ARVEC: Adaptive Reasoning Model with Vision Understanding and Executable Code

Anonymous ACL submission

Abstract

Large Reasoning Models (LRMs) often suffer from the "over-thinking" problem, generating unnecessarily long reasoning on simple tasks. Some strategies have been proposed to mitigate this issue, such as length penalties or routing mechanisms, but they are typically heuristic and task-specific, lacking a general framework for adaptive reasoning. In this paper, we present ARVEC, a unified model that adaptively balances reasoning performance and efficiency across multiple formats through a reinforcement learning framework augmented with length-aware optimization. Beyond conventional natural language inference, ARVEC integrates vision understanding, extending its applicability to multimodal. Moreover, ARVEC integrates executable code into reasoning, enabling substantial reductions in token cost while preserving task performance compared to long CoT. Experiments demonstrate that **ARVEC achieves performance on par with traditional reasoning models trained with GRPO, while reducing token usage by over 70% on average.** We further conduct extensive analyses to validate the effectiveness of ARVEC and the soundness of its design. All the code and resources will be released for future research.

1 Introduction

Large Reasoning Models (LRMs) (Jaech et al., 2024; Guo et al., 2025), trained using Reinforcement Learning with Verifiable Rewards (RLVR), have emerged as a leading paradigm for enhancing the reasoning capabilities of language models. Their strong performance is largely attributed to the generous token budget allowed during inference (i.e., long chain-of-thought; long CoT), which enables the use of advanced reasoning strategies such as self-reflection and self-correction (Guo et al., 2025). However, LRMs are not universal "master keys," as they tend to apply long CoT reasoning indiscriminately across tasks, leading to the "over-thinking" problem (Sui et al., 2025; Yue et al., 2025), and in some cases, even causing performance degradation (Wu et al., 2024; Fan et al., 2025). While recent approaches attempt to mitigate the "over-thinking" problem, most operate from a single perspective: length control. These methods require the model to reason within a fixed token budget or use a separately trained model constrained to specific reasoning lengths (Aggarwal and Welleck, 2025; Alomrani et al., 2025; Zhu and Li, 2025). Such strategies assume prior knowledge of the task to predefine the appropriate reasoning length. Moreover, they lack the flexibility to switch reasoning formats dynamically across tasks, since manually selecting models or adjusting token limits introduces unnecessary human intervention and hinders generalizability.

In real-world applications, models are often required to balance both low-latency responses and complex reasoning. For example, for relatively simple questions, models are expected to provide direct answers without unnecessary deliberation. In contrast, for more challenging questions, models are expected to generate reliable answers through comprehensive reasoning. This raises a critical research question: *How can we equip LRMs with the ability to perform adaptive reasoning based on task demands and input complexity?*

Hybrid reasoning strategies (Lou et al., 2025; Zhang et al., 2025; Wu et al., 2025) provide viable approaches to balance performance and efficiency. For example, ARM (Wu et al., 2025) introduces four distinct reasoning formats and encourages the use of more efficient formats rather than relying exclusively on long CoT. However, these approaches primarily focus on incorporating shorter reasoning formats to mitigate the substantial token cost of long CoT, without explicitly leveraging length-aware reward signals, which limits their effectiveness in reducing token usage. Furthermore, existing adaptive reasoning models are

designed for plain-text inputs and lack the ability to handle visual information. In parallel, recent advances in coding capabilities and their integration into reasoning (Chen et al., 2023; Wang et al., 2024) have highlighted code execution during inference as a powerful mechanism for enhancing reasoning. Incorporating executable code enables models to perform verifiable computations, manage complex logic, and interact with external tools, thereby extending their reasoning capacity well beyond traditional text-based methods.

In this paper, as shown in Figure 1, we propose ARVEC—an **A**daptive **R**easoning model with **V**ision understanding and **E**xecutable **C**ode. ARVEC supports five reasoning formats: *Direct Answer*, *Short CoT*, *Code-Text*, *Code-Exec*, and *Long CoT*. Given a task, ARVEC adaptively selects one of these formats for reasoning. Among them, *Direct Answer*, *Short CoT*, *Code-Text*, and *Long CoT* are based on natural language inference, while *Code-Exec* leverages an external Python interpreter to execute code and return the result as the final answer. This capability empowers the model to solve complex reasoning tasks that require high precision, thereby extending its reach beyond conventional natural language inference.

To enable adaptive reasoning, we curate a high-quality multimodal dataset of 15.1K instances spanning all five reasoning formats, providing a strong foundation for cold-start through supervised fine-tuning (SFT). However, since SFT only teaches the use of different reasoning formats without enabling adaptive selection, we turn to reinforcement learning (RL). Yet conventional Group Relative Policy Optimization (GRPO) (Shao et al., 2024) suffers from format collapse (Wu et al., 2025), where models tend to over-rely on the reasoning format with the highest accuracy. To overcome this issue, we propose GRPO-ALP, a novel variant of GRPO designed to promote adaptive reasoning by encouraging exploration of less frequent formats with relatively lower accuracy during training. While related to Ada-GRPO (Wu et al., 2025), which also addresses format imbalance, our approach goes further by explicitly incorporating response length into the optimization objective. This enables GRPO-ALP not only to achieve balanced format selection but also to guide the model in trading off reasoning accuracy against computational efficiency. In comparison to traditional LRMs, ARVEC demonstrates improved efficiency on straightforward, intuitive tasks while preserving competitive performance on reasoning-intensive tasks.

In summary, our contributions are as follows: *1)* ARVEC achieves a strong balance between performance and efficiency, reducing token usage by over 70% on average across six in-domain and six out-of-domain datasets, spanning both text and multimodal settings. *2)* ARVEC first incorporates executable code into adaptive reasoning, improving code reasoning accuracy and providing a more efficient alternative to long CoT reasoning. *3)* We conduct systematic experiments to validate the design of ARVEC, including ablation studies on its components as well as analyses of length penalty strength and backbone model initialization.

