arrives at the correct final answer. More analysis of LADA can be found in Appendix B.

5 Conclusion

In this paper, we propose a new framework, LADA, for continuous and rapid adaptation of black-box LLMs to unseen and changing domains at test time. LADA meta-trains an adapter on diverse tasks that cover different datasets and error combinations to learn transferable adaptation skills. At test time, the adapter uses a few target-domain positive-negative reasoning pairs for lightweight adaptation, then guides the black-box LLM step-by-step by selecting reliable reasoning steps and steering generation toward domain-appropriate trajectories without parameter updates or prior domain exposure. Experiments on datasets spanning multiple domains validate the effectiveness of LADA.

Limitations

While LADA demonstrates consistent improvements across multiple different target domains, our experiments mainly focus on reasoning-oriented benchmarks, including selected subsets of MMLU and MMLU-Pro. The effectiveness of LADA on more open-ended generation tasks, such as long-form writing, dialogue, or creative generation, remains to be further explored. We leave broader evaluation across more diverse task formats for future work.

Acknowledgments

This research was supported by the Jiangsu Science Foundation (BG2024036), the National Science Foundation of China (62622601, 62576093, 62125602, U24A20324, and 92464301), the New Cornerstone Science Foundation through the XPLOER PRIZE, and the Big Data Computing Center of Southeast University.

References

- Suhana Bedi, Yutong Liu, Lucy Orr-Ewing, Dev Dash, Sanmi Koyejo, Alison Callahan, Jason A. Fries, Michael Wornow, Akshay Swaminathan, Lisa So-leymani Lehmann, Hyo Jung Hong, Mehr Kashyap, Akash R. Chaurasia, Nirav R. Shah, Karandeep Singh, Troy Tazbaz, Arnold Milstein, Michael A. Pfeffer, and Nigam H. Shah. 2025. Testing and evaluation of health care applications of large language models: A systematic review. *JAMA*, 333(4):319–328.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 7432–7439, New York, NY.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint*, arXiv:2110.14168.
- DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, and 181 others. 2024. Deepseek-v3 technical report. *arXiv preprint*, arXiv:2412.19437.
- Yuyang Deng, Ni Zhao, and Xin Huang. 2023. Early chatgpt user portrait through the lens of data. In *Proceedings of the 2023 IEEE International Conference on Big Data*, pages 4770–4775, Sorrento, Italy.
- Mario Döbler, Robert A. Marsden, and Bin Yang. 2023. Robust mean teacher for continual and gradual test-time adaptation. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7704–7714, Vancouver, Canada.
- Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large language models for software engineering: Survey and open problems. In *Proceedings of the 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering*, pages 31–53, Melbourne, Australia.
- Yu Fan, Jingwei Ni, Jakob Merane, Yang Tian, Yoan Hermstrüwer, Yinya Huang, Mubashara Akhtar, Etienne Salimbeni, Florian Geering, Oliver Dreyer, Daniel Brunner, Markus Leippold, Mrinmaya Sachan, Alexander Stremitzer, Christoph Engel, Elliott Ash, and Joel Niklaus. 2026. Lexam: Benchmarking legal reasoning on 340 law exams. In *Proceedings of the 14th International Conference on Learning Representations*, Rio de Janeiro, Brazil.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, Sydney, Australia.
- Zeyu Gan and Yong Liu. 2025. Towards a theoretical understanding of synthetic data in llm post-training: A reverse-bottleneck perspective. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore.
- Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.
- Olga Golovneva, Moya Peng Chen, Spencer Poff, Martin Corredor, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Roscoe: A suite of metrics for scoring step-by-step reasoning. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda.
- Sachin Goyal, Mingjie Sun, Aditi Raghunathan, and J. Zico Kolter. 2022. Test time adaptation via conjugate pseudo-labels. In *Advances in Neural Information Processing Systems 35*, pages 6204–6218, New Orleans, LA.
- Moritz Hardt and Yu Sun. 2024. Test-time training on nearest neighbors for large language models. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language

