RLAE: Reinforcement Learning-Assisted Ensemble for LLMs

Yuqian Fu[1,2], **Yuanheng Zhu**[1,2,†], **Jiajun Chai**[1,2,3],
Guojun Yin[3], **Wei Lin**[3], **Qichao Zhang**[1,2], **Dongbin Zhao**[1,2,†]

[1]Institute of Automation, Chinese Academy of Sciences,
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences,
[3]Meituan
`{fuyuqian2022,yuanheng.zhu}@ia.ac.cn`

Abstract

Ensembling large language models (LLMs) can effectively combine diverse strengths of different models, offering a promising approach to enhance performance across various tasks. However, existing methods typically rely on fixed weighting strategies that fail to adapt to the dynamic, context-dependent characteristics of LLM capabilities. In this work, we propose ***R****einforcement* ***L****earning-****A****ssisted* ***E****nsemble for LLMs* (RLAE), a novel framework that reformulates LLM ensemble through the lens of a Markov Decision Process (MDP). Our approach introduces a RL agent that dynamically adjusts ensemble weights by considering both input context and intermediate generation states, with the agent being trained using rewards that directly correspond to the quality of final outputs. We implement RLAE using both single-agent and multi-agent reinforcement learning algorithms (RLAE_{PPO} and $\text{RLAE}_{\text{MAPPO}}$), demonstrating substantial improvements over conventional ensemble methods. Extensive evaluations on a diverse set of tasks show that RLAE outperforms existing approaches by up to 3.3% accuracy points, offering a more effective framework for LLM ensembling. Furthermore, our method exhibits superior generalization capabilities across different tasks without the need for retraining, while simultaneously achieving lower time latency. The source code is available at here.

1 Introduction

Recent years have witnessed remarkable progress in large language models (LLMs), demonstrating impressive capabilities in natural language understanding and reasoning. However, significant performance variations persist among different models in downstream tasks due to three primary factors: training data bias, architectural differences, and diversity in training algorithms (Jiang et al., 2023b). For example, GPT-4o (OpenAI, 2024) excels in mathematical reasoning, while Claude-series models (Anthropic, 2024) demonstrate superior performance in code generation tasks. Nevertheless, retraining these models with mixed training data, integrating model architectures, or adjusting training algorithms are expensive and impractical. Therefore, these distinct capabilities motivate research into LLM ensemble methods that leverage the diverse strengths of different models (Chen et al., 2025; Frick et al., 2025; Lu et al., 2024), aiming to achieve enhanced performance on given tasks.

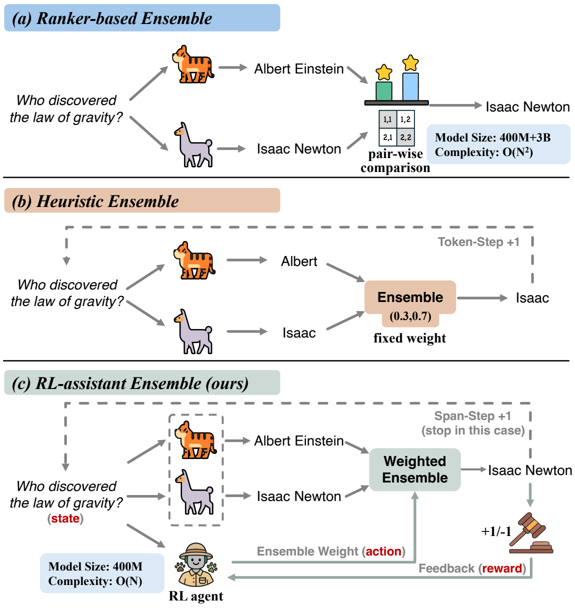


Figure 1: Overview of Ranker-based Ensemble, Heuristic Ensemble, and our RL-assisted Ensemble methods.

The complexity of LLMs presents unique challenges for ensemble, requiring a more flexible framework beyond previous approaches. Figure 1 illustrates the key differences between existing ensemble methods and our proposed framework. Ranker-based ensemble methods, as shown in Figure 1 (a), aim to select the best candidate from

---

† The corresponding authors.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 13452–13466
November 4-9, 2025 ©2025 Association for Computational Linguistics

outputs generated by various LLMs (Jiang et al., 2023b; Tekin et al., 2024). However, these methods face several significant limitations: they require quadratic computational cost for pair-wise comparisons, demand additional time for selection or fusion, and encounter scalability issues when scaling to multiple LLMs. Moreover, ranker-based methods typically operate at the response-level, providing merely coarse-grained ensemble capabilities.

To address these issues, heuristic ensemble methods, shown in Figure 1 (b), perform ensemble operations at each token generation step (Yu et al., 2024; Huang et al., 2024; Mavromatis et al., 2024; Yao et al., 2025). While these token-level approaches fuse probability distributions from different models through weighted averaging, they lack consideration of the border context that affects ensemble performance. Furthermore, both ranker-based and heuristic ensemble methods face two critical challenges in ensemble weight allocation: (1) they rely on fixed or manually designed rules to assign weights, failing to adapt to domain shifts in input texts (e.g., from mathematical proofs to story writing), thus limiting their generalization ability across diverse contexts, and (2) they inadequately consider context dependencies throughout the generation process, resulting in responses that may be locally optimal but globally suboptimal. Consequently, developing effective LLM ensemble methods remains an open research challenge.

In this paper, we propose a *Reinforcement Learning-assisted Framework for LLM Ensemble* (RLAE) that reformulates the ensemble process as a reinforcement learning problem. RLAE employs RL agent(s) to dynamically adjust the ensemble weights of multiple LLMs based on the input prompt and intermediate generation states, in order to maximize the quality of the final response. Our framework is grounded in two fundamental insights: (1) different models exhibit context-specific strengths that vary across tasks, and (2) local decisions in reasoning tasks affect global outcomes through path-dependent effects (Xiong et al., 2025). Specifically, we formulate the ensemble problem as a Markov Decision Process, where the `state` contains the input prompt and generated response history, while the `action` space determines the ensemble weights at each generation step. To strike a balance between response-level and token-level approaches, RLAE implements a span-level ensemble strategy, an approach that has shown considerable promise in recent work (Xu et al., 2025b; Lv et al., 2024). While our framework maintains flexibility in its `reward` function design, we primarily define the output response quality as the reward signal to directly align with the ultimate goal of LLM ensemble. We instantiate RLAE using both single-agent and multi-agent reinforcement learning algorithms. In the multi-agent setting, we treat each LLM as an independent agent, while in the single-agent setting, one agent is responsible for all the ensemble weights (Zhang et al., 2025b). Compared to existing LLM ensemble methods, RLAE achieves an effective balance between ensemble quality and computational efficiency, ensuring the alignment between ensemble objectives and response quality.

We evaluate our method on a diverse set of benchmarks, including general reasoning, mathematical and scientific problem solving, and code generation. Experimental results demonstrate that RLAE outperforms baseline approaches in both performance and computational efficiency. Additionally, our framework exhibits generalization capabilities across different tasks without retraining.