2 Related Work

2.1 Overthinking

Capable of performing elaborative reasoning at inference time, Large Reasoning Models (LRMs) achieve substantial performance gains across a wide range of reasoning benchmarks (Jaech et al., 2024; Guo et al., 2025). However, not all tasks require such an intensive reasoning style, and prior studies have shown that LRMs often incur unnecessary token overhead on relatively simple tasks (Chen et al., 2024). More critically, this inefficiency not only leads to computational waste but also undermines the model's overall effectiveness, as it may produce misleading or confusing content, such as repeated reflections, unwarranted self-doubt, or overconfident simulation (Fan et al., 2025; Cuadron et al., 2025), which could have been avoided with more direct responses (Alomrani et al., 2025). This challenge is particularly troublesome because the very strength of LRMs lies in their extended reasoning steps, making their resolution appear paradoxical (Sui et al., 2025).

2.2 Controllable and Adaptive Reasoning

To improve the efficiency of LRMs, previous work has incorporated length constraints into the reward function (Aggarwal and Welleck, 2025; Luo et al., 2025b; Hou et al., 2025) under reinforcement learning optimization, encouraging shorter responses while maintaining correctness. While this strategy can indeed reduce response length at inference time, it has a key limitation: it relies on accurately estimating the task's complexity. If the estimation is imprecise, the model's performance may be significantly affected (Wu et al., 2025). Another line of research focuses on adaptive reasoning, which

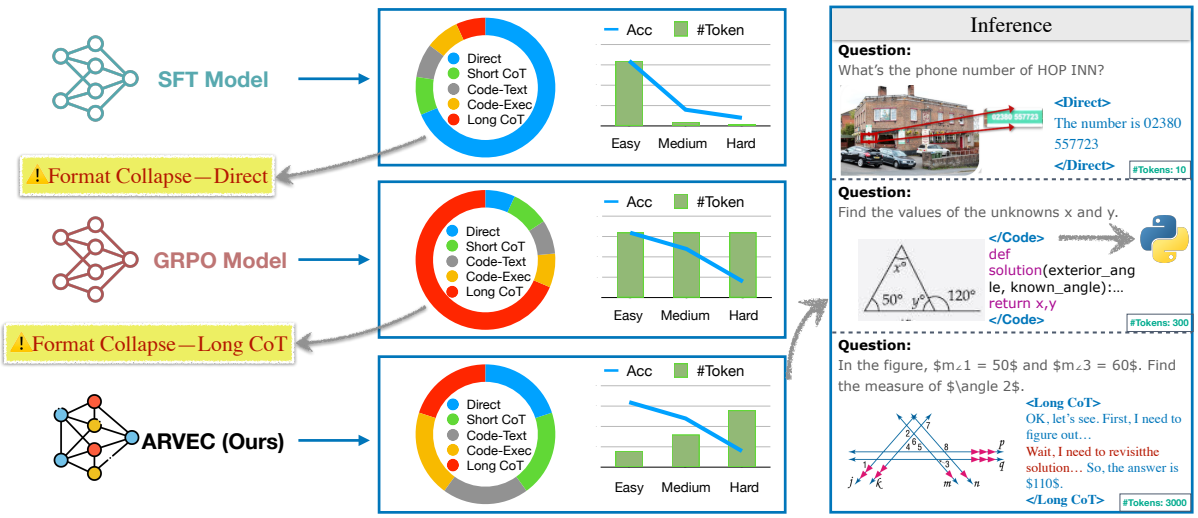

Figure 1: Comparison of reasoning behaviors across models. Unlike the SFT and GRPO models, which consistently rely on a single reasoning format, ARVEC exhibits adaptability by selecting reasoning formats.

allows models to switch among different reasoning formats, such as short CoT for easy tasks and long CoT for more complex ones (Zhang et al., 2025; Luo et al., 2025a). However, this strategy also faces two major limitations: the restricted set of reasoning styles available for handling diverse tasks, and the reliance on task-specific supervision during training without incorporating length awareness into reasoning control. To address these challenges, we propose ARVEC, a unified model capable of flexibly switching among five reasoning styles for different tasks, while explicitly incorporating length awareness into the reasoning process.

3 Methodology

In this paper, we introduce ARVEC, a multi-modal reasoning model that adaptively adjusts its reasoning formats based on tasks while maintaining consistent performance compared to vanilla reasoning models. To train ARVEC, we adopt the following training pipeline: *1)* Data Construction, building multi-modal QA datasets with diverse reasoning formats; *2)* Supervised Fine-tuning (SFT), teaching the model to reason using different formats; *3)* Length-aware Reinforcement Learning (RL), enabling the model to select the most suitable reasoning format to balance performance and token efficiency; and *4)* Inference, applying the trained model for adaptive reasoning across tasks.

3.1 Data Construction

Our training dataset consists of both a plain-text reasoning part and a multi-modal reasoning part. For the plain-text part, we adopt the AQuA-Rat dataset (Ling et al., 2017) used in ARM (Wu et al., 2025). To activate the multi-modal reasoning capability of ARVEC, we further collect data from VisualWebInstruct (Jia et al., 2025) to extend the modality to vision. While *Direct Answer* and *Short CoT* rationales are directly provided within the dataset, we augment it with *Code-Text*, *Code-Exec*, and *Long CoT* rationales using GPT-4o (OpenAI, 2024) and Doubao-Pro-1.6-Thinking (Seed et al., 2025). Moreover, all rationales in the code reasoning format are verified for executability using an external interpreter. This enables ARVEC to support executable code generation during inference. To ensure high-quality supervision, we filter out instances whose rationales yield incorrect answers, resulting in a curated dataset of 15.1K distinctive training examples spanning all reasoning formats.

For the reinforcement learning stage, the flexibility of RLVR allows us to leverage datasets that contain only questions and answers. Specifically, we incorporate three text-only datasets: CommonSenseQA (CSQA) (Talmor et al., 2019), GSM8K (Cobbe et al., 2021), and AIME 1987-2023 (AIME), covering a spectrum from simple commonsense reasoning to complex mathematical problem solving. In addition, we include three multimodal datasets, including MME RealWorld (MMEWorld) (Zhang et al., 2024), Geometry3K (GEO3K) (Lu et al., 2021), and MMK12 (Meng et al., 2025), covering tasks from perception-focused to advanced visual reasoning, ensuring a broad range of reasoning difficulties. The detailed dataset distribution is provided in Table 1. We provide further details of dataset collection and

Table 1: Data distribution of training set in two stages.