- understanding. In *Proceedings of the 9th International Conference on Learning Representations*, Virtual Event.
- Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology*, 33(8):Article 220.
- Jinwu Hu, Zhitian Zhang, Guohao Chen, Xutao Wen, Chao Shuai, Wei Luo, Bin Xiao, Yuanqing Li, and Mingkui Tan. 2025a. Test-time learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 24823–24849, Vancouver, Canada.
- Yihao Hu, Congyu Qiao, Xin Geng, and Ning Xu. 2025b. Selective label enhancement learning for test-time adaptation. In *Proceedings of the 13th International Conference on Learning Representations*, pages 68984–69006, Singapore.
- Quzhe Huang, Mingxu Tao, Chen Zhang, Zhenwei An, Cong Jiang, Zhibin Chen, Zirui Wu, and Yansong Feng. 2023a. Lawyer llama technical report. *arXiv preprint*, arXiv:2305.15062.
- Yangsibo Huang, Daogao Liu, Zexuan Zhong, Weijia Shi, and Yin Tat Lee. 2023b. knn-adapter: Efficient domain adaptation for black-box language models. *arXiv preprint*, arXiv:2302.10879.
- Jonas Hübötter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently learning at test-time: Active fine-tuning of llms. In *Proceedings of the 13th International Conference on Learning Representations*, Singapore.
- Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L  lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, and 7 others. 2024. Mixtral of experts. *arXiv preprint*, arXiv:2401.04088.
- Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2567–2577, Hong Kong, China.
- Muhammad Khalifa, Lajanugen Logeswaran, Moon-tae Lee, Honglak Lee, and Lu Wang. 2023. Grace: Discriminator-guided chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15299–15328, Singapore.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems 35*, pages 22199–22213, New Orleans, LA.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, Koblenz, Germany.
- Jian Liang, Ran He, and Tieniu Tan. 2025. A comprehensive survey on test-time adaptation under distribution shifts. *International Journal of Computer Vision*, 133(1):31–64.
- Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. 2024. Tuning language models by proxy. In *Proceedings of the 1st Conference on Language Modeling*, Philadelphia, PA.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, LA.
- Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. 2022. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, pages 16888–16905, Baltimore, MD.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2023. Gpt-4 technical report. *arXiv preprint*, arXiv:2303.08774.
- Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. Comblm: Adapting black-box language models through small fine-tuned models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2961–2974, Singapore.
- Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the 2022 Conference on Health, Inference, and Learning*, pages 248–260, Virtual Event.
- Joaquin Qui  onero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2022. *Dataset Shift in Machine Learning*. MIT Press.

- Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 4463–4473, Hong Kong, China.
- Wenqi Shi, Ran Xu, Yuchen Zhuang, Yue Yu, Haotian Sun, Hang Wu, Carl Yang, and May Dongmei Wang. 2024. Medadapter: Efficient test-time adaptation of large language models towards medical reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22294–22314, Miami, FL.
- Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. 2023. Ecotta: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11920–11929, Vancouver, Canada.
- Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. 2024. Bbox-adapter: Lightweight adapting for black-box large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 47280–47304, Vienna, Austria.
- Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Man-sooreh Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. Large language models for data annotation and synthesis: A survey. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 930–957, Miami, FL.
- Dandan Wang and Shiqing Zhang. 2024. Large language models in medical and healthcare fields: Applications, advances, and challenges. *Artificial Intelligence Review*, 57(11):Article 299.
- Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno A. Olshausen, and Trevor Darrell. 2021. Tent: Fully test-time adaptation by entropy minimization. In *Proceedings of the 9th International Conference on Learning Representations*, Virtual Event.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of the 11th International Conference on Learning Representations*, Kigali, Rwanda.
- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems 37*, pages 95266–95290, Vancouver, Canada.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35*, pages 24824–24837, New Orleans, LA.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint*, arXiv:2505.09388.
- An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024a. Qwen2 technical report. *arXiv preprint*, arXiv:2407.10671.
- Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024b. Rlcd: Reinforcement learning from contrastive distillation for lm alignment. In *Proceedings of the 12th International Conference on Learning Representations*, Vienna, Austria.
- Hongzhou Yu, Tianhao Cheng, Yingwen Wang, Wen He, Qing Wang, Ying Cheng, Yuejie Zhang, Rui Feng, and Xiaobo Zhang. 2025. Finemedlm-o1: Enhancing medical knowledge reasoning ability of llm from supervised fine-tuning to test-time training. In *Proceedings of the 2nd Conference on Language Modeling*, Montreal, Canada.
- Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. In *Proceedings of the 9th International Conference on Learning Representations*, Virtual Event.
- Zhi Zhou, Jiang-Xin Shi, Peng-Xiao Song, Xiao-Wen Yang, Yi-Xuan Jin, Lan-Zhe Guo, and Yu-Feng Li. 2024. Lawgpt: A chinese legal knowledge-enhanced large language model. *arXiv preprint*, arXiv:2406.04614.