Overall, our **key contributions** are:

- We innovatively formulate the LLM ensemble pipeline as a reinforcement learning task and propose RLAE, a reinforcement learning-assisted framework for ensemble. To the best of our knowledge, this is the first framework that uses reinforcement learning to adaptively optimize ensemble weights.
- We employ both single-agent and multi-agent reinforcement learning algorithms to instantiate our framework, each demonstrating distinct advantages in different scenarios.
- We provide extensive experimental evidence to demonstrate RLAE's effectiveness and generalization capabilities, which improves the performance by up to 3.3% accuracy points compared to previous ensemble methods.

2 Related Works

LLM Ensemble. Recent advances in LLM capabilities have sparked significant interest in ensemble methods that combine multiple models to achieve superior performance. Current approaches can be categorized into three main types: token-level, response-level, and span-level ensembles. Token-level methods, such as GAC (Yu et al., 2024), DEEPEN (Huang et al., 2024), and PACK-LLM (Mavromatis et al., 2024), combine probabil-

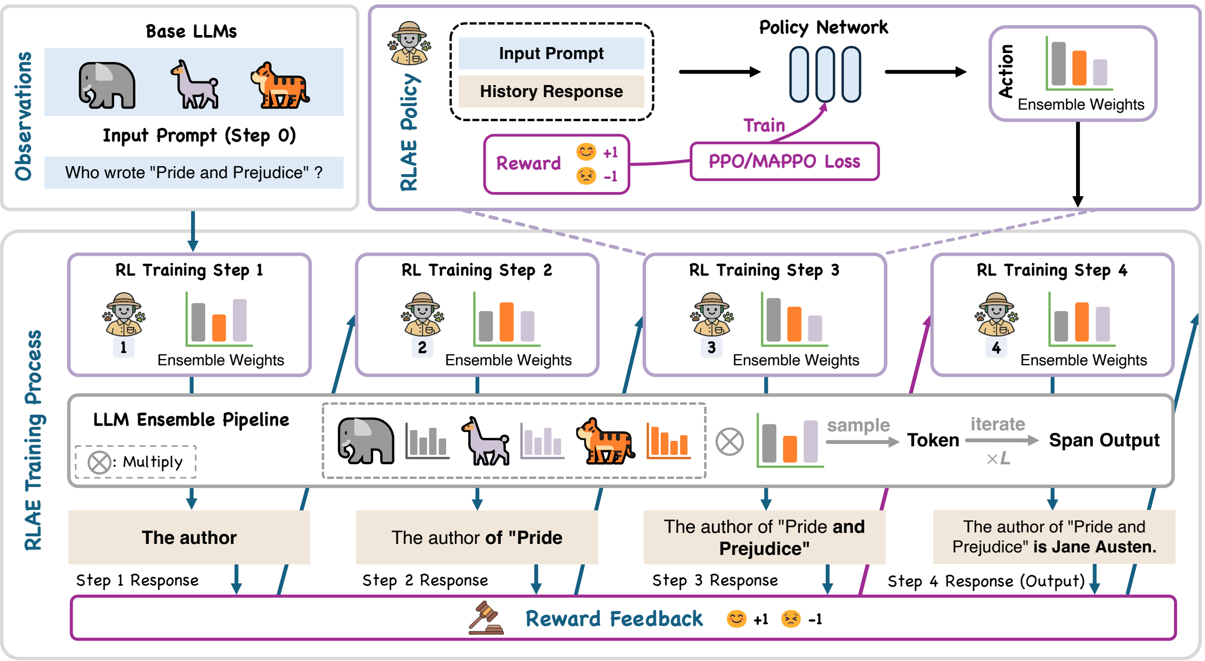


Figure 2: The RLAE framework. Top row: at each generation step, our RL agent makes actions (ensemble weights) based on the input prompt and history response; the reward feedback is used to train the policy network of the RL agent with the PPO or MAPPO loss. Bottom row: visualization of the iterative generation process of our RL-assisted ensemble framework, where the reward feedback from the final output is continuously used to train the RL agent.

ity distributions during generation through vocabulary alignment and distribution projection. While these methods enable fine-grained ensemble, they often compromise semantic coherence and struggle with computational complexity. Response-level approaches, including LLM-BLENDER (Jiang et al., 2023b) and LLM-TOPLA (Tekin et al., 2024), select or combine complete responses after generation. While these methods are practical for black-box LLMs, they are unable to effectively leverage the complementary strengths of different models. Besides, both approaches are constrained by fixed weight allocation strategies and insufficient consideration of context dependencies. In this paper, we propose a RL-assisted framework that dynamically adjusts ensemble weights based on the prompt and generation history. Our method aligns with span-level approaches, such as SWEETSPAN (Xu et al., 2025b) and SPECFUSE (Lv et al., 2024), which operate on sequences of spans to achieve a balance between granularity and efficiency.

Reinforcement Learning in LLM. Reinforcement learning has been extensively used to enhance LLM capabilities. Reinforcement learning from human feedback (RLHF) leverages human preferences to guide model outputs (Ouyang et al., 2022; Tu et al., 2025), while reinforcement fine-tuning (RFT) employs task-specific rewards for performance improvements in tasks (Guo et al., 2025; Fu et al., 2025; Xu et al., 2025a; Chai et al., 2025; Zhang et al., 2025a). Additionally, LLM routing (Zheng et al., 2025) or mixture-of-agents (MoA) (Chakraborty et al., 2025) methods employ RL to dynamically select the most appropriate LLM or expert based on input prompt or intermediate generation steps. However, these applications typically focus on selecting or fine-tuning individual models rather than coordinating multiple models for ensemble. In contrast, our framework represents the first to apply RL for adaptive ensemble weight adjustment, enabling the integration of the strengths of different models.

3 Methodology

In this section, we introduce RLAE, a reinforcement learning-assisted framework designed for dynamic LLM ensemble that facilitates collaborative inference among models with complementary strengths. As illustrated in Figure 2, RLAE employs a RL agent to adjust ensemble weights based on both the input prompt and intermediate generation responses. We organize this section as follows: Section 3.1 formalizes the LLM ensemble problem, Section 3.2 details our *Reinforcement Learning-Assisted LLM Ensemble Framework*, and Section 3.3 presents our training methodology.

3.1 Problem Formulation

LLM Ensemble Problem. Consider a set of K foundational LLMs $\mathcal{M} = \{M_1, \dots, M_K\}$ with corresponding parameters $\{\theta_1, \dots, \theta_K\}$. For a given input prompt $\boldsymbol{x} = (x_1, \dots, x_m)$ of length m, the ensemble generates an output sequence $\boldsymbol{y} = (y_1, \dots, y_H)$ of length H through iterative probability fusion:

$$p(y_h|\boldsymbol{x}, \boldsymbol{y}_{<h}) = \sum_{k=1}^{K} w_h^{(k)} p_{M_k}(y_h|\boldsymbol{x}, \boldsymbol{y}_{<h}, \theta_k), \tag{1}$$

where $w_h^{(k)} \in [0, 1]$ denotes the ensemble weight assigned to model M_k at generation step h, satisfying the constraint $\sum_{k=1}^{K} w_h^{(k)} = 1$, and $p_{M_k}(y_h|\cdot)$ is the probability of the h-th token generated by model M_k. The history context $\boldsymbol{c}_h = (\boldsymbol{x}, \boldsymbol{y}_{<h})$ contains both the input prompt and previously generated tokens, while a weight function $\pi_\phi : \mathcal{C} \times \mathcal{M} \to \Delta^{K-1}$, parameterized by ϕ, determines the weights over the $(K-1)$-dimensional probability simplex.