Dataset	Modality	Format	Size
Supervised Finetuning			
AQuA-Rat	Text	Multiple-Choice	3.0K
	Text	Free-form	7.8K
VisualWebInstruct	Text&Vision	Multiple-Choice	1.9K
	Text&Vision	Free-form	2.2K
Total			15.1K
Reinforcement Learning			
CommonsenseQA	Text	Multiple-Choice	4.9K
GSM8K	Text	Free-form	7.4K
AIME	Text	Free-form	0.9K
Geometry3K	Text&Vision	Multiple-Choice	2.0K
MME RealWorld	Text&Vision	Free-form	4.0K
MMK12	Text&Vision	Free-form	4.0K
Total			23.2K

preprocessing in Appendix A.1.

3.2 Supervised Fine-Tuning

To enable the model to quickly grasp the five reasoning formats and facilitate parsing in the subsequent reinforcement learning stage, we adopt SFT as our cold-start strategy. Specifically, we train the backbone model on the dataset constructed in Stage 1 using the standard auto-regressive objective, formally defined as:

$$p(Y \mid X) = \prod_{i=1}^{|Y|} p_\theta(y_i \mid X_{\mathrm{t},<i}), \tag{1}$$

where θ denotes the trainable parameters, and X and Y represent the input and output token sequences, respectively. However, the model trained via SFT does not learn to select reasoning formats adaptively based on task requirements. The SFT model often relies excessively on direct reasoning across tasks, rather than engaging in informed decision-making (see Appendix A.2).

3.3 Length-aware Reinforcement Learning

Reinforcement Learning has demonstrated its effectiveness in enhancing the reasoning capabilities of language models (Guo et al., 2025), often yielding better generalization than SFT. However, traditional RL algorithms tend to constrain exploration due to reward functions focused on a single objective, such as accuracy. While this design may be acceptable in certain contexts, it can lead to the format collapse problem when applying Group Relative Policy Optimization (GRPO) (Wu et al., 2025). Specifically, format collapse refers to GRPO disproportionately favoring the reasoning format with the highest accuracy, thereby discouraging exploration of other formats, which results in excessive and unnecessary token usage. For example, applying *Code-Exec* indiscriminately for all tasks is inefficient, especially for perception-oriented tasks that do not require advanced reasoning.

To address this limitation, we propose GRPO-ALP, a novel extension of GRPO that promotes diversity in reasoning formats to mitigate format collapse, while incorporating a length-aware reward penalty. By explicitly integrating response length into the optimization objective, GRPO-ALP enables the model to balance reasoning accuracy with computational efficiency, thereby supporting more adaptive and cost-effective format selection.

Following ARM (Wu et al., 2025), GRPO-ALP incorporates a format encouragement factor to amplify the reward r_i corresponding to the response o_i for less frequently sampled reasoning formats, preventing their disappearance and ensuring adequate learning. Formally, we scale the original reward r_i (a binary value indicating accuracy, either 0 or 1) to r'_i as follows:

$$r'_i = \alpha_i(t) \cdot r_i = \frac{G}{F(o_i)} \cdot r_i, \tag{2}$$

where $F(o_i)$ denotes the number of times the reasoning format corresponding to o_i appears within its group O. In this way, underrepresented formats receive proportionally larger rewards, encouraging balanced exploration and mitigating the risk of format collapse.

In addition to encouraging the exploration of less frequent reasoning formats, GRPO-ALP further applies a length penalty to longer formats. This encourages more concise reasoning when possible and allows shorter, efficient formats to stand out when they are sufficient to solve the problem. Formally, we further scale the reward r'_i to r''_i by:

$$r''_i = \beta_i(t) \cdot r'_i, \tag{3}$$

$$\beta_i = \exp\left(-\lambda \cdot \frac{l_i - l_{\min}}{l_{\max} - l_{\min} + \epsilon}\right), \tag{4}$$

where l_i denotes the length of the response associated with o_i, and $l_{\min}$ and $l_{\max}$ represent the minimum and maximum response lengths within the group, respectively. λ is a hyperparameter that controls the strength of the length penalty, and ϵ is a small constant added to avoid division by zero when $l_{\max} = l_{\min}$.

To further stabilize the reward signal throughout training, we apply a cosine decay sched-

ule (Loshchilov and Hutter, 2017) to dynamically adjust the amplified reward. Specifically, given the intermediate reward $r_i'' = \alpha_i \cdot \beta_i \cdot r_i$, we compute the final training reward $\tilde{r}_i$ by modulating r_i'' with a cosine decay factor over the course of training:

$$\tilde{r}_i(t) = \text{Decay}(r_i'', t, T), \tag{5}$$

where t is the current training step and T is the total number of training steps. The cosine decay function is defined as:

$$\text{Decay}(r_i'', t, T) = b + 0.5 \cdot (r_i'' - b) \cdot \left(1 + \cos\left(\pi \cdot \frac{t}{T}\right)\right), \tag{6}$$

where $b = 1$ is the baseline reward to which the decayed reward converges. This schedule gradually reduces the influence of reward amplification as training progresses, encouraging exploration in the early stages and promoting stability and convergence in the later stages.

We adopt the standard advantage $\hat{A}_{i,k}$ computation in GRPO, which is computed based on the group of reshaped rewards $\tilde{r} = \{\tilde{r_1}, \tilde{r_2}, \cdots, \tilde{r_G}\}$:

$$\hat{A}_{i,k} = \frac{\tilde{r_i} - \text{mean}(\{\tilde{r_1}, \tilde{r_2}, \cdots, \tilde{r_G}\})}{\text{std}(\{\tilde{r_1}, \tilde{r_2}, \cdots, \tilde{r_G}\})}. \tag{7}$$

The model is finally optimized by maximizing the standard GRPO objective (Shao et al., 2024).

3.4 Inference

During inference, we set the temperature to 0.7 and the top-p value to 1.0. For all evaluation datasets, we use accuracy as the primary metric. For all reasoning formats except the code reasoning format, the final answer is directly generated by the model. For the code reasoning format, if the generated code is executable, we run it using an external interpreter to produce the final answer (*Code-Exec*); if execution fails, the model's generated output is used instead (*Code-Text*).