A Experimental Details

A.1 Additional Dataset Details

This section presents a detailed description of the evaluation benchmarks and meta-training datasets used in our experiments. We first describe the ten subsets of MMLU and the three subsets of MMLU-Pro, which are treated as dynamically changing target domains and cover quantitative reasoning, logical reasoning, and domain-specific technical problem solving. We then introduce the three datasets used to construct the meta-training data, namely GSM8K, StrategyQA, and ReClor, and describe their task settings and the forms of reasoning they require. The key properties of these datasets are summarized below.

- MMLU ¹ (Hendrycks et al., 2021) is a multitask benchmark designed to evaluate knowledge and reasoning across a wide range of subject areas. In our experiments, we use ten reasoning-related subsets, comprising a total of 1.8K examples: high school physics (Phys.), high school statistics (Stats.), high school microeconomics (Microecon.), high school computer science (CS), machine learning (ML), formal logic (Form. Log.), logical fallacies (Log. Fall.), professional accounting (Prof. Acc.), professional medicine (Prof. Med.), and electrical engineering (EE).
- MMLU-Pro ² (Wang et al., 2024) is an extended benchmark that builds on MMLU with a stronger focus on expert-level reasoning and reduced shortcut cues. It evaluates model performance across professional and applied domains using more complex question structures. In our experiments, we use three subsets from MMLU-Pro, comprising a total of 2K examples: computer science (CS), philosophy (Phil.), and law.
- GSM8K ³ (Cobbe et al., 2021) is a grade-school mathematics word problem dataset with 8.5K examples. Each problem typically requires two to eight reasoning steps based on basic arithmetic operations, and solutions are provided in natural language by human annotators with quality control. The dataset evaluates a model’s ability to perform multi-step numerical reasoning and to produce coherent step-by-step solutions.
- StrategyQA ⁴ (Geva et al., 2021) is a yes/no

question answering dataset with 2.3K examples. The questions require implicit multi-step reasoning, where intermediate reasoning steps are not annotated and must be inferred by the model. The dataset evaluates a model’s ability to carry out implicit reasoning across multiple facts and assumptions.

- ReClor ⁵ (Yu et al., 2020) is a reading comprehension dataset with 6.1K multiple-choice questions collected from standardized graduate admission examinations. Each example consists of a short argumentative passage and a question with four answer options, requiring logical reasoning over the argument structure. The dataset evaluates a model’s ability to perform passage-level logical reasoning in natural language.

A.2 Additional Task Construction Details

Before synthesizing data for meta-training, we obtain step-by-step reasoning traces for the training splits of the relevant datasets. GSM8K directly provides annotated step-by-step reasoning traces, whereas for StrategyQA and ReClor, which lack such annotations, we employ DeepSeek-V3 (DeepSeek-AI et al., 2024) to generate corresponding traces and discard erroneous samples.

Regarding error types in LLM reasoning, Golovneva et al. (2023) categorize them into nine types: grammar, factuality, hallucination, redundancy, repetition, missing step, coherency, commonsense, and arithmetic. We select seven of these types, excluding grammar and arithmetic, and the following provides their descriptions along with synthesized examples. The sentences highlighted in red represent erroneous reasoning steps corresponding to each error type.