Span-Level Ensemble. Traditional LLM ensemble approaches (Yu et al., 2024; Huang et al., 2024; Mavromatis et al., 2024) typically combine probabilities across the entire vocabulary at each generation step. This approach imposes significant computational overhead during inference, adversely affecting both performance and efficiency. To address these challenges while maintaining semantic coherence, we implement span-level ensemble, an approach increasingly explored in recent work (Xu et al., 2025b; Lv et al., 2024). Instead of adjusting weights for individual tokens, our method applies consistent weights at the span level, where each span $\boldsymbol{z}_t$ comprises L consecutive tokens:

$$\boldsymbol{z}_t = (y_{t,1}, y_{t,2}, \dots, y_{t,L}), \tag{2}$$

where $\boldsymbol{z}_t$ represents tokens generated in the t-th span, and L is a predefined span length. This strategic approach reduces decision points from H (token-level) to $\lceil H/L \rceil$ (span-level), substantially improving computational efficiency while ensuring semantic continuity. This design enables the RL agent to focus on examining model contributions at span-level, streamlining the state-action space for RL training and enhancing response quality. Furthermore, following GAC (Yu et al., 2024), we recognize that not all generation steps in the span necessarily require ensembling. Therefore, we selectively ensemble only critical tokens within each span, further optimizing computational efficiency without compromising generation quality.

Essential Elements of MDP. We formulate the LLM ensemble as a Markov Decision Process (MDP) with the following components:

- **`State`** (s_t): The state is represented as $s_t = (\boldsymbol{x}, \boldsymbol{z}_{<t})$, capturing both the input prompt and the response up to the current span t.
- **`Action`** (a_t): The action at span t is defined as $a_t = (w_t^{(1)}, \dots, w_t^{(K)})$, representing the ensemble weights across models for the next span generation, where $\sum_{k=1}^{K} w_t^{(k)} = 1$ and $w_t^{(k)} \in [0, 1]$ for all k.
- **`Reward`** ($r(s_t, a_t)$): The reward function evaluates the generation quality. For terminal states, it uses task-specific metrics. For intermediate states, it employs a process reward model that provides dense feedback.
- **`Transition`** ($P(s_{t+1}|s_t, a_t)$): The transition function defines the environment dynamics based on the current state and action. In LLM ensemble, this is determined by the weighted combination of model probabilities for generating the next span of tokens.
- **`Policy`** ($\pi_\phi(a_t|s_t)$): As the weight function mentioned before, the policy function defines the ensemble weight distribution across models based on the current state.

Previous LLM ensemble approaches often compute fixed weights across the entire dataset (Chen et al., 2025; Lu et al., 2024), which fails to capture the context-dependent nature of model performance. These methods typically assign fixed weights to each model based on performance metrics or heuristic rules, ignoring how model strengths vary across different inputs and generation steps. In contrast, our reinforcement learning-assisted framework, RLAE, learns to dynamically adjust weights based on the prompts and intermediate generation states, effectively adjusting the ensemble weights.

3.2 Reinforcement Learning-Assisted LLM Ensemble Framework (RLAE)

Our ensemble procedure operates in an autoregressive manner, which adjusts weights dynamically at the end of each span. The process is formalized in Algorithm 1 in Appendix A. The algorithm begins with an empty output sequence and iteratively builds the response span by span. At each span, the RL agent evaluates the current state, which includes the input prompt and generated responses. Based on this evaluation, the RL agent samples a weight distribution across the base models, determining their relative contributions to the next span. These weights remain fixed throughout the span generation, with each token sampled from the weighted probability distribution of the ensemble. This process continues until reaching either the desired output length or the termination token `<EOS>`. The RL agent's policy π_ϕ learns to assign appropriate weights by analyzing input patterns and partial generations, effectively identifying which models are most reliable for different contexts. This dynamic adaptation allows the ensemble to leverage the strengths of each base model while mitigating their individual weaknesses, resulting in higher quality generations. Another key challenge in LLM ensemble stems from the fact that different LLMs are trained based on different tokenizers, resulting in diverse vocabulary sets across models. To address vocabulary mismatch, RLAE projects probability vectors from multiple LLMs into a unified vocabulary space through a mapping matrix. More details can be found in Appendix D.

3.3 Training Process of RLAE

For optimization, the RL agent employs Proximal Policy Optimization (PPO) (Schulman et al., 2017) for single-agent training and its multi-agent variant (MAPPO) (Yu et al., 2022) for multi-agent scenarios. The training process consists of iteratively collecting trajectories by executing the current policy on input prompts, followed by policy and value function optimization. Algorithm 2 in Appendix A provides a detailed description of this process.

3.3.1 Single-Agent Training

In the single-agent version RLAE$_{\text{PPO}}$, we formulate the ensemble process as a centralized decision-making problem, where a single RL agent controls the weights of all LLMs simultaneously. At each step, the agent observes the current state and outputs a weight distribution over base models, determining their relative contributions to the output. We optimize RLAE$_{\text{PPO}}$ using PPO, which employs a policy network (actor) that dynamically adjusts ensemble weights, coupled with a value network (critic) that estimates expected returns. The objective function of PPO is:

$$L_{\text{PPO}}(\phi) = \min\left(\frac{\pi_\phi(a|s)}{\pi_{\phi_{old}}(a|s)} A^{\pi_{\phi_{old}}}(s,a),\right.$$
$$\left.\text{clip}\left(\frac{\pi_\phi(a|s)}{\pi_{\phi_{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\phi_{old}}}(s,a)\right), \tag{3}$$

where $\pi_\phi/\pi_{\phi_{old}}$ denotes the importance sampling ratio, which measures the difference between the new and old policies, and ϵ serves as the clipping parameter constraining policy updates. The advantage function $A(s,a)$ is estimated using GAE (Schulman et al., 2016):

$$A(s,a) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \, \delta_{t+l}, \tag{4}$$

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{5}$$

where γ is the discount factor, λ is the GAE parameter, δ_t is the temporal difference (TD) error, and $V(s)$ is the value function. The value network is trained to minimize the mean squared error between its predictions and the actual returns, which is detailed in Appendix C.

3.3.2 Multi-Agent Training

As the number of base models increases, the single-agent approach faces challenges in managing the expanding joint action space and modeling complex interactions between base models. To address these limitations, we reformulate the problem as a Markov game (see Appendix E for details) where each LLM is controlled by an independent RL agent that determines its model's contribution weight. This design offers three advantages: (1) It decomposes the complex joint action space into smaller individual action spaces, reducing the dimensionality of the learning problem for each agent; (2) It enables each RL agent to specialize in understanding its corresponding model's strengths and weaknesses; and (3) It facilitates more flexible scaling through independent agent addition.