4 Experiments

4.1 Experimental Setup

Model We adopt Qwen-2.5-VL-7B (Bai et al., 2025) as our backbone model due to its strong performance and demonstrated potential in reinforcement learning. In addition to the base model, we also evaluate Mimo-7B (Xiaomi, 2025), including both its SFT and RL versions, to further examine the effectiveness of GRPO-ALP on different reasoning models. A detailed analysis of the impact of model choice is provided in Section 6.5.

Baselines We compare ARVEC with models trained using SFT, GRPO (Shao et al., 2024), and Ada-GRPO (ARM) (Wu et al., 2025) to evaluate the effectiveness of our proposed GRPO-ALP. For fairness, all baselines are trained on the same data as ARVEC, and the maximum response length is capped at 4,096 tokens to control inference cost.

Evaluation For evaluation, we assess the generalization capability of ARVEC using two types of test sets: in-domain (ID) and out-of-domain (OOD). The ID sets correspond to the datasets used in RL training, most of which are standard benchmarks. Specifically, we merge AIME24 and AIME25 into a single AIME test set, and for MME World and MMK12, we randomly sample 2K instances from each dataset (see Appendix A.1 for details). To further examine generalization, we select six OOD datasets distinct from the ID sets: OBQA (Mihaylov et al., 2018), MATH500 (Lightman et al., 2023), GPQA-Diamond (Rein et al., 2024) (ranging from easy to hard), BLINK (Fu et al., 2024), ChartQA (Masry et al., 2022), and MMMU (Yue et al., 2024) (covering perception and reasoning).

5 Results

5.1 Main Results

We evaluate ARVEC on both in-domain and out-of-domain datasets, demonstrating its effectiveness in balancing token efficiency and task performance, as shown in Table 2. Specifically, we make the following summary based on the observation:

SFT and GRPO achieve task-specific strengths but lack adaptability. We observe that Qwen-2.5-VL-7B$_{SFT}$ and Qwen-2.5-VL-7B$_{GRPO}$ each excel on certain tasks. For instance, SFT performs strongly on MMEWorld, while GRPO shows advantages on GSM8K. However, neither model maintains consistent benefits across all benchmarks, indicating that they do not capture adaptive reasoning strategies. SFT consumes the fewest tokens, which allows it to perform well on tasks requiring minimal reasoning, such as MMEWorld, but it suffers substantially on more complex tasks, such as GSM8K. In contrast, GRPO achieves strong results on GSM8K but expends excessive tokens on easier tasks such as CSQA, and even performs worse than the more token-efficient SFT on MMEWorld. We further study this in Section 6.1.

ARVEC learns to reason adaptively and achieves better token efficiency than the previous adap-

Table 2: Performance of various models on in-domain and out-of-domain evaluation datasets. For consistency with the training setup and to respect computational limits, the maximum pixel size is set to 512×28×28. For parsing, we adopt Qwen-2.5-14B-Instruct as the LLM judge. "†" indicates the test set is a processed subset.

	Accuracy (↑)							**#Tokens (↓)**						
In-Domain														
Models	Text			Text & Vision			Avg.	Text			Text & Vision			Avg.
	CSQA	GSM8K	AIME	MMEWorld†	GEO3K	MMK12		CSQA	GSM8K	AIME	MMEWorld	GEO3K	MMK12	
Qwen-2.5-VL-7B	70.2	86.3	3.3	27.3	35.6	42.7	44.2	180	291	1050	138	376	496	422
Qwen-2.5-VL-7B$_{SFT}$	75.6	33.3	0	48.2	22.8	53.6	38.9	51	53	92	114	129	87	88
Qwen-2.5-VL-7B$_{GRPO}$	79.0	87.5	10.0	43.4	40.3	50.7	51.8	633	636	3729	557	1333	1462	1392
ARM-7B	79.0	84.3	10.0	45.7	31.5	49.3	50.0	174	282	1959	321	1035	754	754
ARVEC-7B	78.7	83.6	10.0	46.5	37.6	50.5	51.2	107	222	1219	124	391	301	394
Δ (ARVEC − Qwen-2.5-VL-7B$_{GRPO}$)	-0.3	-3.9	0	+3.1	-2.7	-0.2	-0.6	-83.1%	-65.1%	-67.3%	-77.7%	-70.7%	-79.4%	-71.7%
Out-of-Domain														
Models	Text			Text & Image			Avg.	Text			Text & Image			Avg.
	OBQA	MATH500	GPQA-Diamond	BLINK	ChartQA	MMMU		OBQA	MATH500	GPQA-Diamond	BLINK	ChartQA	MMMU	
Qwen-2.5-VL-7B	66.6	60.8	25.8	41.6	67.9	32.4	49.2	149	561	616	134	106	341	318
Qwen-2.5-VL-7B$_{SFT}$	85.7	24.4	32.2	42.2	67.0	45.4	49.5	51	78	84	135	81	105	89
Qwen-2.5-VL-7B$_{GRPO}$	85.0	56.4	32.7	44.8	73.2	49.2	57.1	525	2367	3569	728	553	1460	1534
ARM-7B	86.2	50.3	30.3	45.0	76.8	47.9	56.1	220	771	1013	395	259	678	556
ARVEC-7B	86.2	53.0	32.8	47.1	73.0	50.4	57.1	142	580	820	145	151	285	354
Δ (ARVEC − Qwen-2.5-VL-7B$_{GRPO}$)	+1.2	-3.4	+0.1	+2.3	-0.2	+1.2	0	-73.0%	-75.5%	-77.0%	-80.1%	-72.7%	-80.5%	-77.6%