Factuality

Description: Information about object (i.e. quantity, characteristics) or a personal named entity does not match information provided in the question.

Example: Q: Mrs. Snyder used to spend 40% of her monthly income on rent and utilities. Her salary was recently increased by \$600 so now her rent and utilities only amount to 25% of her monthly income. How much was her previous monthly income? A: Let’s think step by step. Let her previous monthly income be

¹<https://huggingface.co/datasets/cais/mmlu>

²<https://huggingface.co/datasets/TIGER-Lab/MMLU-Pro>

³<https://huggingface.co/datasets/openai/gsm8k>

⁴<https://huggingface.co/datasets/wics/>

⁵[strategy-qa](https://huggingface.co/datasets/metaeval/reclor)

⁵<https://huggingface.co/datasets/metaeval/reclor>

p. The cost of her rent and utilities was 40% of p which is $2p/5$. Her income rose by 500 dollars, making it p plus 500.

Hallucination

Description: Information is not provided in the problem statement and is irrelevant or wrong.

Example: Q: A concert ticket costs \$40. Mr. Benson bought 12 tickets and received a 5% discount for every ticket bought that exceeds 10. How much did Mr. Benson pay in all? A: Let's think step by step. Mr. Benson had a 5% discount for each of the $12 - 10 = 2$ tickets. So, all 12 tickets received a discount of 5 dollars each, even though only the extra tickets should qualify.

Redundancy

Description: Explanation contains redundant information, which even though might be factual, is not required to answer the question.

Example: Q: Hector purchased a container of gumballs. He gave 4 to Todd, then he gave twice as many as he had given Todd to Alisha, and then he gave 5 less than four times as many to Bobby as he had given to Alisha. If Hector had 6 gumballs remaining, what is the total number of gumballs that Hector purchased? A: Let's think step by step. Hector gave 4 gumballs to Todd. Since Alisha received twice as many as Todd, she got 8 gumballs, which is a common pattern in many doubling problems and often appears in similar arithmetic word problems.

Repetition

Description: Step paraphrases information already mentioned in previous reasoning steps.

Example: Q: Boris has 24 books and he donates a fourth of his books to the library. Cameron has 30 books and he donates a third of his books to the library. After donating their books, how many books in total do Boris and Cameron have together? A: Let's think step by step. Boris donates $24 / 4 = 6$ books. Then Boris has a total of $24 - 6 = 18$ books. Cameron donates $30 / 3 = 10$ books. Cameron donates 10 books out of his 30, which is a third of his books.

Missing step

Description: The content of the generated rea-

soning is incomplete and lacks required information to produce the correct answer. If these missing steps are added, the model would produce the correct answer.

Example: Q: A vendor bought apples at 2 for \$3 and plans to sell them at 5 for \$10. He also bought 3 oranges for \$2.70 and plans to sell them at \$1 each. How much is his profit if he sells 5 apples and 5 oranges? A: Let's think step by step. A vendor bought each apple for $\$3/2 = \$1.50$. His profit for each apple is $\$10 - \$1.50 = \$8.50$.

Coherency

Description: Steps contradict each other or do not follow a cohesive story.

Example: Q: Otto's local kitchen supply store charges \$5.00 for the first knife that you need sharpened. They charge \$4.00 for the next 3 knives and \$3.00 for any knife after that. If Otto has 9 knives that need to be sharpened, how much will it cost to sharpen his knives? A: Let's think step by step. Knives 2, 3 & 4 will cost \$4.00 each to sharpen so that's $3 * 4 = \$12.00$. Knives 5-9 will cost \$3.00 each to sharpen so that's $5 * 3 = \$15.00$. Adding only the \$12.00 and \$15.00 for knives 2-9, the total cost is $12 + 15 = \$27.00$

Commonsense

Description: Model lacks relations that should be known from general knowledge about the world (i.e. how to compute velocity, how many inches in one foot, all ducks are birds, etc.).

