

Towards Effective Long Video Understanding: Dynamic MAS Construction via Meta-Agent

Jing Huang^{1*}, Lidong Zhang^{1*}, Yadong Li^{2†}, Xingzhong Xu², Siye Chen²,
Jie Liu⁴, Ming Kong^{1,3†}, and Qiang Zhu^{1†}

¹ Zhejiang University, China

² Ant Group, China

³ Zhejiang Key Laboratory of Geographic Information Science, China

⁴ City University of Hong Kong, China
{huangjin9, zhuq}@zju.edu.cn

Abstract. Long video understanding is fundamentally constrained by massive data volume and high information density. While agent-based methods mitigate context length bottlenecks, they predominantly rely on static, "one-size-fits-all" multi-agent systems (MAS). These rigid architectures are computationally redundant and lack the flexibility to handle diverse, instance-level reasoning demands. To address this, we propose DyMAC (**D**ynamic **M**AS **C**onstruction), a novel framework that dynamically generates instance-specific MAS for each video-question pair. By logically abstracting the MAS into a directed computational graph via automated code generation, DyMAC exhibits exceptional adaptability, seamlessly tailoring its collaborative architecture to the specific complexity of each input. Furthermore, we employ a streamlined two-stage training strategy—Supervised Fine-Tuning (SFT) and Reinforcement Learning with Verifiable Rewards (RLVR)—to enhance the model’s dynamic construction and decision-making capabilities. Extensive experiments across four mainstream benchmarks demonstrate that DyMAC achieves a new state-of-the-art, outperforming the second-best method by 3.3% on average. Notably, it significantly improves inference efficiency compared to existing static approaches, further highlighting its superior adaptability and robustness in complex video scenarios.

Keywords: Long Video Understanding · Dynamic Multi-Agent System
· Instance-Specific Architecture

1 Introduction

The massive data volume and high information burden of long videos pose significant context-length bottlenecks for existing large multimodal models. To address

* Equal contribution.

† Corresponding authors.

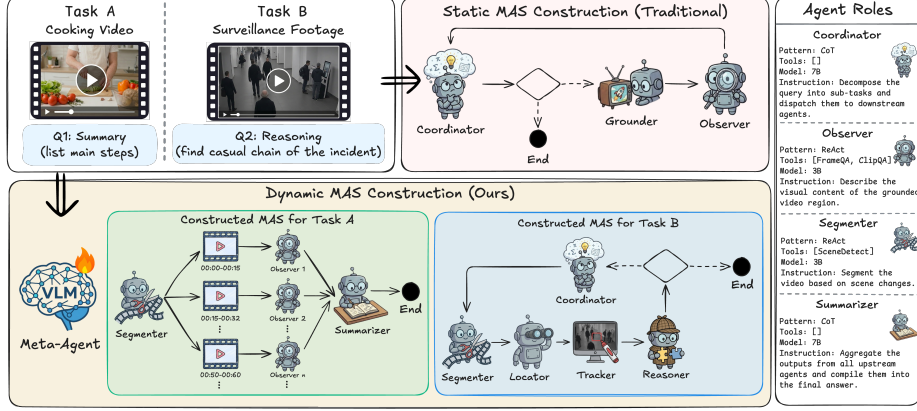


Fig. 1: Comparison between traditional static MAS and our proposed Dynamic MAS Construction (DyMAC). For diverse video understanding tasks, traditional static MAS relies on a fixed, one-size-fits-all architecture. In contrast, DyMAC utilizes a Meta-Agent to adaptively allocate agent roles and construct optimal workflows based on specific task characteristics. As illustrated, DyMAC generates a parallel architecture for summarization tasks (Task A) to enhance efficiency, and constructs a loop-based structure for complex reasoning tasks (Task B) to facilitate multi-step logic.

this, recent agent-based methods utilize large language models as control centers to decompose global video understanding into local sub-tasks via external tools [3, 9, 32], keyframe sampling [21, 23, 29], or iterative QA [12, 15, 17, 36], effectively mitigating memory constraints and information overload [11, 14, 26, 30].

Despite these advances, traditional paradigms predominantly rely on manually constructed, static Multi-Agent Systems (MAS) [10] (Figure 1, top right). To accommodate the most demanding spatio-temporal reasoning cases, these "one-size-fits-all" workflows are often overly complex and rigid. Consequently, they force redundant global analyses even for simple clip retrieval or local QA. This rigidity not only wastes computational resources but also limits the model's flexibility to address diverse, multi-level video understanding demands.

To overcome these limitations, and inspired by recent advances in automated agentic system design [8, 28, 35], we propose DyMAC (Dynamic MAS Construction), a dynamic framework tailored for long video understanding. Unlike static paradigms, DyMAC employs a Meta-Agent (a Vision-Language Model) to dynamically generate an instance-specific MAS for each "video-question" pair. Specifically, DyMAC models the MAS as a directed computational graph. Nodes represent agent roles defined by four meta-attributes (Instruction, Driving Model, Available Tools, and Reasoning Pattern [6, 13, 18, 22, 24, 27]), while edges dictate information flow and control logic. As shown in Figure 1 (bottom), the Meta-Agent adaptively constructs optimal workflows—such as parallel architectures for summarization or loop-based structures for complex reasoning—and executes them via automated code generation.

We optimize DyMAC through a two-stage training strategy. First, Supervised Fine-Tuning (SFT) on a synthesized (Video, Question, MAS code) corpus [31,33] distills system architectural thinking and coding syntax into the model. Second, Reinforcement Learning with Verifiable Rewards (RLVR) [5, 16, 19] further refines its decision-making, balancing reasoning accuracy against execution cost. By accurately capturing the context and difficulty of each specific "video-query" pair, DyMAC’s Meta-Agent dynamically builds tailored collaborative architectures. This adaptive mechanism effectively overcomes the limitations of static workflows, enabling flexible resolution of multi-scale challenges—from global summarization to fine-grained QA—while demonstrating robust cross-domain generalization.