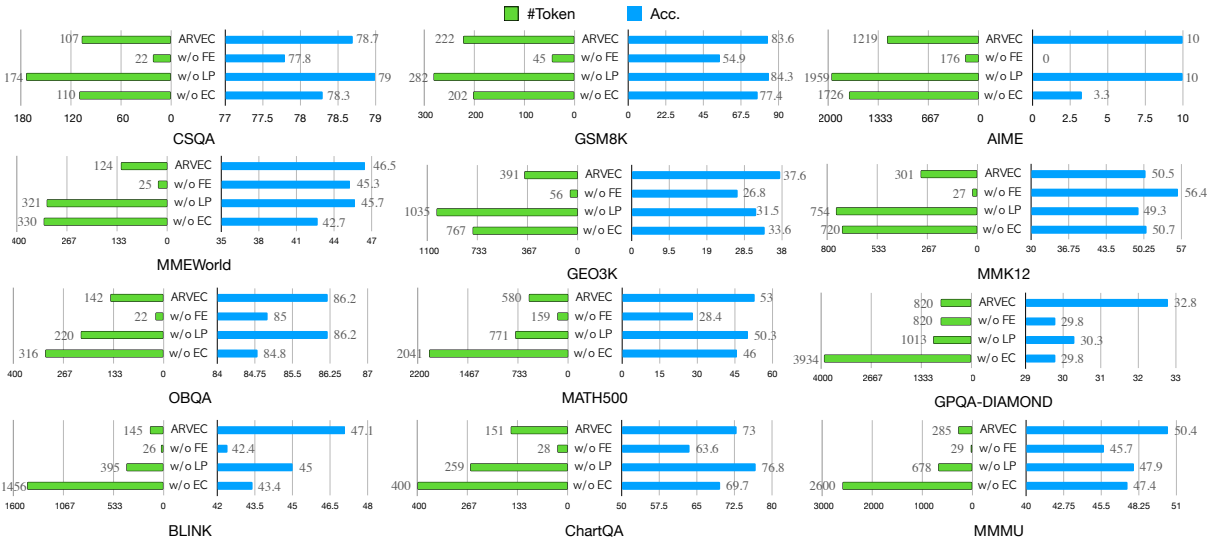

Figure 2: Ablation study of ARVEC across 12 datasets. "w/o FE" denotes the removal of format encouragement rewards while retaining length penalty rewards. "w/o LP" denotes the removal of length penalty rewards while retaining format encouragement rewards. "w/o EC" denotes disabling the code interpreter during both training and inference, forcing the model to reason solely in the code-text format.

tive reasoning strategy. Compared with SFT and GRPO, ARVEC consistently reduces token usage by more than 70% on average relative to GRPO, while maintaining strong performance across both in-domain and out-of-domain tasks. Moreover, by leveraging the length-aware GRPO-ALP, ARVEC demonstrates superior token efficiency, saving over 30% compared with the previous strategy ARM.

5.2 Ablation Study

To evaluate the effectiveness of each module, we conduct ablation experiments by removing it individually and measuring the performance of the resulting models. Specifically, we remove the format encouragement reward ("w/o FE"), the length penalty reward ("w/o LP"), and executable code ("w/o EC") during training and inference, respectively. The results are reported in Figure 2. Our findings are summarized as follows: *1)* **Without format encouragement, models tend to generate extremely short responses, resulting in performance degradation.** In this setting, the length penalty reward disproportionately penalizes longer outputs, which significantly harms performance and leads to large drops across almost all tasks. *2)* **Without the length penalty, responses become substantially longer, yet performance does not improve and may even decline.** As shown in Figure 2, removing the length penalty increases average token usage significantly, but yields no corresponding gains in performance. *3)* **Without code execution, both performance decreases and**

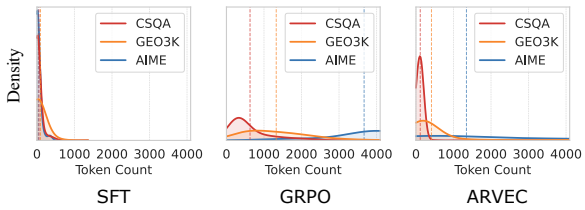

Figure 3: Length distribution of different models across three representative datasets. The dashed vertical line indicates the average token cost for each dataset.

token usage increase. We find that code execution has a positive effect, boosting performance and substantially reducing token costs, especially in the OOD datasets. We attribute this to the fact that, with reliable code execution, tasks requiring heavy reasoning can be offloaded to code, reducing the need for *Long CoT* and thus improving token efficiency. We provide a detailed study in Section 6.3. In summary, each module offers distinct advantages, collectively contributing to enhanced performance and efficiency.

6 Analysis

6.1 Distribution

To illustrate why ARVEC reduces token usage and how it adaptively adjusts reasoning formats across tasks, we present the length distributions of Qwen-2.5-VL-7B$_{SFT}$, Qwen-2.5-VL-7B$_{GRPO}$, and ARVEC on three representative datasets in Figure 3. These datasets, CSQA, GEO3K, and AIME, cover easy, medium, and hard difficulty levels, respectively. We observe that the distribution of Qwen-2.5-VL-7B$_{SFT}$ is concentrated at lower token costs, while Qwen-2.5-VL-7B$_{GRPO}$ exhibits a more dispersed distribution with higher token costs. In contrast, as task difficulty increases, the peak of ARVEC gradually shifts toward higher token counts, demonstrating its ability to allocate longer reasoning adaptively when needed. The reasoning format distribution is provided in Appendix A.2.

6.2 Test-Time Scaling

Test-time scaling enables models to allocate additional tokens at inference to deliver a more reliable answer. We investigate whether ARVEC can benefit from this strategy, particularly under the same token budget as the original reasoning models. Our analysis focuses on two reasoning-intensive tasks, GSM8K and GEO3K, where ARVEC initially lags behind GRPO. We implement test-time scaling via majority voting (Wang et al., 2023): sampling multiple responses and adopting the most frequent one

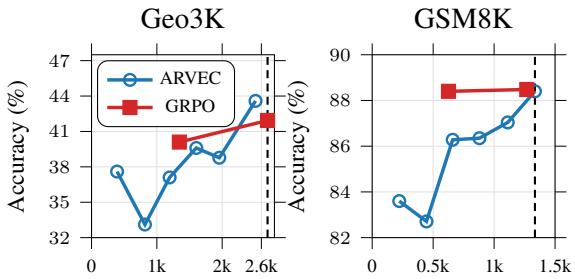

Figure 4: ARVEC vs GRPO with varying token budgets.