Example: Q: Molly got a bike for her thirteenth birthday. She rode her bike 3 miles a day, every day, until she turned 16. How many miles did Molly ride on her bike? A: Let's think step by step. Molly rode for $16 - 13 = 3$ years. That means she rode over a period of 4 full years, from her 13th birthday up to (but not beyond) her 16th birthday.

The prompts used for data synthesis and post-processing are presented in Figure 3.

B Further Analysis

B.1 Evaluation on More Datasets

We conduct additional experiments on five larger and more diverse datasets beyond MMLU: MedMCQA (Pal et al., 2022) and PubMedQA (Jin

Table 5: Reasoning accuracy (%) of different methods, reported as mean $\pm$ standard deviation across diverse datasets spanning different domains.

Methods	Model	MedMCQA	PubMedQA	PIQA	Social IQa	LEXam
ZERO-SHOT-COT	Qwen2-7B	45.74 $\pm$ 1.16	63.85 $\pm$ 1.42	81.74 $\pm$ 0.72	70.21 $\pm$ 1.49	22.96 $\pm$ 1.79
COT-PROMPTING		48.91 $\pm$ 0.78	66.40 $\pm$ 1.14	83.84 $\pm$ 0.60	71.80 $\pm$ 1.16	24.53 $\pm$ 1.31
SELF-CONSISTENCY		53.31 $\pm$ 0.59	68.98 $\pm$ 0.95	84.76 $\pm$ 0.43	73.27 $\pm$ 0.73	26.61 $\pm$ 0.94
BBOX-ADAPTER		52.14 $\pm$ 1.26	68.03 $\pm$ 1.27	84.28 $\pm$ 0.67	72.78 $\pm$ 1.07	25.92 $\pm$ 1.53
LADA		55.19 $\pm$0.68	70.55 $\pm$1.22	85.83 $\pm$0.52	75.73 $\pm$1.19	28.37 $\pm$1.24
ZERO-SHOT-COT	Mixtral-8x7B	48.29 $\pm$ 1.47	67.47 $\pm$ 1.52	79.95 $\pm$ 1.15	64.73 $\pm$ 1.38	20.31 $\pm$ 2.14
COT-PROMPTING		51.80 $\pm$ 1.25	69.20 $\pm$ 1.26	81.33 $\pm$ 0.94	66.68 $\pm$ 1.03	21.99 $\pm$ 1.15
SELF-CONSISTENCY		54.97 $\pm$ 0.79	72.79 $\pm$ 0.84	83.05 $\pm$ 0.61	67.81 $\pm$ 0.71	24.56 $\pm$ 0.96
BBOX-ADAPTER		53.76 $\pm$ 1.45	71.87 $\pm$ 1.35	82.57 $\pm$ 0.87	67.46 $\pm$ 0.88	23.15 $\pm$ 1.28
LADA		57.14 $\pm$1.17	75.07 $\pm$1.03	84.46 $\pm$0.53	69.89 $\pm$0.75	26.43 $\pm$0.84

Table 6: Reasoning accuracy (%) and significance analysis on MMLU subsets using Mixtral-8x7B-Instruct-v0.1. $\bullet/\circ$ indicates whether the performance of LADA is statistically superior/inferior to BBox-Adapter on the corresponding subset under a pairwise t-test at the 0.05 significance level.

Methods	Phys.	Stats.	Microecon.	CS	ML
BBOX-ADAPTER	54.68 $\pm$ 1.38 $\bullet$	63.40 $\pm$ 0.77 $\bullet$	78.82 $\pm$ 0.71 $\bullet$	77.84 $\pm$ 0.89 $\bullet$	48.45 $\pm$ 0.91 $\bullet$
LADA	56.83 $\pm$ 1.05	65.54 $\pm$ 1.16	80.34 $\pm$ 0.49	79.50 $\pm$ 0.68	49.69 $\pm$ 0.42
Methods	Form. Log.	Log. Fall.	Prof. Acc.	Prof. Med.	EE
BBOX-ADAPTER	51.23 $\pm$ 1.28 $\bullet$	76.53 $\pm$ 0.87 $\bullet$	55.04 $\pm$ 0.71 $\bullet$	79.35 $\pm$ 0.68 $\bullet$	67.37 $\pm$ 0.93 $\bullet$
LADA	53.89 $\pm$ 0.86	77.98 $\pm$ 0.58	56.31 $\pm$ 0.59	80.55 $\pm$ 0.41	68.49 $\pm$ 0.71