In RLAE$_{\text{MAPPO}}$, we optimize RL agents using MAPPO, which extends PPO to a multi-agent setting. Each RL agent outputs a scalar logit value through its own policy network. These logits are then concatenated and passed through a `Softmax` function to obtain the final ensemble weights that sum to 1. The agents share a centralized critic that coordinates global rewards, following the centralized training with decentralized execution (CTDE) paradigm (Gronauer and Diepold, 2022). The objective function of MAPPO is:

$$L_{\text{MAPPO}}(\phi) = \frac{1}{K}\sum_{k=1}^{K}\min\left(\frac{\pi_\phi(a^{(k)}|s)}{\pi_{\phi_{old}}(a^{(k)}|s)}A^{(k)},\right.$$
$$\left.\text{clip}\left(\frac{\pi_\phi(a^{(k)}|s)}{\pi_{\phi_{old}}(a^{(k)}|s)}, 1-\epsilon, 1+\epsilon\right)A^{(k)}\right), \tag{6}$$

where $\pi_\phi(a^{(k)}|s)$ is the policy of agent k. The centralized critic takes the global state as input to better estimate the value function, facilitating more effective credit assignment across agents. This design allows for context-aware weight adjustments while maintaining alignment with overall ensemble objectives through shared reward signals.

4 Experiments

4.1 Experimental Settings

Benchmarks. We evaluate our method on seven benchmarks across three capability dimensions: (1) **General Ability**: MMLU (0-shot) (Hendrycks et al., 2021), a multiple-choice dataset covering 57 subjects across STEM, humanities, and social sciences; ARC-C (0-shot) (Clark et al., 2018), containing questions from standardized science exams for grades 3-9; and TriviaQA (5-shot) (Joshi et al., 2017), a factual question-answering dataset compiled by trivia enthusiasts to test retrieval of world knowledge. (2) **Math and Science Ability**: GSM8K (5-shot) (Cobbe et al., 2021), featuring linguistically diverse grade school math word problems requiring multi-step reasoning; PIQA (0-shot) (Bisk et al., 2020), a physical commonsense reasoning dataset; and GPQA (5-shot) (Rein et al., 2024), a graduate-level professional question-answering benchmark focusing on physics, chemistry, and biology. (3) **Code Generation**: MBPP (0-shot) (Austin et al., 2021), where models generate Python code for basic programming problems designed for entry-level programmers.

Base Models. As ensemble methods typically work better with models of comparable performance but diverse capabilities, we select a set of models with similar parameter scales. Our base LLMs includes Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen-2-7B-Instruct (Yang et al., 2024a), Qwen-2.5-7B-Instruct (Yang et al., 2024b), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023a), and OpenChat-3.5 (Wang et al., 2024).

Baselines. We compare our method against three representative LLM ensemble approaches: (1) **PAIRRANKER**, the key component of LLM-BLENDER (Jiang et al., 2023b), which employs pairwise comparison to evaluate candidate outputs from different LLMs and selects the highest-scoring response as the final output (we omit GEN-FUSER due to its significant refusal rate on our benchmarks); (2) **GAC** (Yu et al., 2024), which constructs a union dictionary by combining vocabularies from multiple models and projects each model's token distribution onto this unified space for aggregated token selection at each generation step; and (3) **DEEPEN** (Huang et al., 2024), which leverages relative representation theory to map probability distributions from different models into a shared space for aggregation, using the intersection of model vocabularies as the basis for this projection. These baselines represent diverse approaches to LLM ensemble, spanning ranker-based ensemble and heuristic ensemble methods.

Implementation Details. For the RL agent architecture, we select DeBERTa-V3-Large (He et al., 2021), a compact yet powerful model with around 400M parameters to minimize training overhead. For reward design, we employ a rule-based sparse reward model that provides feedback only at the terminal state of generation, utilizing benchmark-specific evaluation metrics such as accuracy. Although reward computation introduces some computational overhead during training, it is worth noting that RLAE operates without requiring any reward or supervision signals at inference time.

4.2 Main Results

Tables 1 and 2 demonstrate the performance of our proposed RLAE framework compared to base models and baseline ensemble methods across various benchmarks. Overall, our method consistently outperforms existing methods in most scenarios, with $\text{RLAE}_{\text{MAPPO}}$ achieving a superior average score of 70.1 in the primary experimental set, surpassing all base models and ensemble methods. From these results, we derive several observations:

(1) Effective Performance Enhancement with Comparable Base Models. Our experiments demonstrate significant improvements when ensembling models with similar performance levels. For instance, when combining Llama-3.1 and Qwen-2, which have a relatively small performance gap of 1.5 points on MMLU, $\text{RLAE}_{\text{MAPPO}}$ achieves a sub-

Dataset	MMLU		ARC-C		GPQA		GSM8K		MBPP	
	2 LLMs	3 LLMs	2 LLMs	3 LLMs	2 LLMs	3 LLMs	2 LLMs	3 LLMs	2 LLMs	3 LLMs
Llama-3.1-8B-Instruct	66.8	66.8	79.5	79.5	32.8	32.8	84.5	84.5	69.6	69.6
Qwen-2-7B-Instruct	65.3	65.3	81.9	81.9	34.3	34.3	85.7	85.7	67.2	67.2
Qwen-2.5-7B-Instruct	68.2	68.2	84.3	84.3	36.4	36.4	91.6	91.6	76.3	76.3
PairRanker (Jiang et al., 2023b)	63.8	69.1	78.6	82.8	32.7	34.2	86.8	**91.7**	66.1	64.5
GaC (Yu et al., 2024)	67.5	67.8	82.1	84.5	**35.0**	33.4	83.1	88.1	68.6	74.6
DeePEn (Huang et al., 2024)	67.1	68.0	81.8	84.1	33.8	32.6	84.2	86.2	**69.9**	69.9
RLAE$_{\text{PPO}}$	**68.5**	**69.2**	**82.8**	**84.7**	**35.1**	**36.1**	**86.9**	91.3	**70.5**	**75.8**
RLAE$_{\text{MAPPO}}$	**69.1**	**70.1**	**83.4**	**85.6**	34.7	**35.3**	**87.4**	**92.5**	69.8	**75.3**

Table 1: Performance of LLM ensemble. 2 LLMs mean Llama-3.1 and Qwen-2; 3 LLMs mean Llama-3.1, Qwen-2, and Qwen-2.5. **Highlight** indicates the best performing method, while **bold** indicates the second best.

Model	Dataset				
	MMLU	ARC-C	TriviaQA	GSM8K	PIQA
Mistral-7B-Instruct-v0.3	59.3	74.5	64.3	56.5	80.6
OpenChat-3.5	60.8	78.1	61.7	73.4	87.1
PairRanker	60.3	75.9	56.3	67.7	**82.9**
GaC	55.3	73.8	62.2	60.8	69.2
DeePEn	**61.8**	71.6	**67.3**	**69.4**	77.5
RLAE$_{\text{PPO}}$	61.1	**79.2**	**67.5**	66.6	**81.2**
RLAE$_{\text{MAPPO}}$	**62.5**	**78.4**	65.6	**67.9**	80.8

Table 2: Performance of ensemble methods with Mistral 7B and OpenChat 3.5.

stantial 2.3-point improvement over the stronger base model. We observe similar gains on ARC-C, where RLAE$_{\text{MAPPO}}$ increases performance to 83.4 points, representing a 1.5-point improvement over base model performance. However, the improvements become more modest when incorporating Qwen-2.5, which outperforms other models by 2.9 points on MMLU. In this case, the ensemble yields only a 1.9-point improvement to reach 70.1 points, highlighting the challenges in effectively combining models with larger performance disparities.

