Prysma: Efficient Modality Adaptation for SLO-aware LLM-based Video Question Answering

Botao Zhu
Global College
Shanghai Jiao Tong University
zhubotao@sjtu.edu.cn

Xiaoyi Fan
JX Intelligence Technology Co., Ltd.
xiaoyi.fan@ieee.org

Yifei Zhu*
Global College
Shanghai Jiao Tong University
yifei.zhu@sjtu.edu.cn

Abstract

Large language models (LLMs) have demonstrated remarkable performance in video question answering (VQA). To improve responsiveness and answer quality, modern LLM-based VQA services pre-extract multiple LLM-compatible modalities from videos before response generation. However, the performance impact of modality combinations and extraction knob settings has barely been studied. Our analysis shows that both factors substantially affect answer quality and Time-to-First-Token (TTFT) latency. Furthermore, the best configuration depends on both video content and query semantics, necessitating adaptive strategies. We propose Prysma, the first modality adaptation framework for LLM-based VQA. Prysma intelligently adapts the modality configurations to optimize answer quality, subject to a per-query TTFT service-level objective (SLO). It combines an offline modality orchestrator and an online semantic adapter. The offline orchestrator constructs a video knowledge base to warm-start tuning, employs a latency-aware configuration pruner to reduce search space, and runs a two-stage tuner to optimize both video- and query-level configurations. The online adapter selects configurations in real time based on query semantics and tuning history. Extensive evaluations demonstrate that Prysma improves answer quality by up to 82% over SOTA methods under the same SLO setting.

1 Introduction

LLM-based video question answering (VQA) has emerged as a new paradigm for flexible video analysis. To answer users' queries about a specific video, the LLM-based VQA system proceeds through a two-phase pipeline, consisting of an offline modality extraction phase and an online response generation phase. In practice, the modality extraction phase is performed offline by the provider in advance: videos are preprocessed using pretrained models and algorithms to transform them into LLM-compatible modalities, such as vision tokens (VT) [22], captions [32], and audio transcripts [8]. These extracted modalities are stored and can be directly retrieved when online queries arrive, avoiding repeated preprocessing. To extract these modalities, corresponding machine learning models are invoked under specific knobs such as frame rate, resolution, and modality-specific threshold. In the online response generation phase, the extracted multimodal representations are

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

combined with the user's query to form the final input, which is then fed into the LLM to produce a text-based response.

Our comprehensive experiments (§ 2.2) reveal that the choice of input modalities and the knob settings for extracting each modality have a significant impact on latency and answer quality of LLM-based VQA. Across videos, different modality combinations (MCs) exhibit up to $2.34\times$ variation in answer quality and $64.6\times$ variation in TTFT. Further, even within the same MC, the knob settings introduce substantial performance differences. For example, different knob settings of caption or VT lead to up to $1.67\times$ answer quality and $7\times$ latency variation. Ignoring the heterogeneity among modalities and knobs can lead to substantial degradation in answer quality and unnecessary delay.

Beyond our empirical findings, existing explorations of major providers on multimodal video understanding further suggest the practical importance of intelligently adapting modality configurations. For example, AWS presents multiple video-understanding workflows with different cost-performance trade-offs, and lets users select frame-similarity measures and adjust thresholds for removing redundant frames in its smart-sampling stage [2]. It also explicitly provides empirical configuration suggestions based on video content. Azure AI Video Indexer similarly analyzes videos through a heterogeneous pipeline of 30+ AI models, producing transcripts, OCR, faces, and other multimodal insights [3]. These initial explorations further emphasize the importance of content-dependent modality adaptation. However, current practices still largely focus on fixed workflows and rely on manual configuration, lacking a systematic approach to selecting modalities and extraction knobs.

The above empirical and industrial observations give rise to a new system problem in LLM-based VQA that we call *modality adaptation*: selecting the optimal configuration, which is defined as a pair of MC and modality-specific knob settings, to optimize answer quality subject to a per-query TTFT SLO. While existing studies have started to explore tuning for better LLM inference performance [7, 14], they focus on optimizing the LLM serving engine with text-only input, rather than the upstream multimodal pipeline that constructs the VQA input. To the best of our knowledge, this problem has not been systematically studied before.

Compared with configuration tuning for text-based LLM systems, modality adaptation in LLM-based VQA introduces several unique challenges. First, combinatorial and interaction-heavy configuration space. We need to jointly choose the MC and per-modality extraction knobs in a heterogeneous space of over 10^6 configurations (see Appendix A), where cross-modal interactions make performance highly non-additive and hinder transfer of "good" settings across combinations. Second, modality-dependent feasibility boundaries. Different modalities impact the TTFT in various ways (e.g., longer prompts and retrieval overhead) and exhibit widely varying TTFT sensitivities, invalidating one-size-fits-all rules for identifying and pruning SLO-violating configurations. Third, video- and query-conditioned optimality. The best modality configurations vary across videos and even across queries for the same video, making any single static configuration brittle and suboptimal. Such a unique property necessitates fine-grained adaptation for LLM-based VQA with high efficiency.

In this paper, we propose Prysma, a framework that automatically adapts modality configurations for SLO-aware, high-quality LLM-based VQA. We observe that visually similar videos tend to favor similar modality combinations, while semantically similar queries tend to favor similar configurations. *Our key insight is to perform one-shot expensive configuration exploration offline and conduct lightweight, query-specific configuration selection online by taking advantage of the intrinsic locality property identified in configuration preferences.* Specifically, Prysma incorporates an offline modality orchestrator and an online semantic adapter. First, motivated by *inter-video visual consistency* (§ 3.2.1), the offline modality orchestrator builds a video knowledge base and warm-starts tuning by retrieving promising MCs from visually similar videos. Second, leveraging *latency monotonicity* (§ 3.2.3), it adopts a latency-aware configuration pruner to dynamically remove SLO-violating regions of the configuration space, avoiding wasted trials under a strict TTFT SLO and limited budget. Aided by the pruner and knowledge base, a novel two-stage tuner is proposed to first search for a strong video-level configuration, then progressively concentrate on a smaller query subset to refine query-level optimal configurations. Finally, driven by *inter-query semantic consistency* (§ 3.3), the online semantic adapter exploits inter-query similarity and accumulated tuning experience to select an effective configuration for each arriving query in real time. Our main contributions are summarized as follows:

- We conduct a comprehensive study of LLM-based VQA, revealing the complex relationship between modality configurations, answer quality, and latency. Based on this, we formally propose a new optimization problem, namely modality adaptation.
- We identify several key insights for modality adaptation, namely *inter-video visual consistency* for offline tuning, *latency monotonicity* for configuration pruning, and *inter-query semantic consistency* for online adaptation.
- We propose Prysma, the first modality adaptation framework for efficient and high-quality LLM-based VQA. It combines offline tuning and online adaptation to improve answer quality while satisfying TTFT SLO.
- Extensive experiments on multiple video LLMs and VQA benchmarks show that Prysma consistently outperforms prior tuning baselines under the same SLO setting, improving answer quality by up to 82%. The source code is publicly available at `https://github.com/oximi123/Prysma-NeurIPS-2026`.