Table 3: The performance comparison between ARVEC with and without executable code.

	w/ Code-Exec		w/o Code-Exec	
	Prop.(%)	Acc.	Prop. (%)	Acc.
MATH500	**52.6**	**47.7**	22.8	38.6
ChartQA	**47.4**	**70.3**	27.8	65.2
MMMU	**42.0**	**50.2**	21.9	49.8

as the final prediction. To ensure fairness, we constrain the overall token budget and compare performance accordingly. The results are presented in Figure 4. Our findings show that, although ARVEC exhibits a slight performance drop at lower token budgets (e.g., on GSM8K and GEO3K), test-time scaling effectively mitigates this gap. Moreover, as the token budget increases, ARVEC even surpasses GRPO under the same budget, demonstrating that GRPO-ALP not only improves token efficiency but also preserves reasoning capability when sufficient budget is available.

6.3 The Effect of Executable Code

In Section 5.2, we show that incorporating executable code can substantially reduce token costs while simultaneously improving performance. We hypothesize that executable code enhances the accuracy of code reasoning and thereby substitutes for lengthy chain-of-thought reasoning. To validate this, we examine the proportion and accuracy of code reasoning before and after introducing the code interpreter across three OOD datasets, as reported in Table 3. We observe that both the proportion of code and its accuracy increase with the use of executable code, confirming its effectiveness.

6.4 Length Penalty Strength

We further examine the effect of different values of λ, which controls the strength of the length penalty. Specifically, we vary λ from 0.25 to 1 and report the results in Figure 5. The results reveal that the optimal response length is highly task dependent. For relatively easy or perception-oriented tasks such as CSQA and MMEWorld, shorter outputs

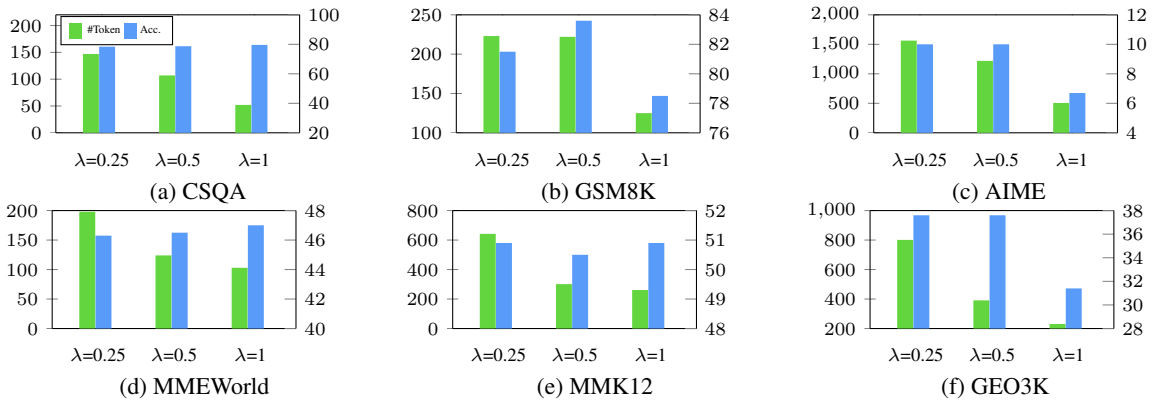

Figure 5: Performance of ARVEC under different length penalty strengths.

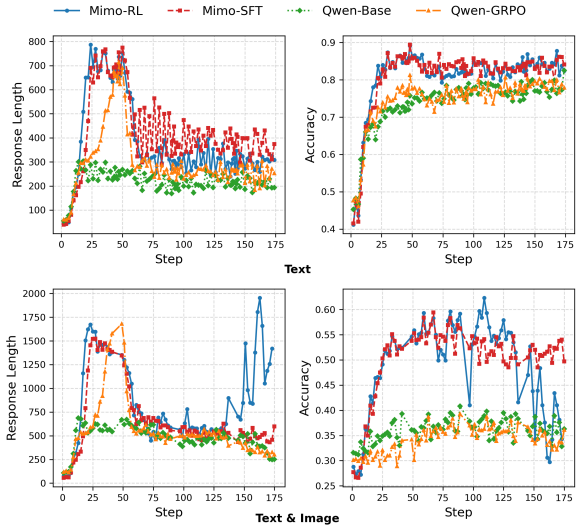

Figure 6: Training trends of ARVEC on text and multimodal datasets with different backbone models.

tend to perform better, since they reduce redundancy, avoid over-reasoning, and align with the fact that these tasks typically require only shallow reasoning or factual recognition. However, on reasoning-intensive tasks such as GSM8K and MMK12, stronger length penalties harm performance, since the models truncate reasoning chains prematurely and fail to capture the multi-step logic needed for correct answers. To balance performance and efficiency, we set λ as 0.5 for all model training in this paper unless otherwise specified.

6.5 Backbone models

For ARVEC, we further investigate the effect of different backbone initializations. Specifically, we compare Mimo-RL and Mimo-SFT (Xiaomi, 2025), which are initialized with RL and SFT, respectively. In addition, we evaluate our own GRPO-initialized Qwen backbone, which is trained with GRPO for 50 steps on the same RL datasets.

The training dynamics of accuracy and response length for these models are illustrated in Figure 6. Our findings suggest several key observations: *1)* **Stronger backbones yield substantial gains.** On the training set, Mimo-based models achieve nearly a 10% improvement in accuracy compared to the weaker model Qwen, highlighting the importance of initialization quality. *2)* **Instability from RL initialization** Despite its higher starting performance, Mimo-RL exhibits noticeable oscillations when trained on multimodal datasets. This instability ultimately reduces its effectiveness and suggests that continual fine-tuning from an RL backbone may amplify variance in cross-modal learning. *3)* **Limited gains from GRPO initialization.** The comparison between models with and without GRPO initialization shows only marginal differences, indicating that lightweight GRPO pretraining alone does not provide a decisive advantage for ARVEC. Greater benefits arise when GRPO is combined with large-scale post-training, as demonstrated in Mimo-RL.

7 Conclusion

In this paper, we propose ARVEC, a multimodal reasoning model capable of adaptive reasoning across tasks. ARVEC is trained with GRPO-ALP, a variant of GRPO that incorporates format encouragement and a length penalty, enabling the model to select among five distinct reasoning formats with high token efficiency. Furthermore, ARVEC integrates executable code into reasoning, which preserves performance while substantially reducing token usage. Comprehensive experiments validate the effectiveness of ARVEC, and ablation studies confirm the soundness of its design. Overall, ARVEC offers a promising direction for adaptive reasoning by leveraging the complementary strengths of different reasoning formats.