(2) RLAE$_{\text{MAPPO}}$ Superiority with Heterogeneous Base Models. In scenarios with significant performance variations among base models, RLAE$_{\text{MAPPO}}$ demonstrates advantages over RLAE$_{\text{PPO}}$. This superiority stems from several key factors: First, the multi-agent framework allows each agent to specialize in modeling a specific LLM's behavior patterns and output characteristics, leading to more accurate weight assignments. Second, the shared critic in MAPPO enables agents to learn from each other's experiences while maintaining their individual policies, facilitating better coordination when base models have complementary strengths. For example, we observe that RLAE$_{\text{MAPPO}}$ learns to leverage Qwen-2's superior performance on STEM questions, which are shown in Appendix G. These advantages are empirically validated in our three-model ensemble results, where RLAE$_{\text{MAPPO}}$ achieves a 0.3 point improvement over the single-agent RLAE$_{\text{PPO}}$, with particularly high gains on questions requiring diverse domain expertise.

(3) RLAE$_{\text{PPO}}$ Advantage in Code Generation. Notably, RLAE$_{\text{PPO}}$ outperforms RLAE$_{\text{MAPPO}}$ on the MBPP programming benchmark (75.8 versus 75.3 in three-model settings). We attribute this to the nature of code generation tasks, which require maintaining global consistency throughout the generation process. This aligns with findings from traditional reinforcement learning research, where single-agent approaches provide more consistent control and better performance in tasks requiring coordinated actions, as demonstrated in robotics control tasks like MuJoCo simulations (Brockman et al., 2016; Peng et al., 2021).

4.3 Analysis

To comprehensively evaluate RLAE, we conduct extensive analyses focusing on three critical aspects: computational efficiency in terms of time latency, generalization capabilities across different tasks, and ablation studies on key components. All analysis experiments utilize a two-LLM ensemble configuration (Qwen-2 and LLaMA-3.1).

Latency Analysis. In order to evaluate the computational efficiency of our method, we test the latency (ms/token) across different ensemble configurations. As illustrated in Figure 3, our method achieves latency comparable to GaC, which is substantially lower than that of PairRanker and DeePEn. Despite incorporating an additional RL agent with 400M parameters, our method maintains competitive efficiency through span-level ensemble optimization, which reduces the frequency of weight adjustments.

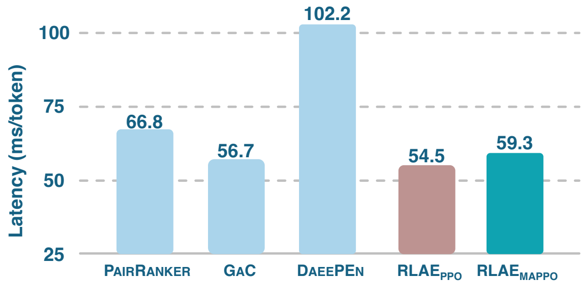


Figure 3: Time latency comparison of different methods.

Dataset	PairRanker	RLAE$_{PPO}$	RLAE$_{MAPPO}$
ARC-C	78.6	82.8	83.4
MMLU → ARC-C	74.8 (-3.8)	**82.2 (-0.6)**	**83.0 (-0.4)**

Table 3: Generalization of RLAE and PairRanker from MMLU to ARC-C.

Generalization of RL Agent. To assess the generalization capability of our method, we conduct a cross-task evaluation by directly applying the RL agent and PairRanker trained on MMLU to ensemble LLMs on ARC-C, a related but distinct task. As shown in Table 3, while PairRanker exhibits significant performance degradation on ARC-C (3.8 points lower compared to direct training), our method demonstrates remarkable generalization with only minimal performance drops (0.6 and 0.4 points for PPO and MAPPO, respectively). These results highlight the generalization capabilities of RLAE across different tasks. More results can be found in Appendix F.

Effect of Ensemble Weights by RL. To evaluate the effectiveness of our RL-based weight generation approach, we conduct comparative experiments against baseline weighting strategies, including uniform weighting and perplexity-based weighting. For perplexity-based weighting, we calculate the perplexity (PPL) score as:

$$\mathrm{PPL}_k(\boldsymbol{x}) = \exp\left(-\frac{1}{m}\sum_{i=1}^{m}\log p_{M_k}(x_i|x_{<i})\right) \tag{7}$$

where lower PPL indicates higher model confidence and receives the higher ensemble weight.

As shown in Figure 4, RLAE outperforms baselines in both single-agent and multi-agent settings. Besides, uniform weighting performs better on MMLU, while perplexity-based weighting achieves superior results on GSM8K. This suggests that previous methods require task-specific weight adjustments, whereas RLAE automatically adapts to different tasks without manual intervention.

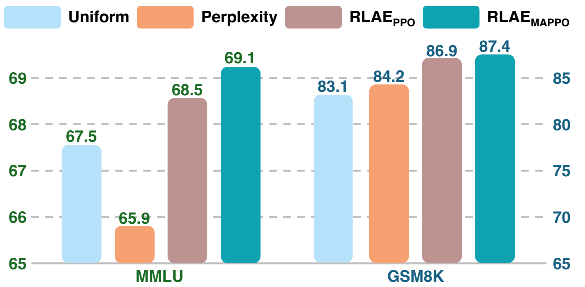
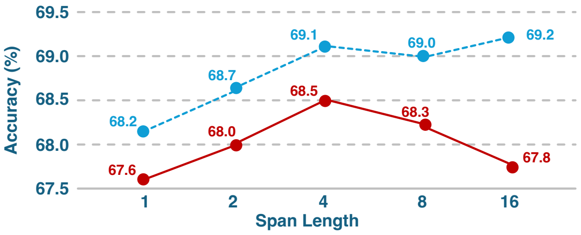
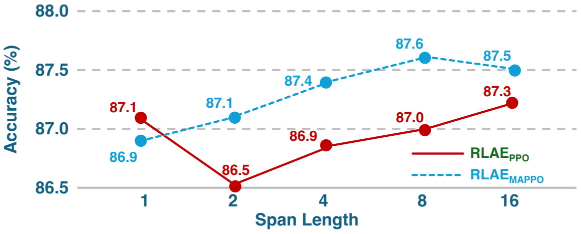


Figure 4: Performance comparison of different weighting methods across tasks.



(a) MMLU



(b) GSM8K

Figure 5: Span length ablation study on different tasks.

Effect of the Span Length. To explore the impact of span length on performance, we conduct an ablation study varying the span length from 1 to 16 and evaluate the performance of different methods. The ablation results are shown in Figure 5, indicating that RLAE$_{PPO}$ is more sensitive to span length compared to RLAE$_{MAPPO}$. Additionally, the impact of span length on performance varies across different tasks. The span length of 4 used in our main experiments proves to be a balanced choice.

5 Conclusion

In this paper, we introduce RLAE, a novel reinforcement learning-assisted framework for LLM ensemble that significantly enhances LLM capabilities by dynamically combining the complementary strengths of different models. By formulating the ensemble problem as a Markov Decision Process at the span level, RLAE adaptively adjusts ensemble weights based on both prompt and responses, enabling flexible and efficient generation. Unlike previous ensemble methods, we employ single-agent and multi-agent RL algorithms to optimize the ensemble process. Extensive experiments demonstrate that our RL-based framework achieves not only substantial improvements but also generalization capabilities across different tasks.