To evaluate the effectiveness of the proposed method, we comprehensively benchmark DyMAC on four widely used long video understanding datasets: Video-MME [7], MLVU [34], LongVideoBench [25], and LVBench [20]. Experimental results demonstrate that, under identical or similar model parameter conditions (e.g., Qwen-VL series [1,2]), DyMAC achieves state-of-the-art (SoTA) performance. Averaging across all four benchmarks, DyMAC outperforms the existing second-best method by 3.3%. Furthermore, compared to other agent-based methods, DyMAC significantly reduces average inference time. These results prove the substantial advantages of Dynamic MAS Construction in both effectiveness and efficiency over traditional static MAS. Meanwhile, qualitative analyses further indicate that DyMAC possesses strong scene adaptability, capable of constructing structurally diverse agent collaboration networks based on flexible input instructions.

In summary, our main contributions are as follows:

- We introduce the concept of Dynamic MAS Construction into the field of long video understanding, proposing the **DyMAC** framework. Driven by a Meta-Agent, this framework dynamically generates an instance-specific MAS based on a particular input, demonstrating significant adaptability compared to traditional static architectures.
- We innovatively represent the MAS logically as a heterogeneous directed computational graph. Through rich node configurations (Instruction, Driving Model, Available Tools, and Reasoning Pattern) and topological designs, we achieve the flexible and efficient construction of instance-specific systems. Concurrently, we propose a two-stage training strategy (SFT + RLVR) that effectively enhances the model’s capability to solve complex reasoning tasks.
- We conduct comprehensive evaluations across four mainstream long video understanding benchmarks. DyMAC not only achieves a new SoTA by outperforming the second-best method by 3.3% on average, but it also realizes effective optimizations in inference efficiency compared to existing agent-based methods, fully validating the superiority of the proposed paradigm.

2 Related Work

2.1 Agent-based Long Video Understanding

A fundamental challenge in long video understanding has been overcoming the inherent context limitations of Large Language Models (LLMs). Initial efforts primarily focused on mitigating information overload through external memory augmentation and structural reformulation. For instance, DrVideo [12] circumvents context length by transforming video understanding into textual retrieval via coarse transcriptions, while VideoAgent [21] relies on unified memory and continuous object tracking. To handle increasingly complex spatiotemporal queries, subsequent research evolved from linear representations to hierarchical topologies. Methods such as VideoTree [23] and VideoLucy [36] construct adaptive tree structures to prune irrelevant frames, whereas Vgent [17] maps visual entities into dynamic graph networks to enable structured, open-source reasoning. As task complexity grew, a dominant trend shifted from single-agent structural parsing to collaborative multi-agent exploration. By decomposing long-horizon reasoning into specialized subtasks, these frameworks drastically improve adaptability. Approaches range from orchestrating multi-granular search tools (Deep Video Discovery [32]) and applying a "MapReduce" paradigm to aggregate local clip perceptions (MR. Video [15]), to facilitating dynamic, multi-round debates among multimodal LLMs (LVAgent [3]). Master-slave architectures like LongVideoAgent [9] have also emerged to centralize coordination among specialized vision agents. Most recently, to transcend the unreliability of training-free prompting, the frontier has moved toward active agent optimization. Frameworks like VideoExplorer [29] utilize reinforcement learning and preference optimization to fine-tune decision-making policies, rewarding faithful reasoning paths while penalizing flawed tool-use trajectories. Departing from these approaches that optimize policies within fixed agent topologies, our proposed DyMAC dynamically generates an entirely instance-specific MAS architecture for each video-query pair, thereby overcoming structural rigidity and achieving superior adaptability for diverse spatio-temporal reasoning demands.

2.2 Automated and Dynamic MAS Construction

The deployment of Multi-Agent Systems (MAS) traditionally relied on hand-designed, static communication topologies, fundamentally restricting their adaptability to diverse or unseen tasks. To overcome this rigidity, early dynamic frameworks like Dynamic LLM-Agent Network [10] and GPTSwarm [35] introduced optimizable agent graphs capable of iterative task routing. This conceptual shift catalyzed the broader Automated Design of Agentic Systems (ADAS) [8], a paradigm where a central "Meta Agent" searches and programs novel agentic workflows directly within a Turing-complete code space. Despite their high flexibility, a critical limitation of these dynamic and automated paradigms is their prohibitive computational overhead. Because they inherently require multiple, expensive LLM inferences during execution to evaluate and adjust the system