8 Limitations

In this paper, we propose ARVEC, a multimodal reasoning model capable of adaptively switching between reasoning formats. A current limitation is that ARVEC, like other adaptive reasoning approaches, relies on a set of pre-defined reasoning formats. While this requirement may constrain full generality, it is mitigated by the fact that such formats can be automatically produced by existing strong closed-source models, as shown in Appendix A.1. We view this dependency as a practical design choice rather than a fundamental bottleneck, and future work could explore more autonomous discovery and refinement of reasoning formats.

References

Pranjal Aggarwal and Sean Welleck. 2025. L1: Controlling how long a reasoning model thinks with reinforcement learning. *arXiv preprint arXiv:2503.04697*.

AIME. Aime problems and solutions.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, and 1 others. 2025. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2023. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, and 1 others. 2024. Do not think that much for 2+ 3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Alejandro Cuadron, Dacheng Li, Wenjie Ma, Xingyao Wang, Yichuan Wang, Siyuan Zhuang, Shu Liu, Luis Gaspar Schroeder, Tian Xia, Huanzhi Mao, and 1 others. 2025. The danger of overthinking: Examining the reasoning-action dilemma in agentic tasks. *arXiv preprint arXiv:2502.08235*.

Chenrui Fan, Ming Li, Lichao Sun, and Tianyi Zhou. 2025. Missing premise exacerbates overthinking: Are reasoning models losing critical thinking skill? In *Second Conference on Language Modeling*.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. 2024. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision*, pages 148–166. Springer.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Bairu Hou, Yang Zhang, Jiabao Ji, Yujian Liu, Kaizhi Qian, Jacob Andreas, and Shiyu Chang. 2025. Thinkprune: Pruning long chain-of-thought of llms via reinforcement learning. *arXiv preprint arXiv:2504.01296*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Yiming Jia, Jiachen Li, Xiang Yue, Bo Li, Ping Nie, Kai Zou, and Wenhu Chen. 2025. Visualwebinstruct: Scaling up multimodal instruction data through web search. *arXiv preprint arXiv:2503.10582*.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Chenwei Lou, Zewei Sun, Xinnian Liang, Meng Qu, Wei Shen, Wenqi Wang, Yuntao Li, Qingping Yang, and Shuangzhi Wu. 2025. Adacot: Pareto-optimal

adaptive chain-of-thought triggering via reinforcement learning. *arXiv preprint arXiv:2505.11896*.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-chun Zhu. 2021. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786.

Haotian Luo, Haiying He, Yibo Wang, Jinluan Yang, Rui Liu, Naiqiang Tan, Xiaochun Cao, Dacheng Tao, and Li Shen. 2025a. Ada-r1: Hybrid-cot via bi-level adaptive reasoning optimization. *arXiv preprint arXiv:2504.21659*.

Haotian Luo, Li Shen, Haiying He, Yibo Wang, Shiwei Liu, Wei Li, Naiqiang Tan, Xiaochun Cao, and Dacheng Tao. 2025b. O1-pruner: Length-harmonizing fine-tuning for o1-like reasoning pruning. *arXiv preprint arXiv:2501.12570*.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, and 1 others. 2025. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *arXiv preprint arXiv:2503.07365*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391.

OpenAI. 2024. Hello GPT-4o.

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

ByteDance Seed, Jiaze Chen, Tiantian Fan, Xin Liu, Lingjun Liu, Zhiqi Lin, Mingxuan Wang, Chengyi Wang, Xiangpeng Wei, Wenyuan Xu, and 1 others. 2025. Seed1. 5-thinking: Advancing superb reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.13914*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and 1 others. 2025. Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024. How easily do irrelevant inputs skew the responses of large language models? In *First Conference on Language Modeling*.

Siye Wu, Jian Xie, Yikai Zhang, Aili Chen, Kai Zhang, Yu Su, and Yanghua Xiao. 2025. Arm: Adaptive reasoning model. *arXiv preprint arXiv:2505.20258*.

LLM-Core-Team Xiaomi. 2025. Mimo-vl technical report. *Preprint*, arXiv:2506.03569.

Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and 1 others. 2025. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, and 1 others. 2024. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567.

Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. 2025. Adaptthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*.

YiFan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, and 1 others. 2024. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? In *The Thirteenth International Conference on Learning Representations*.

Jason Zhu and Hongyu Li. 2025. Towards concise and adaptive thinking in large reasoning models: A survey. *arXiv preprint arXiv:2507.09662*.

A Appendix

A.1 Dataset Collection and Preprocessing

For CSQA, GSM8K, and GEO3K, we adopt the standard training sets for RL training and the validation/test sets for evaluation. For datasets without official training splits, including AIME and MMEWorld, we construct our own training and test sets. Specifically, for AIME, we collect problems and corresponding solutions from 1987–2023, reserving the 60 problems from 2024–2025 as the test set. For MMEWorld, we randomly sample 4,000 instances for training and 2,000 instances for testing. For MMK12, we use the standard test set for evaluation and randomly sample 4,000 instances from the training set for RL training.

A.2 Reasoning Format Distribution

In Figure 7, we show the distribution of reasoning formats across three models and three datasets. The SFT model tends to rely heavily on direct reasoning for all tasks, while the GRPO model predominantly favors long chain-of-thought reasoning. In contrast, ARVEC demonstrates adaptive reasoning by adjusting its format selection according to task characteristics.

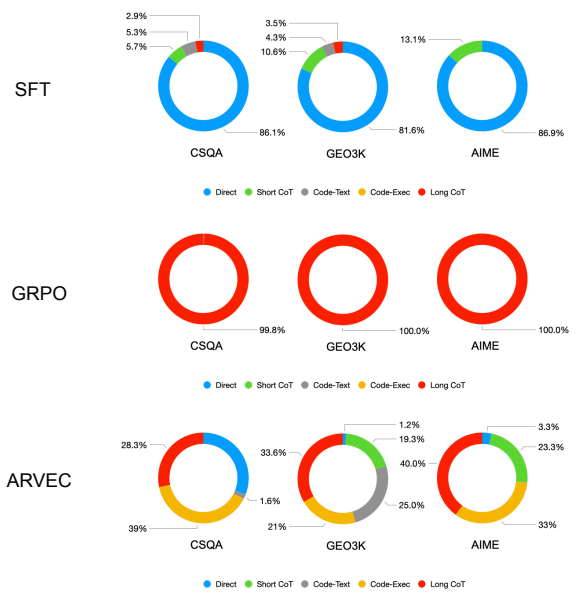

Figure 7: Reasoning format distribution.