Limitations

Due to the inherent nature of the ensemble, our approach, like other ensemble methods, requires increased computational resources compared to single-model inference. While parallel execution on separate GPUs limits latency to that of the slowest model, the computational demand still scales linearly with the number of models. This creates resource challenges, especially during initial inference. Furthermore, the effectiveness of our method is closely tied to reward design, where current metrics such as accuracy may not comprehensively align with generation quality.

Future work could explore efficient model selection mechanisms that identify an optimal subset of models prior to inference, thereby reducing computational overhead while maintaining ensemble effectiveness. Additionally, other promising research directions include developing reward modeling approaches that better capture generation quality and incorporating human feedback to improve alignment with human preferences.

Acknowledgements

This work was supported in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA0480303, in part by National Natural Science Foundation of China under Grants 62206281, 62293541, and 62136008, in part by Beijing Natural Science Foundation under Grant No. 4232056, and in part by Beijing Nova Program under Grant 20240484514.

References

Anthropic. 2024. Introducing the next generation of Claude.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

Jiajun Chai, Sicheng Li, Yuqian Fu, Dongbin Zhao, and Yuanheng Zhu. 2025. Empowering LLM agents with zero-shot optimal decision-making through q-learning. In *The Thirteenth International Conference on Learning Representations*.

Souradip Chakraborty, Sujay Bhatt, Udari Madhushani Sehwag, Soumya Suvra Ghosal, Jiahao Qiu, Mengdi Wang, Dinesh Manocha, Furong Huang, Alec Koppel, and Sumitra Ganesh. 2025. Collab: Controlled decoding using mixture of agents for LLM alignment. In *The Thirteenth International Conference on Learning Representations*.

Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi Yang, Hailong Sun, and Philip S Yu. 2025. Harnessing multiple large language models: A survey on LLM ensemble. *arXiv preprint arXiv:2502.18036*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

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*.

Evan Frick, Connor Chen, Joseph Tennyson, Tianle Li, Wei-Lin Chiang, Anastasios N Angelopoulos, and Ion Stoica. 2025. Prompt-to-leaderboard. *arXiv preprint arXiv:2502.14855*.

Yuqian Fu, Tinghong Chen, Jiajun Chai, Xihuai Wang, Songjun Tu, Guojun Yin, Wei Lin, Qichao Zhang, Yuanheng Zhu, and Dongbin Zhao. 2025. SRFT: A single-stage method with supervised and reinforcement fine-tuning for reasoning. *arXiv preprint arXiv:2506.19767*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sven Gronauer and Klaus Diepold. 2022. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

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

Shengyi Huang, Rousslan Fernand Julien Dossa, Antonin Raffin, Anssi Kanervisto, and Weixun Wang. 2022. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*.

Yichong Huang, Xiaocheng Feng, Baohang Li, Yang Xiang, Hui Wang, Ting Liu, and Bing Qin. 2024. Ensemble learning for heterogeneous large language models with deep parallel collaboration. *Advances in Neural Information Processing Systems*, 37:119838–119860.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pages 157–163. Elsevier.

Jinliang Lu, Ziliang Pang, Min Xiao, Yaochen Zhu, Rui Xia, and Jiajun Zhang. 2024. Merge, ensemble, and cooperate! a survey on collaborative strategies in the era of large language models. *arXiv preprint arXiv:2407.06089*.

Bo Lv, Chen Tang, Yanan Zhang, Xin Liu, Yue Yu, and Ping Luo. 2024. SpecFuse: Ensembling large language models via next-segment prediction. *arXiv preprint arXiv:2412.07380*.

Costas Mavromatis, Petros Karypis, and George Karypis. 2024. Pack of LLMs: Model fusion at test-time via perplexity optimization. In *First Conference on Language Modeling*.

OpenAI. 2024. GPT-4o system card.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. FACMAC: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221.

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*.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Selim Furkan Tekin, Fatih Ilhan, Tiansheng Huang, Sihao Hu, and Ling Liu. 2024. LLM-TOPLA: Efficient LLM ensemble by maximising diversity. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11951–11966, Miami, Florida, USA. Association for Computational Linguistics.

Songjun Tu, Jiahao Lin, Xiangyu Tian, Qichao Zhang, Linjing Li, Yuqian Fu, Nan Xu, Wei He, Xiangyuan Lan, Dongmei Jiang, and Dongbin Zhao. 2025. Enhancing LLM reasoning with iterative DPO: A comprehensive empirical investigation. In *Second Conference on Language Modeling*.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. 2024. OpenChat: Advancing open-source language models with mixed-quality data. In *The Twelfth International Conference on Learning Representations*.

Feng Xiong, Hongling Xu, Yifei Wang, Runxi Cheng, Yong Wang, and Xiangxiang Chu. 2025. HS-STAR: Hierarchical sampling for self-taught reasoners via difficulty estimation and budget reallocation. *arXiv preprint arXiv:2505.19866*.

Kaixuan Xu, Jiajun Chai, Sicheng Li, Yuqian Fu, Yuanheng Zhu, and Dongbin Zhao. 2025a. DipLLM: Fine-tuning LLM for strategic decision-making in Diplomacy. In *International Conference on Machine Learning*. PMLR.

Yangyifan Xu, Jianghao Chen, Junhong Wu, and Jiajun Zhang. 2025b. Hit the sweet spot! span-level ensemble for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 8314–8325, Abu Dhabi, UAE. Association for Computational Linguistics.

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, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024b. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Yuxuan Yao, Han Wu, Mingyang LIU, Sichun Luo, Xiongwei Han, Jie Liu, Zhijiang Guo, and Linqi Song. 2025. Determine-Then-Ensemble: Necessity of top-k union for large language model ensembling. In *The Thirteenth International Conference on Learning Representations*.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of PPO in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624.

Yao-Ching Yu, Chun Chih Kuo, Ye Ziqi, Chang Yucheng, and Yueh-Se Li. 2024. Breaking the ceiling of the LLM community by treating token generation as a classification for ensembling. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1826–1839, Miami, Florida, USA. Association for Computational Linguistics.

Jusheng Zhang, Yijia Fan, Wenjun Lin, Ruiqi Chen, Haoyi Jiang, Wenhao Chai, Jian Wang, and Keze Wang. 2025a. GAM-Agent: Game-theoretic and uncertainty-aware collaboration for complex visual reasoning. *arXiv preprint arXiv:2505.23399*.

Jusheng Zhang, Zimeng Huang, Yijia Fan, Ningyuan Liu, Mingyan Li, Zhuojie Yang, Jiawei Yao, Jian Wang, and Keze Wang. 2025b. KABB: Knowledge-aware bayesian bandits for dynamic expert coordination in multi-agent systems. In *Forty-second International Conference on Machine Learning*.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P Xing, Hongyi Wang, and Huaxiu Yao. 2025. CITER: Collaborative inference for efficient large language model decoding with token-level routing. *arXiv preprint arXiv:2502.01976*.

Appendix

A Pseudocode of RLAE

We describe the ensemble generation and training process in Algorithm 1 and Algorithm 2.