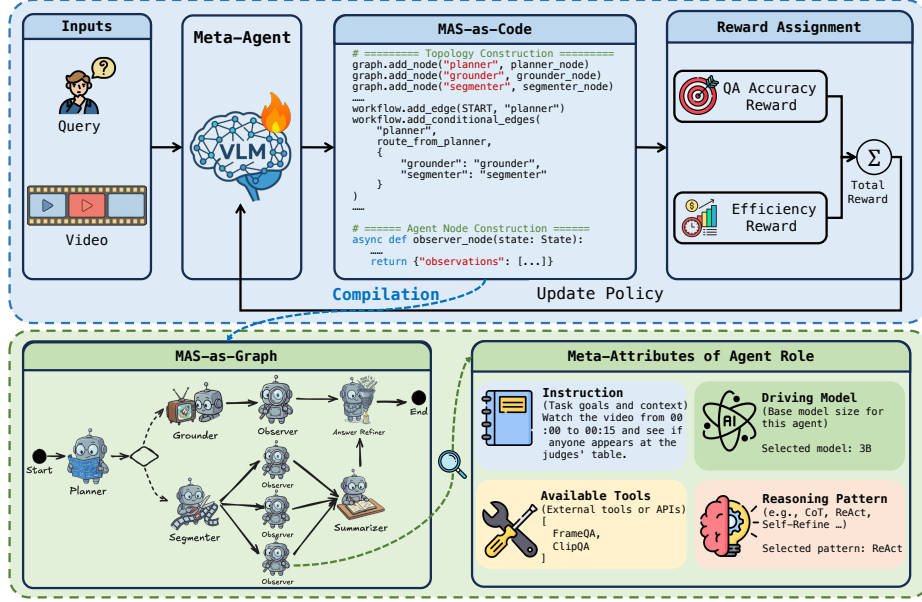


Fig. 2: Overview of the DyMAC framework. The Meta-Agent takes a video and query to generate a multi-agent system as code (MAS-as-Code), which is compiled into a directed computational graph (MAS-as-Graph). Each node represents an Agent Role defined by four meta-attributes: Instruction, Driving Model, Available Tools, and Reasoning Pattern. Finally, the instantiated MAS is evaluated to compute a total reward (QA Accuracy and Efficiency) used to update the Meta-Agent’s policy.

iteratively, they are often impractical for real-time or resource-constrained applications. To resolve this efficiency bottleneck, the latest research reformulates dynamic MAS construction from an iterative search into a single-pass generation problem. A prime example is MAS-GPT [28], which introduces a "MAS-as-Code" generative framework. By directly generating complete agent workflows in a single forward pass, such methods successfully bypass costly multi-round LLM interactions, pointing toward a new standard for efficient and scalable automated agent deployment. While such single-pass generative methods significantly enhance general deployment efficiency, DyMAC distinctly advances this paradigm for multimodal video understanding by dynamically routing heterogeneous driving models and reasoning patterns, utilizing RLVR to explicitly optimize the trade-off between predictive accuracy and systemic execution cost.

3 Methodology

In this section, we present the **DyMAC** (Dynamic MAS Construction) framework, designed to address the inherent spatial-temporal complexities of long video understanding. The remainder of this section is organized as follows: We

first provide an overview of the DyMAC architecture in Sec. 3.1. Next, we formalize the dynamic generation process and the logical compilation of the MAS driven by the central Meta-Agent in Sec. 3.2. Finally, we detail the progressive two-stage optimization paradigm, encompassing Supervised Fine-Tuning (SFT) and Reinforcement Learning with Verifiable Rewards (RLVR), in Sec. 3.3.

3.1 Overview

As illustrated in Figure 2, the core workflow of the DyMAC framework diverges significantly from static multi-agent paradigms. Given a long video and a corresponding user query, the system utilizes a Vision-Language Model (VLM) functioning as a central Meta-Agent to evaluate the task’s specific reasoning demands. Rather than directly predicting the final response, the Meta-Agent synthesizes an executable programmatic representation, termed *MAS-as-Code*. This code explicitly defines the topological structure and node configurations of a customized Multi-Agent System (MAS). Subsequently, the MAS-as-Code is deterministically compiled into a directed computational graph (*MAS-as-Graph*), wherein individual nodes signify specialized agent roles, and directed edges establish the execution workflow and information routing protocols. To achieve an optimal equilibrium between predictive accuracy and systemic efficiency, the Meta-Agent’s policy is refined through a progressive two-stage training strategy: Supervised Fine-Tuning (SFT) followed by Reinforcement Learning with Verifiable Rewards (RLVR) optimized via the Group Relative Policy Optimization (GRPO) algorithm.

3.2 Dynamic MAS Construction via Meta-Agent

MAS-as-Code. Formally, let V denote the input long video and Q denote the accompanying user query. We define the joint multimodal input as $\mathbf{X} = (V, Q)$. The Meta-Agent, parameterized by θ with a policy denoted as π_θ , functions as the orchestration controller. To construct the MAS dynamically, the Meta-Agent decodes a sequence of tokens comprising an executable standard code block C . This generative procedure is modeled as:

$$C \sim \pi_\theta(\cdot \mid \mathbf{X}) \quad (1)$$

The generated code sequence C programmatically specifies the graph topology, effectively translating complex reasoning requirements into a structured, logical formulation.

MAS-as-Graph. Following generation, the code block C is compiled to instantiate the MAS as a heterogeneous directed computational graph, formulated as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ denotes the set of dynamically instantiated agent nodes (incorporating functional roles such as Planners, Grounders, Segmenters, and Observers), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of directed edges

that dictate the conditional execution logic and data dependencies among the constituent agents.

To guarantee functional diversity and adaptability, each agent node $v_i \in \mathcal{V}$ is uniquely characterized by four meta-attributes, formally defined as a tuple:

$$v_i = \langle \mathcal{I}_i, \mathcal{M}_i, \mathcal{T}_i, \mathcal{P}_i \rangle \quad (2)$$

The components of this tuple are defined as follows:

- **Instruction** ($\mathcal{I}_i$): The customized system prompt that delineates the specific task objective and contextual constraints for the i -th agent.
- **Driving Model** ($\mathcal{M}_i$): The specific foundation model allocated to power the node, routed according to the computational complexity of the assigned sub-task.
- **Available Tools** ($\mathcal{T}_i$): The discrete set of external tools or Application Programming Interfaces (APIs) that the agent is authorized to invoke for auxiliary information retrieval.
- **Reasoning Pattern** ($\mathcal{P}_i$): The cognitive paradigm designated for the node, encompassing strategies such as Chain-of-Thought, ReAct, or Self-Refinement.

Upon the complete instantiation of $\mathcal{G}$, the execution flow is initiated at a designated start node and propagates through the intermediate sub-agents. The process concludes when a terminal node consolidates the accumulated intermediate observations to formulate the final predictive output.