et al., 2019) for medical reasoning, PIQA (Bisk et al., 2020) for physical commonsense reasoning, Social IQa (Sap et al., 2019) for social interaction reasoning, and LEXam (Fan et al., 2026) for legal reasoning. We evaluate all methods with Qwen2-7B-Instruct and Mixtral-8x7B-Instruct-v0.1 over three runs, and report the results in Tables 5. The results show that LADA consistently improves performance across these datasets and backbone models, demonstrating that its effectiveness is not limited to MMLU subsets but generalizes to broader domains and reasoning tasks.

B.2 Significance Analysis

We conduct a more rigorous significance analysis between LADA and the training-based baseline, BBOX-ADAPTER. Specifically, on Mixtral-8x7B-Instruct-v0.1, we rerun both BBOX-ADAPTER and LADA with 10 random seeds across the MMLU subsets and perform pairwise t-tests at the 0.05 significance level. As shown in Table 6, LADA significantly outperforms BBOX-ADAPTER across

the evaluated subsets, indicating that the observed gains are statistically reliable rather than caused by random variation.

B.3 Synthetic Data Analysis

According to Gan and Liu (2025), the generalization error upper bound of a model trained with synthetic data can be decomposed into two parts, namely the distributions’ divergence and the generalization error with respect to synthetic data. Their reverse-bottleneck analysis suggests that, while synthetic data may introduce some distributional mismatch, the information gain contributed by the generative model can tighten the generalization bound. In practice, the bound is closely related to the generative model’s capability and to the method used to construct and curate synthetic data. To investigate the impact of these factors, in addition to DeepSeek-V3, we consider GPT-5 and Qwen3-8B (Yang et al., 2025) as alternative data generation models. We also conduct an ablation study on the filtering process applied to the synthesized

Table 7: Reasoning accuracy (%) of LADA across different target domains using adapters trained on synthetic data generated by different generative models, reported as mean $\pm$ standard deviation.

Models	Filter	Phys.	Stats.	Log. Fall.
DeepSeek-V3	✓	57.17 $\pm$ 1.38	64.35 $\pm$ 1.39	75.26 $\pm$ 0.29
GPT-5	✓	57.40 $\pm$ 1.67	64.51 $\pm$ 1.17	75.46 $\pm$ 0.61
Qwen3-8B	✓	56.95 $\pm$ 1.15	64.20 $\pm$ 1.34	75.26 $\pm$ 0.71
DeepSeek-V3	✗	56.51 $\pm$ 1.91	63.89 $\pm$ 1.85	74.64 $\pm$ 1.54

data. The results on three domains using Qwen2-7B-Instruct are shown in Table 7. We observe that the effectiveness of the synthetic data is robust to the choice of the generative model, and the performance remains strong even when using Qwen3-8B, suggesting that the information gains are not tied to a single oracle. Moreover, filtering consistently improves performance by reducing noise in the synthesized data, thereby enhancing its effectiveness for meta-training.

B.4 Discussion on the Practicality of Supervision in LADA

The core objective of LADA is to correct reasoning trajectories under distribution shift by identifying unreliable intermediate steps before they propagate downstream errors. Achieving this requires supervision that is aligned with the granularity at which reasoning failures occur. Unlike answer-level feedback, which provides only coarse signals and leaves the location of errors ambiguous, step-level positive-negative supervision directly isolates whether a candidate reasoning step is appropriate given the current context (Cobbe et al., 2021; Khalifa et al., 2023). This design choice allows the adapter to focus on local reliability signals, which is important for identifying faulty steps early before errors propagate.