Algorithm 1 Ensemble Generation for RLAE

```
Input: Prompt $\boldsymbol{x}$, base models $\{M_1, \ldots, M_K\}$, controller policy $\pi_\phi$, span length $L$
Output: Generated response $\boldsymbol{y}$
Initialize $\boldsymbol{y}_0 \leftarrow \emptyset$
for $t = 0, 1, \ldots, \lceil H/L \rceil - 1$ do
    // Construct current state
    $s_t \leftarrow (\boldsymbol{x}, \boldsymbol{z}_{<t})$
    // Sample weights from policy
    $a_t \leftarrow (w_t^{(1)}, \ldots, w_t^{(K)}) \sim \pi_\phi(\cdot|s_t)$
    for $h = tL, \ldots, \min((t+1)L - 1, H)$ do
        Generate ensemble probability by Eq. (1)
        // Sample next token
        $y_h \sim P(y_h|\boldsymbol{x}, \boldsymbol{y}_{<h})$
        $\boldsymbol{z}_{<t+1} \leftarrow \boldsymbol{z}_{<t} \oplus y_h$
    end for
end for
return $\boldsymbol{y}$
```

Algorithm 2 Training Process for RLAE

```
Input: Base models $\{M_1, \ldots, M_K\}$, initial policy $\pi_\phi$, dataset $\mathcal{D}$
Output: Trained policy $\pi_\phi$
for each epoch do
    Initialize buffer $\mathcal{B} \leftarrow \emptyset$
    for prompt $\boldsymbol{x}$ in $\mathcal{D}$ do
        Initialize $\boldsymbol{y}_0 \leftarrow \emptyset$, trajectory $\tau \leftarrow \emptyset$
        for each span $t$ do
            $s_t \leftarrow (\boldsymbol{x}, \boldsymbol{z}_{<t})$, $a_t \sim \pi_\phi(\cdot|s_t)$
            // Sample Spans
            Generate span using Algorithm 1
            Compute reward $r_t$
            Add $(s_t, a_t, r_t)$ to trajectory $\tau$
        end for
        Add $\tau$ to buffer $\mathcal{B}$
    end for
    Update policy using Eq. (3) or Eq. (6)
end for
return $\pi_\phi$
```

B Hyperparameters and Computational Resources

The hyperparameters used in our experiments are shown in Table A1. All experiments are conducted on 8 NVIDIA A100 GPUs.

Hyperparameter	**Explanation**	**Values**
ϵ	Clip range	0.2
γ	Key GAE parameter	0.99
λ	Key GAE parameter	0.95
L	Span length	4
$\lvert B \rvert$	Buffer size	128
`lr`	Maximum learning rate	1e-4
`lr_scheduler`	Learning rate schedule	cosine
`num_epochs`	Number of training epochs	3
`entropy_coef`	Entropy coefficient	0.01

Table A1: Hyperparameters used in our experiments.

C Details of Value Network

The value function (critic) plays a crucial role in reinforcement learning, as it estimates the expected return of the current state. In our work, we adopt a shared network architecture (Huang et al., 2022) where both the value network and policy network utilize the same DeBERTa-v3-large model. Then RLAE builds a value head and a policy head that share the output of the DeBERTa-v3-large model. The value network is trained to minimize the mean squared error (MSE) loss between its predictions and the target values:

$$\mathcal{L}_V = \sum_{(s_t,a_t,r_t)} (V_\phi(s_t) - \hat{V}_t)^2 \qquad \text{(A1)}$$

where $V_\phi(s_t)$ is the value prediction for state s_t, and $\hat{V}_t = \sum_{l=0}^{T-t} \gamma^l r_{t+l}$ is the target value computed by the reward function r_t.

D Vocabulary Mismatch

We adopt the approach proposed in GAC (Yu et al., 2024) to address this vocabulary mismatch, which projects probability vectors from multiple LLMs into a unified vocabulary space through a mapping matrix. Specifically, we construct a comprehensive unified vocabulary set V_u by aggregating tokens from each base model in our ensemble. During the construction of this unified vocabulary, we eliminate any duplicate tokens while preserving all unique tokens present across the different model vocabularies. For tokens that exist in the unified set

V_u but are absent from a particular model's vocabulary V_k, we implement a principled zero-probability assignment strategy, where the generation probability of such tokens from model M_k is explicitly set to 0. This principled approach allows us to aggregate outputs from different models at each generation step to select the next token, ensuring comprehensive coverage of the token space while maintaining proper probability distributions across models with heterogeneous tokenization schemes. Recently, UNITE (Yao et al., 2025) proposes to match the vocabulary by considering only the union of top-k tokens from each model. This approach eliminates the need for full vocabulary alignment and reduces computational overhead, while being complementary to our method.

E Markov Game

For the multi-agent setting, we extend beyond the Markov Decision Process (MDP) framework and formulate the ensemble problem as a Markov Game (MG) (Littman, 1994). A MG generalizes MDP to multiple interacting agents and is formally defined by a tuple $\langle \mathcal{K}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where:

- $\mathcal{K} = \{1, \ldots, K\}$ is the set of agents, with each agent corresponding to ensemble weight.
- $\mathcal{S}$ is the state space, representing the current prompt and generation history.
- $\mathcal{A}$ is the action space, which determines the ensemble weight assigned to the corresponding LLM's output.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability function.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function for all agents.
- $\gamma \in [0, 1]$ is the discount factor.

At each step, all agents observe the current state s_t and simultaneously select actions a_k according to their policies π_k. The environment then transitions to a new state s_{t+1} based on the joint actions, and each agent receives its individual reward r. The goal of each agent is to maximize its expected discounted return:

$$J_k(\pi_k) = \mathbb{E}_{\pi_k, \pi_{-k}} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \quad \text{(A2)}$$

where π_{-k} denotes the joint policy of all agents except agent k. This formulation enables cooperative behavior among agents through the reward design, while allowing each agent to learn specialized ensemble policies based on their corresponding LLM's strengths.

F Additional Experimental Results on Generalization

We provide additional experimental results on the generalization in Tables A2. The results show that the RL-assisted framework can achieve better generalization on different tasks.

Dataset	PAIRRANKER	RLAE$_{\text{PPO}}$	RLAE$_{\text{MAPPO}}$
GPQA	32.7	35.1	34.7
ARC-C → GPQA	27.3 (-5.4)	**33.3 (-1.8)**	**32.6 (-2.1)**

Table A2: Generalization from ARC-C to GPQA.

G Ensemble Weights Visualization

In this section, we provide the visualizations of the ensemble weights across different benchmarks. We first visualize the ensemble weight differences between *different benchmarks*, as shown in Figure A1. The experimental results demonstrate that different benchmarks require distinct ensemble weights. For instance, Llama-3.1-8B-Instruct achieves higher weights on MMLU, while Qwen-2-7B-Instruct obtains higher weights on STEM-related tasks like GPQA and GSM8K. Through reinforcement learning training, our proposed method RLAE can adaptively adjust ensemble weights to achieve better performance across different benchmarks. Then, we provide a demo of the ensemble weights visualization *in a single response*, as shown in Figures A2 and A3. We conducted the visualization on two different prompts, and the experimental results show that RLAE can leverage the advantages of different models by adjusting ensemble weights, thus achieving better performance on different tasks.