3.3 Training Strategy

Supervised Fine-Tuning. To construct the demonstration corpus $\mathcal{D}_{SFT}$, we select videos exceeding 20 minutes from the CG-Bench [4] dataset. For each multimodal input $\mathbf{X}$, we employ Qwen3-VL-235B-A22B-Instruct [1] at a decoding temperature of 1.0 to synthesize 8 distinct MAS-as-Code candidates, denoted as $\{C_m\}_{m=1}^8$. Upon compiling and executing these candidates, we strictly retain input instances where at least 6 out of the 8 instantiated graphs yield predictions consistent with the ground truth. To guarantee topological diversity, we select the optimal candidate C^* from the correct subset of each retained instance by maximizing structural complexity, which is quantified via Abstract Syntax Tree (AST) parsing. This robust pipeline yields a high-quality SFT dataset $\mathcal{D}_{SFT}$ consisting of 3,519 samples, featuring structurally diverse graphs and an average video length of 28.1 minutes.

Given the constructed demonstration dataset $\mathcal{D}_{SFT} = \{(\mathbf{X}^{(j)}, C^{*(j)})\}_{j=1}^{|\mathcal{D}_{SFT}|}$, we initialize the Meta-Agent’s syntactic generation proficiency and fundamental graph-construction capabilities via Supervised Fine-Tuning. The optimization objective employs the standard auto-regressive next-token prediction loss over the optimal candidates:

$$\mathcal{L}_{SFT}(\theta) = -\mathbb{E}_{(\mathbf{X}, C^*) \sim \mathcal{D}_{SFT}} [\log \pi_\theta(C^* \mid \mathbf{X})] \quad (3)$$

This stage effectively distills the core principles of multi-agent orchestration and programmatic syntax into the Meta-Agent.

RLVR Optimization via GRPO. While SFT ensures grammatical validity and establishes basic topological priors, it does not strictly optimize the decision-making process concerning the trade-off between reasoning efficacy and computational expenditure. To address this, we introduce the Reinforcement Learning with Verifiable Rewards (RLVR) mechanism. To construct the training distribution for this phase, denoted as $\mathcal{D}_{RL}$, we continue to sample videos exceeding 20 minutes from the CG-Bench dataset. For each instance, we employ Qwen2.5-VL-7B-Instruct to perform inference, generating 8 sampled trajectories. To ensure a meaningful learning signal, we filter out trivial instances where all 8 sampled paths yield correct predictions. From the remaining subset, we randomly sample 4,000 instances to formally establish $\mathcal{D}_{RL}$.

We formulate a joint reward function $\mathcal{R}(C, \mathbf{X})$ to evaluate the quality of the instantiated MAS-as-Graph. The cumulative reward is defined as a linearly weighted combination of two distinct components:

$$\mathcal{R}(C, \mathbf{X}) = \lambda_1 \mathcal{R}_{acc}(C, \mathbf{X}) + \lambda_2 \mathcal{R}_{eff}(C) \quad (4)$$

where λ_1 and λ_2 serve as scaling coefficients. The Accuracy Reward, $\mathcal{R}_{acc} \in \{0, 1\}$, functions as a verifiable objective metric, which is assigned a value of 1 if the terminal node’s output aligns with the ground truth, and 0 otherwise.

Conversely, $\mathcal{R}_{eff}$ signifies the Efficiency Reward, formulated to explicitly penalize computational overhead and redundant topological complexity. For each instantiated node $v_i \in \mathcal{V}$ in the compiled graph $\mathcal{G}$, we compute its execution cost as $c(v_i) = w(\mathcal{M}_i) \cdot t_i$. Here, t_i aggregates the total token consumption of the node (encompassing input, output, and tool-invocation tokens), and $w(\mathcal{M}_i)$ is a pre-defined weight coefficient directly proportional to the parameter scale of the assigned driving model $\mathcal{M}_i$. The overarching efficiency penalty of the generated code C , denoted as $P_{eff}(C)$, is determined by the maximum cumulative execution cost along any path within the graph:

$$P_{eff}(C) = \max_{p \in \mathcal{P}} \sum_{v_i \in p} c(v_i) \quad (5)$$

where $\mathcal{P}$ represents the set of all valid root-to-leaf execution paths in $\mathcal{G}$. During the optimization phase, for a given input $\mathbf{X}$ and a sampled group of K distinct candidates $\{C_k\}_{k=1}^K$, we map these penalties to a standard $[0, 1]$ interval. The final efficiency reward for the k -th sample is yielded by inverting the intra-group min-max normalized penalty:

$$\mathcal{R}_{eff}(C_k) = 1 - \frac{P_{eff}(C_k) - \min_j P_{eff}(C_j)}{\max_j P_{eff}(C_j) - \min_j P_{eff}(C_j) + \eta} \quad (6)$$

where $j \in \{1, 2, \dots, K\}$, and η is a trivially small constant applied to prevent zero division. This formulation guarantees that within any sampled group, the most computationally efficient architecture receives the maximal efficiency reward of 1.

To ensure stable policy optimization for the Meta-Agent, we utilize Group Relative Policy Optimization (GRPO) [16]. Given the established training distribution $\mathcal{D}_{RL}$, the relative advantage A_k for the k -th sampled code C_k is calculated by standardizing the joint rewards within the specific group:

$$A_k = \frac{\mathcal{R}(C_k, \mathbf{X}) - \mu_{\mathcal{R}}}{\sigma_{\mathcal{R}} + \eta} \quad (7)$$

where $\mu_{\mathcal{R}} = \frac{1}{K} \sum_{j=1}^K \mathcal{R}(C_j, \mathbf{X})$ and $\sigma_{\mathcal{R}}$ represent the empirical mean and standard deviation of the joint rewards across the K samples, respectively, and η provides numerical stability when $\sigma_{\mathcal{R}} \rightarrow 0$. The policy parameter θ is iteratively updated by maximizing the following objective function:

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}_{\mathbf{X} \sim \mathcal{D}_{RL}, \{C_k\}_{k=1}^K \sim \pi_{\theta_{old}}} \left[\frac{1}{K} \sum_{k=1}^K \min \left(r_k(\theta) A_k, \text{clip}(r_k(\theta), 1 - \epsilon, 1 + \epsilon) A_k \right) - \beta \mathbb{D}_{KL}(\pi_{\theta}(\cdot | \mathbf{X}) \parallel \pi_{ref}(\cdot | \mathbf{X})) \right] \quad (8)$$

where $r_k(\theta) = \frac{\pi_{\theta}(C_k | \mathbf{X})}{\pi_{\theta_{old}}(C_k | \mathbf{X})}$ denotes the probability ratio, ϵ is the clipping hyperparameter bounding the policy update step, π_{ref} represents the fixed reference policy utilized to constrain excessive distributional shifts, and β is the coefficient scaling the Kullback-Leibler (KL) divergence penalty. Through this formalized objective, the Meta-Agent is trained to dynamically construct instance-specific architectures that jointly maximize predictive accuracy and systemic efficiency.

4 Experiment

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate our model on four long-term video QA benchmarks. Video-MME [7] comprises 900 videos (averaging 1,018 seconds) and 2,700 QA pairs, evaluated by accuracy without subtitles. MLVU [34] contains 1,730 videos (averaging 930 seconds) and 3,102 questions across nine categories, assessed via the M-Avg metric. LongVideoBench (LVB) [25] includes 3,763 videos (averaging 473 seconds) and 6,678 subtitled QA pairs, where we report accuracy on the validation split. Finally, LVBench [20] targets six core temporal capabilities using 103 exceptionally long videos (averaging 4,101 seconds) and 1,549 QA pairs, measured by standard accuracy.

Implementation Details. By default, the Meta-Agent is Qwen2.5-VL-7B-Instruct [2], initialized with 32 video frames and prompted to generate Lang-Graph code to construct the MAS. For single-agent configurations, we use Qwen2.5-VL-3B-Instruct or 7B-Instruct as driving models. These agents operate on six predefined reasoning paradigms from the Qwen-Agent framework: Chain-of-Thought [24], ReAct [27], Plan-and-Execute [18], Self-Refine [13], LLM-Debate [6],

Table 1: Performance comparison with other methods across various benchmarks.

Method	Model Size	Video-MME	MLVU	LVBench	LVB.	Avg.	Time(S)
VideoLucy [36]	7B + 671B	72.5	76.1	58.8	-	-	19.9
<i>End-to-End MLLMs</i>							
Vanilla	7B	65.1	68.8	45.3	56.0	58.8	<u>1.5</u>
LLaVA-Video [33]	7B	63.3	70.8	-	58.2	-	-
Rewatch-R1 [31]	7B	65.6	-	43.3	-	-	-
VideoThinker [19]	7B	53.7	54.8	59.7	48.9	54.3	-
<i>Multi-turn Tool Invocation</i>							
Conan [14]	7B	60.5	63.4	39.2	56.6	54.9	-
LongVT [26]	7B	64.3	-	41.3	-	-	-
Video-Zoomer [5]	7B	65.2	-	41.5	57.7	-	-
Video-o3 [30]	7B	66.5	72.1	47.6	<u>60.5</u>	<u>61.7</u>	0.9
<i>Agent-Based</i>							
ReAct [27]	7B	68.0	71.2	47.5	57.5	61.1	8.3
VideoRAG [11]	-	62.1	<u>72.4</u>	-	58.7	-	20.8
Vgent [17]	7B	<u>68.9</u>	72.1	-	59.7	-	24.1
VideoExplorer [29]	32B/7B	-	58.6	51.4	-	-	-
Ours	7B	70.6	73.2	<u>55.6</u>	60.7	65.0	4.6

and Self-Consistency [22]. The Meta-Agent’s training pipeline is built upon the `verl` and `vLLM` frameworks. All experiments are conducted on $8 \times$ A100 GPUs, with detailed hyperparameters provided in the supplement.

4.2 Main Comparison

Table 1 provides a comprehensive comparison between our proposed method and existing state-of-the-art (SOTA) approaches across several long-video understanding benchmarks. Our method consistently achieves superior performance, particularly on Video-MME, MLVU, and LVB. Notably, our approach attains an average accuracy of 65.0%, surpassing the runner-up (Video-o3) by a significant margin of 3.3 percentage points. In terms of computational efficiency, following the evaluation protocol of Vgent, our method records an average inference time of 4.6s per minute of video. This represents a substantial reduction in latency compared to prominent agent-based models such as ReAct (8.3s), VideoRAG (20.8s), and Vgent (24.1s), demonstrating an optimal trade-off between predictive accuracy and inference speed.

4.3 Ablation Study

To comprehensively evaluate the individual contributions of the core components in DyMAC, we conduct detailed ablation studies. We systematically analyze the training strategy, reward configurations, dynamic model allocation, and the diversity of reasoning patterns.

Table 2: Performance Comparison of Different Training Strategies and Modality.

Method	MLVU	Video-MME	LVB.	Avg.
Vanilla	68.8	65.1	56.0	63.3
MAS-Text	55.3	50.8	46.6	50.9
+SFT	67.7	66.0	55.6	63.1
+RL	69.5	67.8	56.0	64.4
MAS-Visual	54.3	51.2	48.0	51.2
+SFT	<u>70.9</u>	<u>68.5</u>	<u>57.2</u>	<u>65.5</u>
+RL	73.2	70.6	60.7	68.2

Impact of Training Strategies and Modality. As shown in Table 2, deploying an untrained dynamic MAS results in a significant performance drop compared to the Vanilla baseline. This degradation is primarily due to a massive number of code compilation errors generated by the unoptimized Meta-Agent. Here, MAS-Text denotes that the Meta-Agent only receives the user query as input to construct the system, whereas MAS-Visual indicates it receives both the video and the query. Implementing Supervised Fine-Tuning (+SFT) substantially recovers and improves performance by resolving these syntax and compilation issues. Furthermore, the MAS-Visual configuration consistently outshines its MAS-Text counterpart. This clearly demonstrates that in the domain of long video understanding, providing the Meta-Agent with prior knowledge of the video content is crucial for designing and constructing an effective MAS architecture. Ultimately, applying Reinforcement Learning (+RL) yields the best average performance (68.1%), confirming that our two-stage SFT+RLVR optimization is essential for unlocking the full potential of dynamic MAS.