A.3 Implementation Details

A.3.1 Data Collection

For VisualWebInstruct, each question is paired with both an answer and a rationale (short CoT). To obtain long CoT reasoning, we prompt Doubao-V1.6-Thinking, with the prompt provided below. For code-based reasoning, we prompt Doubao-V1.6-Thinking to generate a function, which is then executed by an external Python interpreter. We retain only those cases where both the long chain-of-thought and the generated code yield the correct ground truth.

Prompt for generating Long CoT:

```
You are a helpful assistant who can help
 me extract the answer and rationale
from the following text. Note that you
should help me remove special characters
 and only return plain text. Except for
the final answer, all the rest in the
response should be regarded as rationale
. Besides, you should tell me whether
the question type is free-form or
multiple-choice.  For a multiple-choice
question, only return the answer in ['A
', 'B',' C', D'].
Remove the unit if the answer is a
number. Follow the format: Answer: xxx\n
\nRationale: xxx\n\nFormat: xxx

Question: <PLACEHOLDER>
```

Prompt for generating Code:

```
You are a helpful assistant who can help
 me extract the answer and rationale
from the following text. Note that you
should help me remove special characters
 and only return plain text. Except for
the final answer, all the rest in the
response should be regarded as rationale
. Besides, you should tell me whether
the question type is free-form or
multiple-choice.  For a multiple-choice
question, only return the answer in ['A
', 'B',' C', D'].
Remove the unit if the answer is a
number. Follow the format: Answer: xxx\n
\nRationale: xxx\n\nFormat: xxx

Question: <PLACEHOLDER>
```

A.3.2 SFT

We train the SFT model using LoRA (Hu et al., 2022), with a rank of 8. The maximum sequence length is set to 8,192 tokens, and training is performed for up to 6 epochs. We use a learning rate of 2×10^{-4} with a cosine scheduler and a warmup ratio of 0.1. The validation split is 0.1, sampled from the training set. The batch size for training is 2 per GPU, with a gradient accumulation step of 8. The SFT model is trained through 8×A100, which took ∼12 hours.

A.3.3 RL

We implement reinforcement learning using the VERL framework (Sheng et al., 2024). The training batch size is set to 480, with a mini-batch size of 96.

During the rollout stage, each instance generates 8 samples, with a temperature of 0.7 and a top-p value of 1.0. The learning rate is fixed at 1×10^{-6}. The RL model is trained through 8×A100 for 150 steps, which took∼12 hours.

A.3.4 Data Examples

We provide an example with four reasoning formats below.

```
Question:
<image>
The following bullet points appear in a
proposal.
What is the most likely heading for the
section these bullet points are in?
Reduce medical errors by 25% within one
year of implementing the new EHR system.
Increase patient satisfaction with the
EHR system by 10% within one year of
implementation.
Reduce administrative costs associated
with the EHR system by 15% within one
year of implementation.

<ANSWER>
Project Objectives
</ANSWER>


<COT>
The bullet points outline specific,
measurable goals to be achieved within a
 specified timeframe after implementing
a new EHR system. They focus on outcomes
 or targets that the proposal aims to
achieve, which aligns with the heading
\"Project Objectives.\" The other
options do not match the content of the
bullet points as closely.
</COT>
<ANSWER>
Project Objectives
</ANSWER>

<CODE>
def determine_heading_for_bullet_points
():
    headings = [
        "Potential Solutions",
        "Project Objectives",
        "Statement of the Problem",
        "Implementation Plan"
    ]
    bullet_points = [
        "Reduce medical errors by 25%
        within one year of implementing
        the new EHR system.",
        "Increase patient satisfaction
        with the EHR system by 10%
        within one year of
        implementation.",
        "Reduce administrative costs
        associated with the EHR system
        by 15% within one year of
        implementation."
    ]
    # Analysis:
    # The bullet points are all specific
    , measurable targets to be achieved
    after implementation.
    # This matches the definition of "
    Project Objectives" best.
    # They are not solutions, not a
    problem statement, and not an
    implementation plan.
    answer = "Project Objectives"
    index = headings.index(answer)
    return {
        'bullet_points': bullet_points,
        'chosen_heading': answer,
        'heading_index': index,
        'answer': answer
    }

>>> determine_heading_for_bullet_points
()
</CODE>
<OBSERVATION>
output = {
    'bullet_points': [
        "Reduce medical errors by 25%
        within one year of implementing
        the new EHR system.",
        "Increase patient satisfaction
        with the EHR system by 10%
        within one year of
        implementation.",
        "Reduce administrative costs
        associated with the EHR system
        by 15% within one year of
        implementation."
    ],
    'chosen_heading': 'Project
    Objectives',
    'heading_index': 1,
    'answer': 'Project Objectives'
}</OBSERVATION>
<ANSWER>
Project Objectives
</ANSWER>

<LONG_COT>
Got it, let's see. The question is about
 finding the right heading for those
bullet points. First, I need to remember
 what each section title means.

The bullet points are all about specific
, measurable goals: reduce errors by
25%, increase satisfaction by 10%,
reduce costs by 15. All within a year of
 implementing the EHR. \n\nNow look at
the options:

- Potential Solutions: That's what you
might do to solve a problem, not the
goals.

- Project Objectives: Objectives are the
 specific goals a project aims to
achieve. That fits because these are the
 things the project (implementing the
EHR) wants to accomplish.

- Statement of the Problem: That's about
 the issues or problems that need to be
addressed, not the goals.
```

```

- Implementation Plan: That's how you'll
 carry out the project, like steps or
timeline, not the goals themselves.

So the bullet points are clearly the
objectives of the project. The heading
should be Project Objectives.
</LONG_COT>
<ANSWER>
Project Objectives
</ANSWER>
```