Importantly, step-level contrastive signals can arise naturally in existing LLM-based systems. Multi-sample decoding procedures often produce diverse candidate reasoning steps, some of which lead to successful solutions while others derail the reasoning process (Wang et al., 2023). These outcomes implicitly define positive-negative contrasts at intermediate steps. Similarly, user interactions such as corrections, clarifications, or flagged hallucinations frequently target specific parts of a reasoning chain rather than the entire response. In addition, as demonstrated in our experiments, such

pairs can be synthesized offline using off-the-shelf language models and filtered through automatic judging mechanisms. These signals can be collected offline to form a retrieval pool covering a broad range of scenarios, and queried at test time as needed.

Last but not least, rather than relying on extensive domain-specific resources, LADA leverages a few positive-negative information to facilitate adaptation of a meta-trained adapter at test time. This design choice reflects a balance between adaptability and practicality, and supports the feasibility of the proposed setting under real-world deployment conditions.

C Potential Risks

We did not identify any major risks associated with our work. One limited potential risk is that noisy or biased target-domain adaptation examples may cause the adapter to prefer unreliable reasoning steps, leading to incorrect outputs. This can be mitigated by using quality-controlled adaptation examples and applying standard safety checks in practical deployment.

D Use of Scientific Artifacts

The MMLU, MMLU-Pro, GSM8K, and StrategyQA datasets are released under the MIT License. The ReClor dataset does not provide an explicit license. For models, the Qwen models are released under the Apache-2.0 License, and the Mistral model is also released under the Apache-2.0 License.

We use all existing artifacts in accordance with their intended purposes and applicable license terms. Any artifacts newly created in this work are intended solely for research purposes. Our study relies exclusively on publicly available datasets, and we do not collect any private or personally identifiable information ourselves. For data privacy and anonymization, we defer to the release protocols and preprocessing procedures established by the original dataset providers.

Prompt for Data Synthesis

You are a data synthesis assistant.

The input is a correct reasoning process, which can be formalized as the sequence "Q: <question>

A: Let's think step by step. $r_{start} \cdots r_i \cdots r_{end}$ ", where each r denotes a reasoning step.

The type of error you need to synthesize is: "Coherency", which refers to: "Steps contradict each other or do not follow a cohesive story".

First, you need to select a reasoning step r_i from the reasoning steps that is prone to the "Coherency" error during LLM reasoning. Then:

- Generate a correct version $right(r_i)$ by rewriting the original step r_i in a different form, while preserving its original meaning.
- Generate a corresponding faulty version $wrong(r_i)$ by introducing the error into r_i .

The faulty step should resemble realistic reasoning mistakes commonly made by large language models, rather than artificial or exaggerated errors.

Next, you need to construct a positive and a negative sample. You only need to include the reasoning steps up to and including the synthesized correct step $right(r_i)$ (for the positive sample), or up to and including the synthesized incorrect step $wrong(r_i)$ (for the negative sample). Do not include the rest of the reasoning chain. In terms of output format, please return the positive and negative samples in a list. Each sample should be a string, and each reasoning step should occupy one line, as follows:

```
[ "Q: <question> \n A: Let's think step by step. \n  $r_{start}$  \n  $\cdots$   $right(r_i)$ ",  
  "Q: <question> \n A: Let's think step by step. \n  $r_{start}$  \n  $\cdots$   $wrong(r_i)$ " ]
```

Finally, check carefully whether the data synthesis and output format meet the specified requirements, and briefly explain the reasons.

Your task:

Prompt for Post-processing

You are a judge model for filtering synthesized reasoning samples.

The input consists of 10 pairs of positive and negative reasoning samples, where the positive and negative samples differ only in the final reasoning step. The negative samples simulate the "Coherency" error in the final reasoning step, defined as "Steps that contradict each other or do not follow a cohesive line of reasoning."