Table 3: Performance Comparison of Different Reward Settings

Reward Settings	MLVU	Video-MME	Time (S)
Accuracy + Efficiency	73.2	70.6	4.6
Accuracy	72.7	70.0	13.2
–	70.9	68.5	15.5

Effectiveness of Reward Design. Table 3 investigates the impact of the joint reward function during the RLVR stage. Relying solely on the Accuracy reward improves task performance over the SFT baseline (e.g., from 70.9% to 72.7% on MLVU) but incurs high inference latency (13.2s). By integrating the Efficiency penalty alongside Accuracy, the model maintains high predictive performance (73.2% on MLVU) while drastically reducing the average inference time to 4.6s. This confirms that our composite reward formulation successfully guides

the Meta-Agent to construct streamlined, non-redundant computational graphs without sacrificing reasoning quality.

Table 4: Performance Comparison of Different Driving Models

Driving Model	MLVU	Video-MME	Time (S)
3b	69.7	66.9	3.6
7b	73.5	70.5	9.8
3b/7b	73.2	70.6	4.6

Dynamic Model Allocation. The flexibility of the Driving Model ($\mathcal{M}_i$) attribute is ablated in Table 4. Using exclusively a smaller 3B model ensures rapid inference (3.6s) but limits reasoning accuracy (69.7% on MLVU). Conversely, relying entirely on a 7B model maximizes accuracy but significantly increases computational overhead (9.8s). The dynamic allocation strategy (3b/7b), which routes sub-tasks based on their instance-level complexity, achieves an optimal balance. It matches the superior accuracy of the 7B configuration while keeping the inference time (4.6s) highly competitive.

Table 5: Performance Comparison of Different Reasoning Patterns

Reasoning Patterns	MLVU	Video-MME	LVB.	Avg.
ReAct	69.8	67.5	56.3	64.5
ReAct, CoT	71.0	68.2	57.9	65.7
ReAct, CoT, LLM-Debate	71.8	69.1	58.8	66.6
All Six Patterns	73.2	70.6	60.7	68.2

Influence of Reasoning Patterns. Finally, Table 5 evaluates the significance of cognitive diversity by incrementally expanding the available Reasoning Patterns ($\mathcal{P}_i$). Starting with only the ReAct paradigm yields a baseline average of 64.5%. Adding Chain-of-Thought (CoT) and LLM-Debate progressively increases the average accuracy to 65.7% and 66.6%, respectively. Providing the Meta-Agent with the complete set of six reasoning patterns further elevates the overall performance to 68.2%. This steady improvement validates that a rich repertoire of cognitive strategies empowers the Meta-Agent to flexibly construct the most effective reasoning paths for diverse, long-horizon video queries.

Comparison with Static Agent Systems. Fig. 3 compares DyMAC against several established static agent paradigms across multiple benchmarks. Since various state-of-the-art methods often utilize disparate toolsets, we conduct this

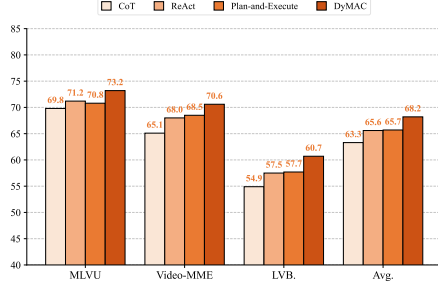


Fig. 3: Performance Comparison between DyMAC and Static Agent Systems.

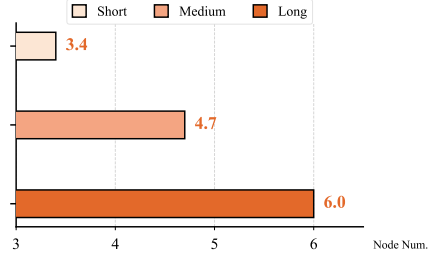


Fig. 4: Node Counts of DyMAC-structured MAS on Various Video-MME Sets.

controlled experiment to isolate the impact of the agent architecture itself. As shown, DyMAC consistently outperforms static configurations such as CoT, ReAct, and Plan-and-Execute. Notably, while the best static baseline (Plan-and-Execute) achieves an average accuracy of 65.7%, DyMAC elevates this to 68.2%. This consistent lead across MLVU, Video-MME, and LVB benchmarks underscores that the dynamic construction of Multi-Agent Systems (MAS) is inherently more robust than relying on a fixed reasoning structure for diverse video tasks.

Adaptability in MAS Construction. To evaluate the adaptability of DyMAC in handling varying task complexities, we analyze the scale of the generated MAS across different Video-MME subsets in Fig. 4. The results reveal a clear positive correlation between video duration and the number of reasoning nodes. For "Short" video clips, DyMAC constructs a concise MAS with an average of 3.4 nodes. As the temporal horizon extends to "Medium" and "Long" videos, the complexity of the MAS naturally scales to 4.7 and 6.0 nodes, respectively. This trend validates that the Meta-Agent can effectively perceive the difficulty of the query and dynamically allocate more cognitive resources (nodes) to long-horizon scenarios, ensuring efficiency without sacrificing reasoning depth.