2 Background and Motivation

2.1 LLM-based VQA

LLM-based VQA is an emerging VQA serving paradigm that leverages LLMs to answer human queries about videos. In a typical LLM-based VQA pipeline [6], the system first extracts LLM-compatible modalities from the raw video, and then feeds the multimodal representations together with a user's query into the LLM to generate an answer. The interaction is subject to a per-query TTFT SLO [7, 14]. TTFT measures the delay from sending a query to receiving the first generated token, and the SLO specifies the maximum TTFT for each request.

Early LLM-based VQA systems [17, 25, 34] rely mainly on a single modality (e.g., vision tokens from sampled frames), assuming that visual representations alone are sufficient for downstream reasoning. However, subsequent research [8, 28] reveals that visual tokens capture only appearance and motion cues, while semantic details, such as speech content, text in scenes, and object-level relationships, require additional modalities like audio, OCR, and object detection records. Consequently, modern LLM-based VQA pipelines expand from single-modality inputs to multimodal integration, combining visual, linguistic, and acoustic information to achieve a comprehensive understanding [6, 20, 28].

2.2 Data Analysis and Motivation

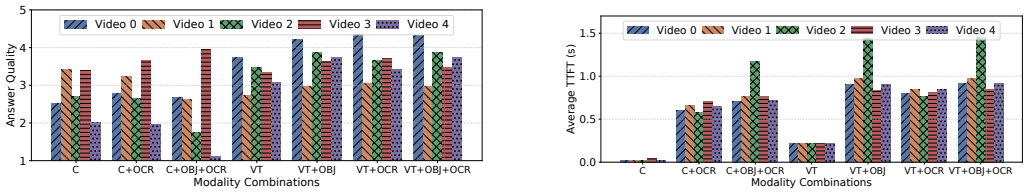
In this part, we conduct extensive measurements to reveal the necessity for modality adaptation.

2.2.1 Setup

We use MSVD-QA [29], a well-established open-ended VQA dataset, and select a subset of 500 videos with the largest number of associated queries (totaling nearly 18,000 queries) for motivation. We extract vision token (VT), caption (C), object detection record (OBJ), and optical character recognition (OCR) using the default knob settings provided in the corresponding literature [20, 32, 34], and follow the prompt template in [20] to merge these modalities as input. We follow the practice in [20, 28], which uses the video LLM to extract modality-related keywords from each query in order to retrieve relevant information for OBJ and OCR. We use LLaVA-Video-7B [34] as our video LLM. We report TTFT and answer quality, where the latter is evaluated using LLM-based scoring [26] on a 0–5 scale (higher is better). Following the community-recognized evaluation protocol [8, 21, 30], we use GPT-3.5-turbo as the judge.

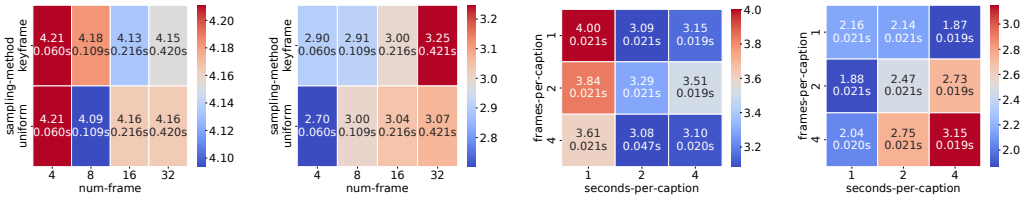
2.2.2 Necessity for Modality Adaptation

We first investigate the influence of different MCs on various videos. For illustration, we present the results for five videos across seven different MCs in Fig. 1. The five videos cover diverse content, including chatting, vehicles, animals, cooking, and sports. We observe that different MCs perform inconsistently across videos, and each MC exhibits varying degrees of influence on the average TTFT and answer quality. The gap between the best and worst MCs reaches up to 2.34× in average answer quality, and 64.6× in average TTFT driven by the surge in prompt tokens and the added overhead of



(a) Average Answer Quality (b) Average TTFT

Figure 1: Average answer quality (a) and TTFT (b) of five videos with different MCs.



(a) Video A (VT). (b) Video B (VT). (c) Video A (Caption). (d) Video B (Caption).

Figure 2: Answer quality (top in each entry) and TTFT (bottom in each entry) of two videos with different knob settings of vision token (VT) and caption.

LLM-based retrieval. Notably, no single MC remains optimal for all videos in terms of both TTFT and answer quality, indicating that different videos require distinct optimal MCs.

Next, we investigate the knob settings within individual modalities. We focus on two commonly used modalities, namely caption and VT. In Fig. 2c and 2d, two videos exhibit completely opposite preferences for the caption knob setting. In Fig. 2a and 2b, increasing *num-frame* (i.e., number of frames encoded as VT) improves the performance of Video B, but has a limited or even negative effect on Video A. The worst knob setting can have up to 30% and 67% answer quality gap compared to the best knob setting for VT and caption, respectively. These results demonstrate that, in addition to the MC, the optimal knob setting of each modality also varies across videos, and significantly influences the answer quality and the TTFT. We provide a holistic view of such heterogeneity across all evaluated videos in Appendix B.

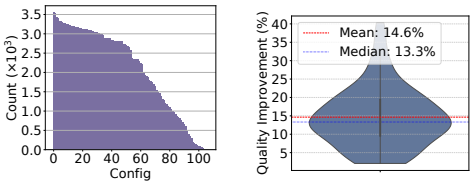
We further analyze the performance-configuration relationship at a finer-grained query-level. We evaluate 400 randomly selected configurations per video, constrained by a 2.0s TTFT SLO. We define query-level optimality as selecting the configuration that yields the highest answer quality (allowing ties) for each query. As shown in Fig. 3a, around 110 configurations are selected as query-level optimal by at least one query, ranked by the number of queries for which they are optimal. Notably, a query may have multiple configurations that achieve optimality. Even the most frequently selected configuration covers less than one-fifth of all queries, meaning no configuration consistently outperforms the others. We compare video-level and query-level optimality. Fig. 3b shows that the latter improves the average answer quality by 14.6%, with the maximum improvement reaching nearly 40%. It reveals a substantial gap between video-level and query-level optima and highlights the strong potential of adapting configurations at the query level.

The above findings show that the modality configurations exert strong influences on both answer quality and latency, making adaptive and fine-grained modality tuning not just beneficial but essential for reliable VQA performance.

3 Prysma Framework

3.1 Framework Overview

The architecture of Prysma is shown in Fig. 4. It mainly consists of an offline modality orchestrator and an online semantic adapter. The service provider specifies a TTFT SLO and a tuning budget measured in full-query-set trials. When a video with a set of queries is uploaded, the offline modality



(a) Count of configs when achieving query-level optimality.

(b) Improvement compared with video-level optimality.

Figure 3: Query-level optimal configurations.