Your task is to select the best subset of 6 pairs from the 10 candidates based on three evaluation criteria: quality, diversity, and fidelity:

- Quality: Positive samples must be logically correct and coherent, and negative samples must contain a plausible commonsense reasoning error that is not trivial, random, or nonsensical.
- Diversity: Prefer pairs that cover diverse failure modes, and avoid many highly similar cases.
- Fidelity: Negative reasoning should resemble realistic LLM mistakes and remain subtle, fluent, contextually coherent, and believable.

Output Requirement:

- Keep the same output format as the input.
- Only remove low-quality ones.
- Do not add explanations or comments.
- Carefully verify formatting before returning the final filtered set.

Your task:

Figure 3: Prompt used for data synthesis and filtering.

Q: A block is dragged along a table and experiences a frictional force, f , that opposes its movement. The force exerted on the block by the table is

Choice: A. zero, B. parallel to the table, C. perpendicular to the table, D. neither parallel nor perpendicular to the table

A: Let's think step by step.

Step 1: The force of friction, f , is a force that opposes the motion of an object. (0.87, →)

Step 2: In this case, the frictional force, f , is acting in the opposite direction of the block's movement. (0.96, →)

Step 3: The normal force, which is the force exerted by the table on the block, is always perpendicular to the surface of contact. (0.90, →)

Step 4: Therefore, the normal force is perpendicular to the table. (0.89, →)

Step 5: However, the force of friction is not the same as the normal force. (0.72, →)

Step 6: The force exerted on the block by the table is the normal force, which is perpendicular to the table, not the frictional force. (-0.69, ☹) (Coherency Error)

Step 6: The force exerted on the block by the table is the resultant of the normal force and the frictional force, so it is neither parallel nor perpendicular to the table. (0.79, →)

Answer: D.

Q: A student is recording a song on her computer. When the recording is finished, she saves a copy on her computer. The student notices that the saved copy is of lower sound quality than the original recording. Which of the following could be a possible explanation for the difference in sound quality?

Choice: A. The song was saved using fewer bits per second than the original song. B. The song was saved using more bits per second than the original song. C. The song was saved using a lossless compression technique. D. Some information is lost every time a file is saved from one location on a computer to another location.

A: Let's think step by step.

Step 1: To determine the cause of the difference in sound quality, we need to consider how sound quality is affected when saving files on a computer. (0.97, →)

Step 2: When a file is saved, it is usually compressed to reduce the size of the file. This compression process can result in some information being lost, which may affect the sound quality. (0.93, →)

Step 3: Since the saved copy has lower sound quality than the original recording, it suggests that some information was indeed lost during the saving process. (-0.52, ☹) (Repetition Error)

Step 3: Saving with fewer bits per second reduces the data used to represent the sound, so the quality decreases, meaning it was likely saved with a lower bit rate. (0.93, →)

Answer: D.

Q: Sussman Co. prepared cash-basis financial statements for the month ended January 31. A summary of Sussman's January activities follows: Credit sales of \$5600. Collections of \$1900 relating to January credit sales. Accrued salaries of \$1200. By what amount will Sussman's cash-basis income for the month ended January 31 increase as a result of restating these activities to the accrual basis of accounting?

Choice: A. \$2,500, B. \$3,700, C. \$4,400, D. \$4,900

A: Let's think step by step.

Step 1: Calculate total credit sales for January, which is \$5,600. (0.87, →)

Step 2: Deduct collections from January credit sales (\$1,900) from total credit sales to find the remaining balance to be collected under accrual basis, which is \$3,700 (\$5,600 - \$1,900). (0.94, →)

Step 3: Add accrued salaries (\$1,200) to the remaining balance to account for expenses not yet paid, resulting in an increase of \$4,900 (\$3,700 + \$1,200). (-0.93, ☹) (Commonsense Error)

Step 3: Subtract accrued salaries of \$1,200 (an expense under accrual accounting) from the \$3,700 revenue increase to get the net income increase of \$2,500 (\$3,700 - \$1,200). (0.89, →)

Answer: A.

Figure 4: Case study of LADA, where parentheses show (score, action); → denotes moving to the next step, and ☹ denotes regenerating the current step.