4.4 Qualitative Analysis

We present a qualitative example of our dynamic MAS execution process in Figure 5. The query asks for the color of the shoes worn by a man during his run. Relying on a traditional static pipeline might lead to incorrect predictions, as the model could easily be distracted by motion blur during the fast-paced running phase or fail to correlate it with clearer, earlier frames. As illustrated, our Meta-Agent effectively mitigates this by dynamically compiling an instance-specific "Fan-out & Fan-in" graph architecture (*MAS-as-Graph*).

Initially, a Temporal Locator (Node I) identifies and segments the relevant video timestamps. Recognizing the multi-stage nature of the video, the system

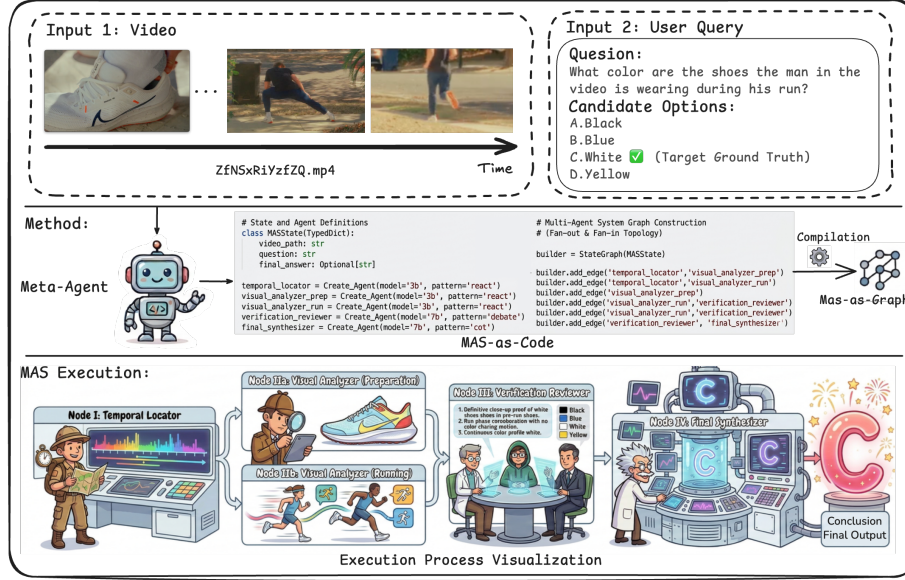


Fig. 5: A qualitative example of DyMAC. Tailored to the current video query, DyMAC dynamically constructs a highly effective MAS. Adopting a "fan-out & fan-in" topology, this customized MAS is highlighted by its parallel retrieval using a Visual Analyzer (Preparation) and a Visual Analyzer (Running). Such an on-demand design endows DyMAC with high effectiveness and adaptability.

dynamically branches out to parallel Visual Analyzers (Nodes IIa and IIb) to independently process the pre-run preparation phase and the active running phase. Crucially, to resolve potential visual ambiguities, a Verification Reviewer (Node III) utilizing an LLM-Debate reasoning pattern aggregates these parallel findings. It successfully cross-references the definitive close-up evidence of the white shoes from the preparation phase with the continuous, motion-blurred visual profile of the running phase. This dynamic, multi-step verification forms an enhanced and robust reasoning chain, allowing the Final Synthesizer (Node IV) to accurately deduce the ground truth option "C" (White), successfully overcoming the challenge of fine-grained visual recognition in complex temporal sequences.

5 Conclusion

In this paper, we introduced DyMAC, a novel framework that advances long video understanding by transitioning from rigid, static multi-agent systems to dynamic, instance-specific architectures. Driven by a central Meta-Agent, DyMAC innovatively formulates multi-agent orchestration as a code generation task, compiling adaptive computational graphs tailored to the specific spatial-temporal reasoning demands of each video-query pair. Furthermore, our proposed two-stage optimization strategy—combining Supervised Fine-Tuning with

Reinforcement Learning with Verifiable Rewards (RLVR)—effectively aligns the model to balance reasoning accuracy against execution efficiency. Extensive evaluations across four mainstream benchmarks demonstrate that DyMAC establishes a new state-of-the-art, significantly outperforming existing methods while drastically reducing inference latency. Ultimately, DyMAC provides a highly flexible, scalable, and robust paradigm for complex reasoning in multimodal applications.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grants 42394060 and 42394064, and by Ant Group.

References

1. Bai, S., Cai, Y., Chen, R., Chen, K., Chen, X., Cheng, Z., Deng, L., Ding, W., Gao, C., Ge, C., et al.: Qwen3-vl technical report. arXiv preprint arXiv:2511.21631 (2025)
2. Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., Lin, J.: Qwen2.5-vl technical report. arXiv preprint arXiv:2502.13923 (2025)
3. Chen, B., Yue, Z., Chen, S., Wang, Z., Liu, Y., Li, P., Wang, Y.: Lvagent: Long video understanding by multi-round dynamical collaboration of mllm agents. arXiv preprint arXiv:2503.10200 (2025)
4. Chen, G., Liu, Y., Huang, Y., He, Y., Pei, B., Xu, J., Wang, Y., Lu, T., Wang, L.: Cg-bench: Clue-grounded question answering benchmark for long video understanding. arXiv preprint arXiv:2412.12075 (2024)
5. Ding, Y., Zhang, Y., Lai, X., Chu, R., Yang, Y.: Videozoomer: Reinforcement-learned temporal focusing for long video reasoning. arXiv preprint arXiv:2512.22315 (2025)
6. Du, Y., Li, S., Torralba, A., Tenenbaum, J.B., Mordatch, I.: Improving factuality and reasoning in language models through multiagent debate. In: Forty-first international conference on machine learning (2024)
7. Fu, C., Dai, Y., Luo, Y., Li, L., Ren, S., Zhang, R., Wang, Z., Zhou, C., Shen, Y., Zhang, M., et al.: Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 24108–24118 (2025)
8. Hu, S., Lu, C., Clune, J.: Automated design of agentic systems. arXiv preprint arXiv:2408.08435 (2024)
9. Liu, R., Liu, Z., Tang, J., Ma, Y., Pi, R., Zhang, J., Chen, Q.: Longvideoagent: Multi-agent reasoning with long videos. arXiv preprint arXiv:2512.20618 (2025)
10. Liu, Z., Zhang, Y., Li, P., Liu, Y., Yang, D.: Dynamic llm-agent network: An llm-agent collaboration framework with agent team optimization. arXiv preprint arXiv:2310.02170 (2023)
11. Luo, Y., Zheng, X., Li, G., Yin, S., Lin, H., Fu, C., Huang, J., Ji, J., Chao, F., Luo, J., et al.: Video-rag: Visually-aligned retrieval-augmented long video comprehension. arXiv preprint arXiv:2411.13093 (2024)