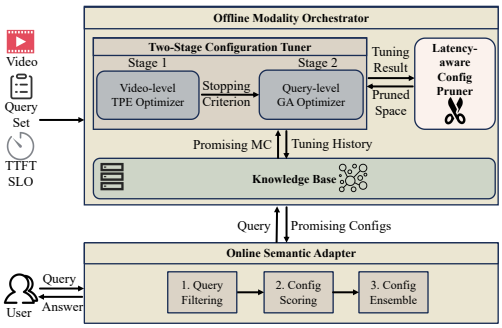


Figure 4: Prysma framework.

orchestrator efficiently tunes the modality configurations for it. The orchestrator contains three parts: a video knowledge base, a latency-aware configuration pruner, and a two-stage configuration tuner. After the offline tuning, the tuning history is stored in the knowledge base, and the video is ready for online question answering. Each query will be processed by the online semantic adapter to select the best configuration through query filtering, configuration scoring, and configuration ensemble.

3.2 Offline Modality Orchestrator

In this part, we introduce our offline modality orchestrator, which contains a video knowledge base, a two-stage configuration tuner, and a latency-aware configuration pruner. The two-stage tuner leverages the knowledge base to warm-start tuning, and the pruner to prune SLO-violating candidates.

3.2.1 Video Knowledge Base

Our earlier analysis reveals large variations in optimal MCs across videos, implying that video content strongly influences modality effectiveness. While the diversity of video content challenges the identification of optimal MCs, we reveal that the MC preferences across videos exhibit high *inter-video visual consistency*, i.e., videos with similar visual features are more likely to share the same top-performing MCs (see Appendix C.1).

Based on this insight, we first select and tune a small set of representative videos, and subsequently construct a knowledge base, which is then used to optimize and accelerate the modality adaptation process for the remaining videos. We use a clustering algorithm (i.e., K-means) to group the videos into multiple clusters using their features extracted by CLIP [22]. We select the videos closest to each cluster center, with the number of selected videos from each cluster proportional to the cluster's size. We store the video features and the tuning processes, including each evaluated configuration and its performance, and leverage them to warm-start and accelerate the tuning of the remaining videos.

3.2.2 Two-stage Configuration Tuner

With the representative videos and their tuning history in the knowledge base, we present our two-stage configuration tuner, which efficiently optimizes the best configurations for both the entire video and individual queries. Using the knowledge base, we first estimate the value of each MC that quantifies the expected answer quality it will deliver on the current video. Specifically, we extract the CLIP features of the current video and compute its similarity with the video features stored in the knowledge base. The value of each MC for the target video v is then calculated as follows:

$$val_{mc}^{v} = \frac{\sum_{v' \in V_{mc}} -\ln(1 - sim(v, v') + \epsilon) \cdot s_{mc}^{v'}}{|V_{mc}|}, \tag{1}$$

where V_{mc} denotes the set of videos that have been evaluated on mc, $s_{mc}^{v'}$ denotes the score of the current MC on video v', obtained by maximizing the average query answer quality of all evaluated configurations associated with that MC. Instead of directly multiplying the similarity term (clipped to $[0, 1]$), we employ the function $-ln(1 - x)$ to amplify the impact of higher similarity values, giving greater weight to videos with higher visual similarity. We also add a small constant ϵ (e.g., 10^{-6}) to ensure numerical stability.

Stage 1: coarse-grained video-level tuning. The knowledge base provides us with a set of promising MCs for each video. Since different MCs may exhibit different value val_{mc}^{v} and budget sensitivity, we design an adaptive budget allocation strategy to prioritize MCs and allocate trials effectively under a limited tuning budget. We first allocate the total budget to each MC proportionally to val_{mc}^{v} and rank MCs by their allocated budget for tuning accordingly. We then tune each selected MC as follows: we employ the Tree-structured Parzen Estimator (TPE) [5], which is well-suited for tree-structured configuration spaces in modality adaptation, to search for the optimal configuration. TPE models two densities:

$$l(x) = p(x \mid y < y^*), \quad g(x) = p(x \mid y \geq y^*), \tag{2}$$

where x denotes a candidate configuration, y is its objective value, and y^* is a threshold. The expected improvement (EI) of candidate x is expressed as:

$$\mathrm{EI}(x) \propto \frac{l(x)}{g(x)}. \tag{3}$$

After each trial, we update the global best average performance on the tuning set and estimate the potential of each remaining MC by randomly sampling several configurations within that MC and averaging their EIs. If any remaining MCs show higher estimated EI than the current one, we recycle the remaining budget of the current MC and reassign it to that more promising MC; conversely, when the current MC's budget is exhausted but it has achieved improvements since its latest budget update, we pull extra budget from the lowest-value MC to extend the current tuning. The pseudocode and details of our dynamic budget allocation strategy are provided in the Appendix F.1.

The process aims for a video-level optimal configuration to maximize average answer quality across all queries. However, query diversity can make a single configuration suboptimal for all queries, potentially trapping the tuner in a limited subspace and missing configurations that could benefit specific query subsets. Additionally, poorly answered queries in some videos can cause video-level optimization gains to plateau or disappear after a few trials, wasting budget (see Appendix E).

Stage 2: fine-grained query-level tuning. To address this issue, we perform query-level tuning only on poorly answered queries. However, running a separate tuner for each query is too expensive, as it frequently triggers modality extraction and scales poorly with the number of queries. Instead, we tune an adaptively selected subset of low-quality queries and update the subset as tuning progresses, capturing query-level heterogeneity while remaining scalable. This strategy brings two benefits: First, it reduces the cost per trial. Since LLM inference dominates the evaluation cost, while evaluating one configuration on the full query set Q_v costs 1 unit budget, evaluating it on a subset Q'_v costs $|Q'_v|/|Q_v|$ units theoretically, allowing us to explore more configurations. Second, it alleviates the dominance of well-answered queries, enabling the tuner to enhance the performance of poorly answered queries by exploring alternative and potentially more suitable configuration regions.

Our solution is further designed as follows. We use the early-stopping method proposed in [15] that evaluates the expected reduction in simple regret and terminates the tuning once the improvement becomes negligible. When the stopping condition is satisfied, we switch to Stage 2, focusing on poorly answered queries and searching for query-level optimal configurations. For each trial, we construct a subset consisting of queries whose best-so-far score falls below the current average across all queries, while excluding queries that have already achieved an empirically high score (e.g., 4.5). We employ the Genetic Algorithm (GA), because it more effectively escapes local optima in hybrid search spaces via crossover and mutation operations [23], while its higher sample demand [12] is mitigated by the prior exploration of TPE and the reduced query set. We repeat the above query-level tuning until the tuning budget runs out. The pseudocode of our two-stage tuner, together with the tuning process of videos in the knowledge base are provided in Appendix F.

3.2.3 Latency-aware Configuration Pruner

Considering the vast search space, we design a latency-aware configuration pruner to avoid SLO-violating trials. Our key insight is that LLM-based VQA exhibits a strong form of *latency monotonicity*. First, the number of input tokens increases monotonically with the inclusion of additional modalities. Incorporating a new modality into an existing MC therefore will not reduce the TTFT. Second, many knobs in LLM-based VQA monotonically affect the number of input tokens and consequently latency, independent of other knob settings. Examples include the *num-frame* parameter in VT, lower *detect-threshold* values, and higher *fps* values in OBJ, all of which increase the number of input tokens.