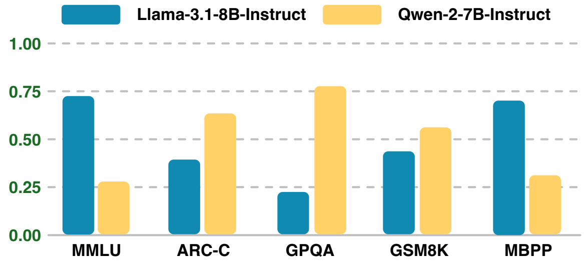


Figure A1: Ensemble weights across different benchmarks. We calculate the average weights across different benchmarks.

Ensemble Weight Visualization Demo 1

Prompt: 1.5 chickens lay 1.5 eggs in 1.5 days. How many eggs will 9 chickens lay in 9 days?
Answer: 54

Llama-3.1-8B-Instruct: Step 1: Determine the rate at which the chickens lay eggs. 1.5 chickens lay 1.5 eggs in 1.5 days, so the rate of egg laying per chicken per day is 1.5 eggs / 1.5 chickens / 1.5 days   1 egg / 1 chicken / 1 day. Step 2: Calculate the number of eggs 1 chicken lays in 9 days. Since 1 chicken lays 1 egg in 1 day, in 9 days, 1 chicken will lay 9 eggs. Step 3: Calculate the total number of eggs 9 chickens will lay in 9 days. Since 1 chicken lays 9 eggs in 9 days, 9 chickens will lay 9   9   81 eggs in 9 days. The final answer is: $\boxed{81}$

Qwen-2-7B-Instruct: To solve the problem, we can start by determining the egg-laying rate of the chickens and then apply that rate to the given conditions. First, we know that 1.5 chickens lay 1.5 eggs in 1.5 days. We can use this information to find out how much one chicken can lay in 1.5 days. Since 1.5 chickens lay 1.5 eggs in 1.5 days, one chicken would lay: $\frac{1 \quad \text{eggs}}{1 \quad \text{chickens}} = 1$ egg in 1.5 days So, one chicken lays 1 egg in 1.5 days. Now, we need to find out how many eggs one chicken lays in one day. Since 1 chicken lays 1 egg in 1.5 days, it will lay: $\frac{1 \text{ egg}}{1 \quad \text{days}} = \frac{2}{}$ egg per day Next, we need to find out how many eggs 9 chickens will lay in 9 days. Since one chicken lays $\frac{2}{}$ egg per day, 9 chickens will lay: $9 \times \frac{2}{}$ eggs per day $= 6$ eggs per day Now, we need to find out how many eggs 9 chickens will lay in 9 days. Since 9 chickens lay 6 eggs per day, in 9 days they will lay: 9×6 eggs $= 54$ eggs So, 9 chickens will lay 54 eggs in 9 days.

RLAE: To solve this problem, we need to determine the rate at which the chickens lay eggs and then use that rate to find out how many eggs 9 chickens will lay in 9 days. First, let's find the rate at which the chickens lay eggs. We know that 1.5 chickens lay 1.5 eggs in 1.5 days. This means that 1 chicken would lay: $\frac{1 \quad \text{eggs}}{1 \quad \text{chickens}} = 1$ egg per chicken in 1.5 days. Next, we need to find out how many eggs 1 chicken lays in 1 day. Since 1 chicken lays 1 egg in 1.5 days, the rate for 1 chicken per day is that: $\frac{1 \text{ egg}}{1 \quad \text{days}} = \frac{2}{}$ eggs per day. Now, we need to find out how many eggs 9 chickens will lay in 9 days. If 1 chicken lays $\frac{2}{}$ eggs per day, then 9 chickens will lay: $9 \times \frac{2}{}$ eggs per day $= 6$ eggs per day. If 9 chickens lay 6 eggs per day, then in 9 days they will lay: 6 eggs per day $\times$ 9 days $= 54$ eggs. Therefore, the number of eggs that 9 chickens will lay in 9 days is: $\boxed{54}$

(**Legend:** The background colors indicate the relative weights of **Llama-3.1-8B-Instruct** and **Qwen-2-7B-Instruct** in the ensemble. Darker shades represent higher weights, showing each model's contribution to the final response.)

Figure A2: Visualization demo 1 of ensemble weights in a single response.

Ensemble Weight Visualization Demo 2

Prompt: A decade ago, the population of a city was 55,182 people. Now, it is 170 larger. What is the city's current population?
Answer: 148,991

Llama-3.1-8B-Instruct: Let's solve this step by step: Step 1: Determine the original population of the city. The original population of the city was $55,182$ people. Step 2: Calculate the increase in population. The city's population is now 170% larger than it was a decade ago. To find the increase, we need to calculate 170% of the original population: 170% of $55,182$ is $\frac{1\ 0}{100} \times 55,182$. Step 3: Calculate 170% of the original population: $\frac{1\ 0}{100} \times 55,182 = 1.7 \times 55,182 = 93,909.4$ Step 4: Calculate the current population of the city. To find the current population, we add the original population to the increase: $55,182 + 93,909.4$ Step 5: Perform the addition: $55,182 + 93,909.4 = 149,091.4$ Step 6: Since population must be a whole number, we round $149,091.4$ to the nearest integer. Therefore, the final answer is: $\boxed{149,091}$

Qwen-2-7B-Instruct: To find the current population of the city, we start with the population from a decade ago and determine how much it has increased. We know that the population is now 170 larger than it was a decade ago. This means the current population is 100 of the original population plus an additional 170 of the original population. Let's denote the population a decade ago as P_{l}. According to the problem, $P_{l} = 55,182$. The current population, P, is given by: $P = P_{l} + 1.7 \times P_{l} = P_{l} \times (1 + 1.7) = P_{l} \times 2.7$ Substituting the value of P_{l}: $P = 55,182 \times 2.7$ Now, we perform the multiplication: $55,182 \times 2.7 = 55,182 \times (2 + 0.7) = 55,182 \times 2 + 55,182 \times 0.7$, $55,182 \times 2 = 110,364$, $55,182 \times 0.7 = 38,627.4$, $110,364 + 38,627.4 = 148,991.4$. Since population is typically expressed as a whole number, we round 148,991.4 to the nearest whole number, which is 148,991. Thus, the current population of the city is 148,991.

RLAE: To find the current population of the city, we need to calculate how much larger the population has become and then add that amount to the original population. First, let's determine what 170 larger means in terms of the original population: $\text{Increase} = \text{Original Population} \times \text{Percentage Increase}/100$. Given: $\text{Original Population} = 55,182$, $\text{Percentage Increase} = 170\%$. Now, plug these values into the formula: $\text{Increase} = 55,182 \times \frac{1\ 0}{100} = 55,182 \times 1.7 = 93,809.4$. Since population numbers must be whole numbers (you can't have a fraction of a person), we'll round this to the nearest whole number: $\text{Increase} \quad 93,809$. Now, add this increase to the original population to get the current population: $\text{Current Population} = \text{Original Population} + \text{Increase} = 55,182 + 93,809 = 148,991$. So, the city's current population is approximately 148,991 people.

(**Legend:** The background colors indicate the relative weights of **Llama-3.1-8B-Instruct** and **Qwen-2-7B-Instruct** in the ensemble. Darker shades represent higher weights, showing each model's contribution to the final response.)

Figure A3: Visualization demo 2 of ensemble weights in a single response.