12. Ma, Z., Gou, C., Shi, H., Sun, B., Li, S., Rezaatofghi, H., Cai, J.: Drvideo: Document retrieval based long video understanding. arXiv preprint arXiv:2406.12846 (2024)
13. Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoy, S., Yang, Y., et al.: Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems* **36**, 46534–46594 (2023)
14. Ouyang, K., Liu, Y., Yao, L., Cai, Y., Zhou, H., Zhou, J., Meng, F., Sun, X.: Conan: Progressive learning to reason like a detective over multi-scale visual evidence. arXiv preprint arXiv:2510.20470 (2025)
15. Pang, Z., Wang, Y.X.: Mr. video: “mapreduce” is the principle for long video understanding. arXiv preprint arXiv:2504.16082 (2025)
16. Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models. arXiv preprint arXiv:2402.03300 (2024)
17. Shen, X., Zhang, W., Chen, J., Elhoseiny, M.: Vgent: Graph-based retrieval-reasoning-augmented generation for long video understanding. arXiv preprint arXiv:2510.14032 (2025)
18. Wang, L., Xu, W., Lan, Y., Hu, Z., Lan, Y., Lee, R.K.W., Lim, E.P.: Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In: *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*. pp. 2609–2634 (2023)
19. Wang, S., Jin, J., Wang, X., Song, L., Fu, R., Wang, H., Ge, Z., Lu, Y., Cheng, X.: Video-thinker: Sparking" thinking with videos" via reinforcement learning. arXiv preprint arXiv:2510.23473 (2025)
20. Wang, W., He, Z., Hong, W., Cheng, Y., Zhang, X., Qi, J., Ding, M., Gu, X., Huang, S., Xu, B., et al.: Lvbenc: An extreme long video understanding benchmark. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 22958–22967 (2025)
21. Wang, X., Zhang, Y., Zohar, O., Yeung-Levy, S.: Videoagent: Long-form video understanding with large language model as agent. arXiv preprint arXiv:2403.10517 (2024)
22. Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171 (2022)
23. Wang, Z., Yu, S., et al.: Videotree: Adaptive tree-based video representation for llm reasoning on long videos. arXiv preprint arXiv:2405.19209 (2024)
24. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* **35**, 24824–24837 (2022)
25. Wu, H., Li, D., Chen, B., Li, J.: Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems* **37**, 28828–28857 (2024)
26. Yang, Z., Wang, S., Zhang, K., Wu, K., Leng, S., Zhang, Y., Li, B., Qin, C., Lu, S., Li, X., et al.: Longvt: Incentivizing" thinking with long videos" via native tool calling. arXiv preprint arXiv:2511.20785 (2025)
27. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In: *The eleventh international conference on learning representations* (2022)
28. Ye, R., Tang, S., Ge, R., Du, Y., Yin, Z., Chen, S., Shao, J.: Mas-gpt: Training llms to build llm-based multi-agent systems. arXiv preprint arXiv:2503.03686 (2025)

29. Yuan, H., Liu, Z., et al.: Videoexplorer: Think with videos for agentic long-video understanding. arXiv preprint arXiv:2506.10821 (2025)
30. Zeng, X., Zhang, Z., Zhu, Y., Li, X., Wang, Z., Ma, C., Zhang, Q., Huang, Z., Ouyang, K., Jiang, T., et al.: Video-o3: Native interleaved clue seeking for long video multi-hop reasoning. arXiv preprint arXiv:2601.23224 (2026)
31. Zhang, C., Wang, Z., Ma, Y., Peng, J., Wang, Y., Zhou, Q., Song, J., Zheng, B.: Rewatch-r1: Boosting complex video reasoning in large vision-language models through agentic data synthesis. arXiv preprint arXiv:2509.23652 (2025)
32. Zhang, X., Jia, Z., Guo, Z., Li, J., Li, B., Li, H., Lu, Y.: Deep video discovery: Agentic search with tool use for long-form video understanding. arXiv preprint arXiv:2505.18079 (2025)
33. Zhang, Y., Wu, J., Li, W., Li, B., Ma, Z., Liu, Z., Li, C.: Llava-video: Video instruction tuning with synthetic data. arXiv preprint arXiv:2410.02713 (2024)
34. Zhou, J., Shu, Y., Zhao, B., Wu, B., Liang, Z., Xiao, S., Qin, M., Yang, X., Xiong, Y., Zhang, B., et al.: Mlvu: Benchmarking multi-task long video understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13691–13701 (2025)
35. Zhuge, M., Wang, W., Kirsch, L., Faccio, F., Khizbullin, D., Schmidhuber, J.: Gptswarm: Language agents as optimizable graphs. In: Forty-first International Conference on Machine Learning (ICML) (2024)
36. Zuo, J., Deng, Y., Kong, L., Yang, J., Jin, R., Zhang, Y., Sang, N., Pan, L., Liu, Z., Gao, C.: Videolucy: Deep memory backtracking for long video understanding. arXiv preprint arXiv:2510.12422 (2025)