Building on this insight, we propose a simple yet effective training-free configuration pruner that reduces the search space by eliminating SLO-violating configurations. We first profile the relationship between input token count and TTFT by running samples across a range of prompt lengths. Before tuning, we start with the VT modality, which typically dominates input length due to the large number of tokens generated per frame. Its contribution to the token count is consistent across videos and straightforward to estimate. This enables us to determine the maximum feasible configuration with VT and prune all configurations with knob settings that exceed this bound. We additionally profile other runtime components (e.g., retrieval overhead) in modern LLM-based VQA pipelines [20] to further narrow the search space. Finally, we identify latency-monotonic knobs by measuring token-count changes under controlled knob perturbations (with other parameters fixed).

During tuning, we prune configurations based on the token counts of previously extracted modalities, leveraging the monotonicity property to enhance pruning efficiency. Simultaneously, once a configuration violates the SLO, we safely discard all configurations with more modalities and larger settings for these latency-monotonic knobs, since they are guaranteed to incur higher TTFT. The details and the pseudocode of our pruner are provided in Appendix G.

3.3 Online Semantic Adapter

Despite the fine-grained query-level tuning, real-world VQA systems inevitably encounter new and unseen queries at inference time. It motivates the need for an online adapter that can dynamically infer configuration preferences for each incoming query. We reveal that the configuration preferences across queries exhibit high *inter-query semantic consistency*, i.e., semantically similar queries are likely to share the same top-performing configurations (see Appendix C.2).

Based on this insight, we propose an online semantic adapter to identify the most promising configuration for a given query q. We first compute the semantic similarity between q and all tuned queries of the corresponding video, and collect those whose similarity exceeds a predefined threshold s. The threshold is determined offline via a grid search on the knowledge base to maximize the average answer quality. This filtering step mitigates noise from semantically irrelevant queries, stabilizing subsequent configuration selection.

With the filtered queries, we next design a weighted-voting ensemble method to obtain the best configuration for answering the current query. Specifically, for each configuration c in the tuning history, we gather all filtered queries Q_c that have been evaluated on it in the tuning history, and compute its value val_c^q for question q as follows:

$$val_c^q = \frac{\sum_{q' \in Q_c} -ln(1 - sim(q, q') + \epsilon) \cdot s_c^{q'}}{|Q_c|}. \tag{4}$$

Here, $s_c^{q'}$ denotes the evaluation score of query q' under configuration c. This value considers both the semantic similarity (clipped to $[0, 1]$) between two queries and the answer quality using the configuration. We use the same function $-ln(1 - x)$ to amplify the impact of queries with higher similarity. We then compare the values of all such configurations and select the one with the highest val_c^q as the optimal configuration for answering q. If multiple configurations achieve the same highest value, we choose the one with the lowest measured TTFT. If no query exceeds the threshold, we apply the video-level optimal configuration obtained during the offline tuning. We further discuss deployment considerations (e.g., storage and cold-start) in Appendix D.

4 Evaluation

4.1 Experiment Setting

Implementation: We implement Prysma in $\sim$3K lines of Python with PyTorch and Optuna [1]. We leverage the prompt provided in [20] to integrate different modalities. We conduct the experiments on a server equipped with two NVIDIA H800 GPUs, an AMD EPYC 9654 Processor, and 120 GB host RAM. The number of clusters K is selected by maximizing the Silhouette Score. We average CLIP features from 16 uniformly sampled frames as a video representation. The tuning budget is 30 full-set trials. The stopping criterion is realized by Optuna's EMMREvaluator.

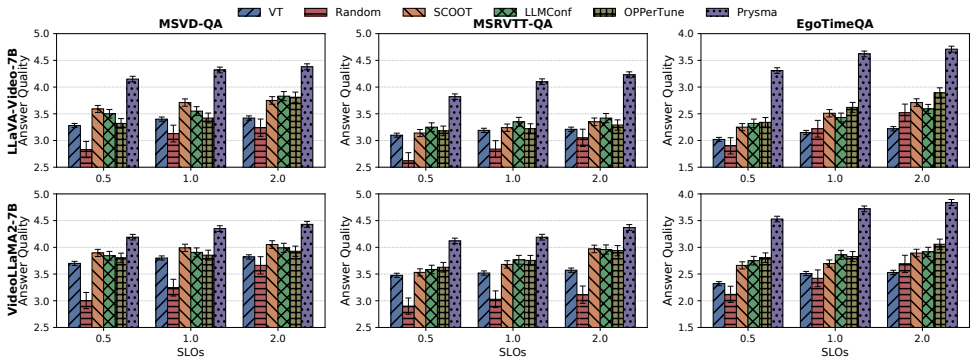


Figure 5: Answer quality over two video LLMs and three VQA datasets under three SLOs.

Models: We choose two widely used video LLMs, namely LLaVA-Video-7B [34] and VideoLLaMA2-7B [8]. We use OpenAI CLIP ViT-Large [22] and all-MiniLM-L6-v2 [27] to extract the features, and `gpt-3.5-turbo-0125` as our judge.

Datasets: We use two short-video datasets, MSVD-QA and MSRVTT-QA [29], and one long-video dataset, EgoTimeQA [11]. In addition, we include a multi-choice Video-MME benchmark [13] to evaluate Prysma's transferability across datasets.

Evaluation Setting: We consider three TTFT SLOs of 0.5s, 1.0s and 2.0s, which align with interactive latency targets commonly adopted in long-context LLM serving [19]. The answer quality of any SLO-violating query is treated as 0. The tuning budget is set to 30 full-set trials, consistent with [7]. Other detailed settings are provided in Appendix H.

Evaluation protocol: We tune on 40% of queries per video for MSVD-QA and MSRVTT-QA and 30% for EgoTimeQA. The main comparison evaluates the full query set, including tuning and held-out queries; Appendix I.5 separately reports results on queries not used for tuning.

Baselines: We compare Prysma with five baselines: (1) VT, which uses only uniformly sampled vision tokens and selects the best SLO-satisfying frame count; (2) Random, which randomly samples configurations; (3) SCOOT [7], a GP-based BO tuner with random-forest hidden-constraint modeling; (4) LLMConf [14], a genetic-algorithm tuner using Chebyshev distance; and (5) OPPerTune [24], a reinforcement learning (RL)-based cloud-service tuner.

4.2 Overall Performance Comparison

We report the average answer quality achieved under the given SLO. The error bars denote standard deviation over five random seeds. As shown in Fig. 5, Prysma consistently outperforms all competing baselines across all datasets and video LLMs. Specifically, Prysma improves answer quality by 13%–82%, 20%–66%, 10%–47%, 11%–48%, and 10%–41% compared with VT, Random, SCOOT, LLMConf, and OPPerTune, respectively. Overall, the Random sampler performs the worst in most cases, particularly under stricter SLOs, as many of its blindly searched configurations violate the SLO. The VT baseline benefits from looser TTFT SLOs because it can sample more frames and capture richer visual cues, but the gain is modest on short-video datasets like MSVD-QA and MSRVTT-QA.

While the other three tuners consistently outperform fixed VT and the naive Random baseline, none fundamentally solves modality adaptation for LLM-based VQA. SCOOT uses a random-forest pruner to filter SLO-violating configurations, but its GP-based BO struggles in large hybrid spaces with many categorical/discrete knobs. OPPerTune trains an RL agent online and treats continuous and discrete knobs separately, yet it typically needs many trials and converges slowly when rewards are sparse under strict SLOs. LLMConf completely relies on GA operations for tuning, which are sample-hungry. Under tight budgets, it fails to deliver efficient adaptation.

Prysma's advantage is most pronounced on long-video, high-difficulty datasets such as EgoTimeQA. The rich, extended content makes combining multiple modalities crucial for improving LLM-based VQA, consistent with recent studies [20, 28]. Our results also expose the weakness of single-modality reliance (e.g., VT), which underlies many prior approaches [8, 18]. For example, on EgoTimeQA

with LLaVA-Video-7B, even a random sampler can outperform VT-only configurations under loose SLOs. Overall, these results underscore the importance of multimodal integration and adaptive tuning for boosting answer quality.

4.3 System Analysis

Ablation study We conduct ablation studies in Appendix I.1 to isolate the impact of four main components in Prysma: video knowledge base, latency-aware configuration pruner, two-stage tuning algorithm and online semantic adapter. The results confirm that each component contributes positively to the performance of Prysma.

Sensitivity analysis We study the sensitivity of Prysma to the size of the video knowledge base and the modality pool in Appendix I.2. The results show that (1) a relatively small proportion of the videos is enough to balance diversity and efficiency for video knowledge base construction, and (2) Prysma exhibits great robustness and consistently outperforms other baselines under modality pools with different modalities.

Overhead analysis We report the runtime and tuning overhead of Prysma in Appendix I.3. The results demonstrate that its online adaptation incurs only millisecond-level latency. In addition, the offline tuning cost is worthwhile compared to the substantial improvement, and such cost can be effectively amortized in production environments.

Human preference calibration We present a human-preference calibration study in Appendix I.4. The results show that human judgments largely agree with LLM-based evaluation and consistently prefer answers optimized by Prysma compared to other baselines.

Cold-start and transferability We provide cold-start performance on the three main datasets (MSVD-QA, MSRVTT-QA, and EgoTimeQA) and cross-dataset transferability on Video-MME dataset in Appendix I.10. The results show that configurations discovered on tuned videos can be effectively reused for cold-starting unseen videos and transferred to videos that belong to different datasets.

Held-out evaluation We present held-out evaluation in Appendix I.5. The results show that Prysma outperforms the strongest baseline by 9–44% across all 18 combinations of three datasets, two video LLMs, and three TTFT SLOs. With LLaVA-Video-7B at 1.0 s, the gains are 18%, 26%, and 41% on MSVD-QA, MSRVTT-QA, and EgoTimeQA, respectively, demonstrating that configuration preferences transfer to unseen queries. Relative to mixed-query evaluation, held-out quality decreases by about 2.5% for Prysma and 3.5–5% for the tuned baselines, leaving Prysma's relative gains comparable or higher.

Judge robustness We evaluate judge robustness in Appendix I.6 by re-scoring fixed held-out answers with GPT-4o and Vicuna-13B-v1.5. The results show that Prysma retains gains of 28% and 25%, respectively, compared with 27% under the tuning judge, averaged across the three datasets with LLaVA-Video-7B at a 1.0 s SLO. Per-question scores correlate strongly with the tuning judge ($\rho = 0.83$), and five repeated evaluations of identical answers show a mean absolute score difference of 0.04.

Larger video LLMs We evaluate larger models in Appendix I.7. The results show that Prysma improves answer quality by 14–31% with PLLaVA-13B and 15–34% with Qwen3-VL-32B-Instruct at a 2.0 s TTFT SLO. Both models are served in BF16, and the largest gains occur on EgoTimeQA.

Raw quality and SLO compliance We report raw answer quality and SLO compliance in Appendix I.8 for MSVD-QA with LLaVA-Video-7B. The results show 12–18% raw-quality gains across 0.05–10 s SLOs and violation rates below 1% at the main 0.5–2.0 s SLOs. At 1.0 s, Prysma improves raw quality from 3.69 to 4.24 and reduces the violation rate from 3.60% to 0.53%. At the tightest 0.05 s SLO, Prysma reduces violations from 37.49% to 12.18%. Quality saturates beyond 3.0 s, reaching 4.32 at 10 s with zero observed violations.

Batched serving We study batched serving in Appendix I.9 with LLaVA-Video-7B on EgoTimeQA. The results show that Prysma retains a 43% quality gain at batch size four and a 2.0 s SLO after recalibrating the token-to-TTFT profile. Average TTFT is 1.82 s for Prysma and 1.93 s for the strongest baseline, showing that modality adaptation remains effective under batching.

5 Related Work

Configuration Tuning Modern applications, such as machine learning and databases, typically contain multiple tunable knobs that affect runtime performance. Extensive efforts have been devoted to developing efficient configuration tuning algorithms [16, 33]. QTune [16] encodes the query vector and proposes a DS-DDPG model to tune the knobs of database systems. OpAdviser [33] leverages transfer learning for database tuning. With the proliferation of LLMs, recent efforts such as SCOOT [7] and LLMConf [14] focus on optimizing the parameters of LLM inference engines. However, these works primarily focus on single-modality inference without considering multimodal interactions. In contrast, Prysma adapts modality configuration across videos and queries, achieving substantially higher answer quality.

LLM-based VQA LLM-based VQA has been widely explored [8, 18, 20, 28]. VideoLLaVA [18] binds visual signals to the language feature space to unify visual representations. VideoLLaMA2 [8] incorporates a tailor-made spatial-temporal convolution connector and an audio branch to enrich the multimodal understanding capabilities. VideoAgent [28] and VideoRAG [20] further enhance video LLMs by incorporating additional modalities to improve answer quality. While these works improve model architectures and training paradigms, they rely on static modality extraction pipelines. Prysma shows that modality preferences and configuration sensitivities vary widely, making static pipelines suboptimal. It offers a new path to enhance the performance of multimodal systems.

6 Conclusion

We propose Prysma, the first SLO-aware modality adaptation framework for LLM-based VQA. It combines an offline modality orchestrator with an online semantic adapter. Offline, Prysma builds a knowledge base to accelerate tuning, applies a latency-aware configuration pruner to filter SLO-violating settings, and uses a two-stage tuner to identify both video- and query-level optimal configurations. Online, it leverages semantic similarity to pick a promising configuration for each incoming query. Extensive experiments show that Prysma consistently boosts answer quality while meeting SLOs, substantially outperforming SOTA methods.

Acknowledgments and Disclosure of Funding

This work is supported by the National Key R&D Program of China (Grant No. 2024YFC3017100) and the National Natural Science Foundation of China (Grant No.62672299). We also thank the compute resource support from JX Intelligence Technology Co., Ltd.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proc. ACM SIGKDD Conf. on Knowl. Discovery Data Mining (KDD)*, pages 2623–2631, Aug. 2019.

[2] AWS. Unlocking video insights at scale with amazon bedrock multimodal models, 2026. URL `https://aws.amazon.com/cn/blogs/machine-learning/unlocking-video-insights-at-scale-with-amazon-bedrock-multimodal-models/`.

[3] Azure. Azure ai video indexer insights, 2026. URL `https://learn.microsoft.com/en-us/azure/azure-video-indexer/insights-overview`.

[4] Fu Bang. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proc. 3rd Workshop for Natural Lang. Process. Open Source Softw. (NLP-OSS 2023)*, pages 212–218, December 2023.

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 2546–2554, Dec. 2011.

[6] Gohar Irfan Chaudhry, Esha Choukse, Haoran Qiu, Íñigo Goiri, Rodrigo Fonseca, Adam Belay, and Ricardo Bianchini. Murakkab: Resource-efficient agentic workflow orchestration in cloud platforms. *arXiv preprint arXiv:2508.18298*, 2025.

[7] Ke Cheng, Zhi Wang, Wen Hu, Tiannuo Yang, Jianguo Li, and Sheng Zhang. SCOOT: slo-oriented performance tuning for LLM inference engines. In *Proc. ACM Web Conf. (WWW)*, pages 829–839, Apr. 2025.

[8] Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, and Lidong Bing. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *CoRR*, abs/2406.07476, 2024.

[9] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 35:16344–16359, Dec. 2022.

[10] databricks. Building a cost-optimized chatbot with semantic caching, 2024. URL `https://www.databricks.com/blog/building-cost-optimized-chatbot-semantic-caching`.

[11] Shangzhe Di and Weidi Xie. Grounded question-answering in long egocentric videos. In *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit. (CVPR)*, pages 12934–12943, Jun. 2024.

[12] Stephanie Forrest. Genetic algorithms. *ACM Comput. Surv. (CSUR)*, 28(1):77–80, Mar. 1996.

[13] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Caifeng Shan, Ran He, and Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit. (CVPR)*, pages 24108–24118, 2025.

[14] Jingkai He, Pengfei Chen, Yilun Wang, Haiyu Huang, Chuanfu Zhang, Haojia Huang, and Danwen Chen. LLMConf: Knowledge-enhanced configuration optimization for large language model inference. In *Proc. Int. Symp. on Qual. of Serv. (IWQoS)*, pages 1–10, Jul. 2025.

[15] Hideaki Ishibashi, Masayuki Karasuyama, Ichiro Takeuchi, and Hideitsu Hino. A stopping criterion for bayesian optimization by the gap of expected minimum simple regrets. In *Proc. Int. Conf. on Artif. Intell. Stat. (AISTATS)*, volume 206, pages 6463–6497, Apr. 2023.

[16] Guoliang Li, Xuanhe Zhou, Shifu Li, and Bo Gao. Qtune: A query-aware database tuning system with deep reinforcement learning. *Proc. VLDB Endow. (VLDB)*, 12(12):2118–2130, 2019.

[17] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023.

[18] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proc. Conf. on Empirical Methods in Natural Lang. Process. (EMNLP)*, pages 5971–5984, November 2024.

[19] Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: KV cache compression and streaming for fast large language model serving. In *Proc. ACM Spec. Int. Group on Data Commun. (SIGCOMM)*, pages 38–56, Aug. 2024.

[20] Yongdong Luo, Xiawu Zheng, Xiao Yang, Guilin Li, Haojia Lin, Jinfa Huang, Jiayi Ji, Fei Chao, Jiebo Luo, and Rongrong Ji. Video-rag: Visually-aligned retrieval-augmented long video comprehension. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

[21] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proc. Ann. Meet. Assoc. Comput. Linguist. (ACL)*, pages 12585–12602, 2024.

[22] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Int. Conf. on Mach. Learn. (ICML)*, pages 8748–8763, Jul. 2021.

[23] Mohaimenul Azam Khan Raiaan, Sadman Sakib, Nur Mohammad Fahad, Abdullah Al Mamun, Md Anisur Rahman, Swakkhar Shatabda, and Md Saddam Hossain Mukta. A systematic review of hyperparameter optimization techniques in convolutional neural networks. *Decis. Anal. J. (DAJ)*, 11:100470, Jun. 2024.

[24] Gagan Somashekar, Karan Tandon, Anush Kini, Chieh-Chun Chang, Petr Husak, Ranjita Bhagwan, Mayukh Das, Anshul Gandhi, and Nagarajan Natarajan. Oppertune: Post-deployment configuration tuning of services made easy. In *USENIX Symp. Networked Syst. Des. Implementation (NSDI)*, 2024.

[25] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. Moviechat: From dense token to sparse memory for long video understanding. In *Proc. IEEE/CVF Conf. on Comput. Vis. Pattern Recognit. (CVPR)*, pages 18221–18232, Jun. 2024.

[26] Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, Ali Vosoughi, Chao Huang, Zeliang Zhang, Pinxin Liu, Mingqian Feng, Feng Zheng, Jianguo Zhang, Ping Luo, Jiebo Luo, and Chenliang Xu. Video understanding with large language models: A survey. *IEEE Trans. on Circuits Syst. Video Technol. (TCSVT)*, pages 1–1, May 2025.

[27] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, pages 5776–5788, Dec. 2020.

[28] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *Eur. Conf. on Comput. Vis. (ECCV)*, pages 58–76, Oct. 2024.

[29] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proc. ACM Multimedia (MM)*, pages 1645–1653, Oct. 2017.

[30] Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See-Kiong Ng, and Jiashi Feng. Pllava : Parameter-free llava extension from images to videos for video dense captioning. *CoRR*, abs/2404.16994, 2024.

[31] Yifan Yu, Yu Gan, Nikhil Sarda, Lillian Tsai, Jiaming Shen, Yanqi Zhou, Arvind Krishnamurthy, Fan Lai, Hank Levy, and David E. Culler. Ic-cache: Efficient large language model serving via in-context caching. In *Proc. Symp. Operating Syst. Princ. (SOSP)*, pages 375–398, Oct. 2025.

[32] Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. A simple LLM framework for long-range video question-answering. In *Proc. Conf. on Empirical Methods in Natural Lang. Process. (EMNLP)*, pages 21715–21737, 2024.

[33] Xinyi Zhang, Hong Wu, Yang Li, Zhengju Tang, Jian Tan, Feifei Li, and Bin Cui. An efficient transfer learning based configuration adviser for database tuning. *Proc. VLDB Endow. (VLDB)*, 17(3):539–552, 2023.

[34] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Llava-video: Video instruction tuning with synthetic data. *Trans. Mach. Learn. Res. (TMLR)*, 2025, 2025.

Table 1: Common Modalities and Their Typical Knobs for LLM-based VQA

Modality	Brief Description	Typical Knobs		
		Name	**Type**	**Range / Values**
Vision Token	Patch embeddings produced by a vision encoder (e.g., ViT) that preserve local spatiotemporal cues.	sampling-method	categorical	{uniform, keyframe, ...}
		num-frame	discrete	$[4, 128]$
Caption	Human-written or automatically generated sentences summarizing the main events, actors and scenes. They offer a concise semantic overview of the video.	caption-model	categorical	{LLaVA, LaViLa, ...}
		frames-per-caption	discrete	$[1, 30]$
		seconds-per-caption	discrete	$[1, 30]$
Audio	Audio track transcribed into text using Automatic Speech Recognition (ASR), providing speech content, dialogue flow and sentiment.	asr-model	categorical	{whisper-small, ...}
		chunk-length	discrete	$[1, 30]$
		sampling-rate	discrete	$[8\ kHz, 16\ kHz]$
OCR	Optical Character Recognition: text strings extracted from frames. These textual cues enrich scene context and help precisely locate key information.	ocr-model	categorical	{EasyOCR, ...}
		num-frame	discrete	$[4, 128]$
		text-thresh	continuous	$[0.1, 0.9]$
OBJ	Object Detection Record: bounding boxes plus class labels for salient objects; often combined with tracking and action recognition to capture temporal interactions.	detect-thresh	continuous	$[0.1, 0.9]$
		detector	categorical	{RTDETR, APE, ...}
		fps	discrete	$[1, 30]$

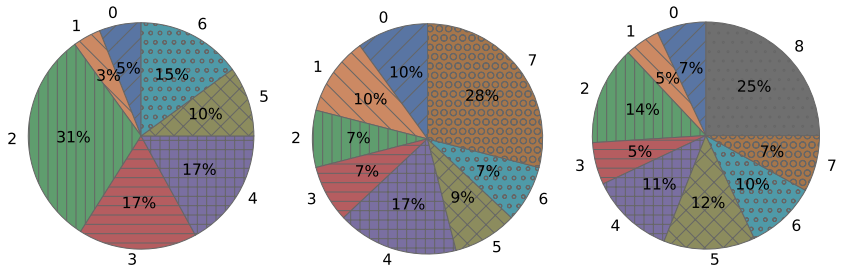
A Common Modalities and Their Typical Knobs for LLM-based VQA

We summarize several commonly used modalities in LLM-based VQA, along with their brief descriptions and representative knobs, in Table 1. These modalities convert raw videos into LLM-consumable inputs, ranging from low-level visual representations (e.g., vision tokens) to higher-level semantic signals such as captions, ASR transcripts, OCR text, and object detection records. Importantly, each modality introduces its own set of knobs that control what information is extracted and how it is encoded, directly affecting both answer quality and end-to-end TTFT. For example, vision tokens depend on the frame sampling strategy and the number of sampled frames; captioning is shaped by the caption model and its temporal granularity (frames/seconds per caption); audio transcription is governed by the ASR model, chunk length, and sampling rate; OCR quality is influenced by the OCR model, sampled frames, and text-confidence thresholds; and object detection outcomes depend on the detector choice, detection thresholds, and extraction frame rate.

The joint choice of modalities and their knob settings induces a large, heterogeneous configuration space with mixed categorical, discrete, and continuous dimensions. Even after discretizing continuous knobs into a small number of candidate values, the resulting search space remains on the order of 10^6 configurations, obtained by multiplying the numbers of candidate values for the knobs in Table 1. This combinatorial explosion poses fundamental challenges to existing approaches: exhaustive evaluation is infeasible, naive search wastes budget on low-value regions, and the strong cross-modal interactions make it difficult to identify high-quality configurations reliably under practical latency and budget constraints.

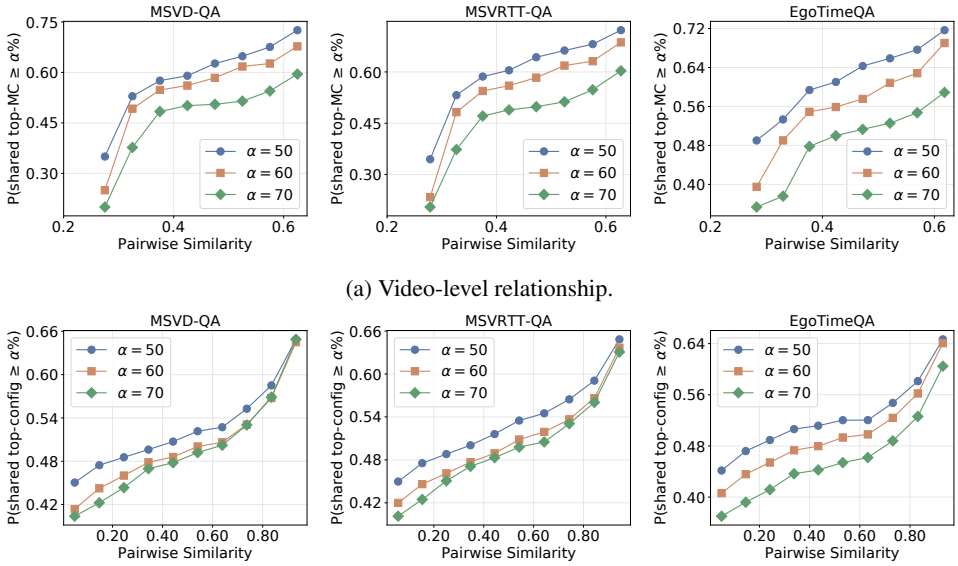
B Diversity of Configuration Preferences across all Evaluated Videos

We provide a holistic view of the diversity of optimal modality choices and knob settings over the whole 500 evaluated videos. In Fig. 6, each pie slice denotes the fraction of videos for which a particular modality combination (MC) or knob setting achieves the best answer quality. The outer labels correspond to the IDs of the associated MC/setting. The distributions are highly dispersed: optimal choices are spread across many candidates rather than concentrated on a single dominant option. Even the most frequently selected MC/setting accounts for fewer than one-third of the videos, indicating that "one-size-fits-all" configurations are inherently suboptimal. This pronounced video-level heterogeneity highlights the limitation of static modality pipelines and motivates adaptive configuration selection that can tailor modality extraction to individual videos.



(a) MCs (b) VT (c) Caption

Figure 6: Frequency distribution of video-level optimal (a) MCs, (b) VT knob settings and (c) caption knob settings across all videos.



(a) Video-level relationship.

(b) Query-level relationship.

Figure 7: Probability of sharing over $\alpha\%$ top MCs/configurations vs. similarity among different (a) videos, (b) queries on three datasets.

C Details of Insights

C.1 Insights for Offline Tuning

The MC preferences across videos exhibit high inter-video visual consistency, i.e., videos with similar visual features are more likely to share the same top-performing MCs. Our earlier analysis reveals large variations in optimal MCs across videos, implying that video content strongly influences modality effectiveness. While the diversity of video content challenges the identification of optimal MCs, we investigate whether the optimal configurations are inherently related to the video content. We define configurations whose scores fall within 0.5 points of the best-performing configuration as top configurations. An MC is then considered a top MC if any of its configurations belong to this top configuration set. We uniformly sample 16 frames from each video and extract frame-level features using CLIP [22]. These features are averaged across frames to obtain the video-level representation. For each pair of videos, we compute the cosine similarity between their representations and divide the similarity range into equal-width bins. Since a video may have multiple top MCs, for each bin, we compute the probability that a pair of videos shares more than $\alpha\%$ of their top MCs. We choose three $\alpha \in \{50, 60, 70\}$. The results in Fig. 7a indicate that similar videos are more likely to share a substantial portion of their top MCs. Given that the provider typically maintains a large pool of

videos, the search space can be significantly reduced by leveraging promising MCs from similar, well-optimized videos.

C.2 Insights for Online Adaptation

The configuration preferences across queries exhibit high inter-query semantic consistency, i.e., semantically similar queries are more likely to share the same top-performing configurations. Despite the fine-grained query-level tuning, real-world VQA systems inevitably encounter new and unseen queries at inference time. It motivates the need for dynamically inferring configuration preferences for each incoming query. Our earlier finding shows that visually similar videos have more shared top MCs. It naturally raises a related question: do semantically similar queries also exhibit consistent configuration preferences? We begin by encoding each query using the widely adopted embedding model all-MiniLM-L6-v2 [27], and compute the cosine similarity between all query pairs. Similar to Fig. 7a, Fig. 7b shows the relationship between query similarity and the probability of sharing at least $\alpha\%$ of the same top configurations. The results reveal a clear trend: similar queries are more likely to share the same top configurations (defined in C.1). This is intuitive since semantically similar queries are more likely to ask about the same person or objects, or a highly overlapping clip of the video, which are captured by similar configurations. This observation suggests that query semantics play an essential role in determining the optimal configuration. Consequently, incorporating query-aware strategies can facilitate online configuration adaptation and improve responsiveness to diverse query intents.

D Discussion on Real-world Deployment

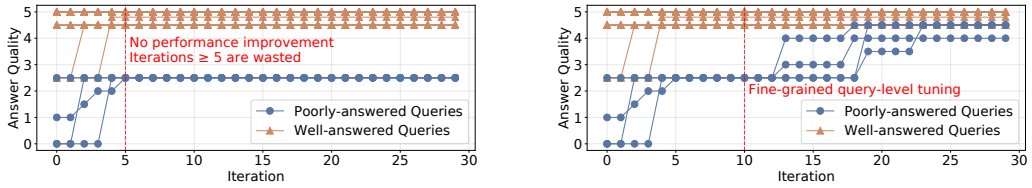
D.1 Cold-start Problem

In real-world scenarios, a newly uploaded video may have few or no user-submitted queries. For such cases, we can leverage inter-video similarity to suggest a promising configuration from a similar video, and subsequently fine-tune it once sufficient user queries become available. Moreover, numerous efficient query-generation pipelines have been developed in the VQA community. As reported in [11], using the Llama2-13B model on an NVIDIA A100 (80GB) GPU can produce approximately 20K QA pairs per hour. By utilizing these off-the-shelf pipelines, we can generate synthetic queries at low cost to facilitate modality adaptation for cold-start videos.

D.2 Storage Considerations

While the proposed online semantic adapter enhances performance, one may be concerned about its additional storage overhead, as it involves processing and retaining intermediate outputs from multiple modalities and configurations. This concern is effectively mitigated by two key factors. First, modalities other than VT have negligible storage costs: over 95% of storage consumption comes from VT (e.g., 90 MB for 64 frames), whereas other modalities are stored in text format and require less than 20 KB per knob setting. We empirically find that, for a subset of videos, accurate VQA can be achieved without storing VT, which substantially reduces storage demand.

Second, we observe significant redundancy across different configurations, particularly for storage-intensive VT. For example, 16 uniformly sampled frames form a strict subset of 32 uniformly sampled frames, implying that when both configurations are selected, it is sufficient to store only the outputs corresponding to the larger frame count. Similar redundancy also exists in other modalities, such as OBJ extracted under different sampling rates. We explicitly exploit this property by only considering overlapping configurations (e.g., $[4, 8, 16, ...]$ for *num-frame* knob) during tuning and incrementally merging modalities extracted under overlapping configurations, thereby reducing both the configuration search space and the storage requirements. As a result, Prysma achieves a storage reduction of 11–23% by avoiding redundant VT storage and by selectively omitting VT when other modalities suffice compared to pure vision-token-based design, which is commonly adopted by existing video LLMs [8, 34].



(a) Pure video-level tuning with TPE.

(b) Video and query-level tuning with our two-stage tuner.

Figure 8: Comparison of pure video-level tuning and our two-stage tuning.

E Pure Video-level Tuning vs. Two-stage Tuning

We present an example in Fig. 8a to illustrate the inefficiency of pure video-level tuning. The example shows the tuning process for a video with 16 queries, among which 4 are identified as poorly answered. Thanks to our knowledge base, most queries achieve high answer quality early in the tuning process: the majority quickly reach scores above 4.5 within the first five trials, with their individual tuning curves largely overlapping in the figure. However, the remaining queries exhibit stagnant performance, with scores hovering around 2.5 even after all 30 trials. This behavior arises because the video-level objective is dominated by the well-answered majority, causing the tuner to further refine configurations that benefit these queries while remaining largely agnostic to the poorly answered ones. As a result, purely video-level tuning makes little progress beyond the first five iterations, leading to nearly 83% of the tuning budget being wasted.

In Fig. 8b, we further present the tuning process of the same video as in Fig. 8a using our two-stage tuner to show its effectiveness. After 10 iterations, the stopping criterion is met, triggering the transition to Stage 2. In this stage, the tuner shifts from coarse-grained video-level optimization to fine-grained query-level tuning, explicitly focusing on the poorly answered queries. As shown in the figure, the answer quality of these poorly answered queries begins to improve steadily after the transition. This demonstrates that our two-stage tuner effectively utilizes the remaining budget to refine configurations for previously under-optimized queries, overcoming the stagnation observed in pure video-level tuning.

F Details of Two-Stage Configuration Tuning

We provide the pseudocode of our two-stage configuration tuning method in Algorithm 1. The algorithm accepts a video v and its query set Q_v, a tuning budget B, a video knowledge base KB, and a configuration space C as input. It outputs the tuning history h_v that contains both video-level and query-level optimal configurations. Each history entry $h \in h_v$ stores a configuration c, its achieved answer quality and TTFT. We first calculate the initial value and assign a budget for each promising MC (lines 1–4). Next, we perform coarse-grained video-level tuning with TPE tuner to seek video-level optimal configurations (lines 6–14). During tuning, the budget is intelligently adjusted by a dynamic budget allocation strategy (introduced below). The video-level tuning ends when the early-stopping condition is met, and we transition to stage 2, namely fine-grained query-level tuning (lines 15–20) for poorly answered queries using GA tuner. We next introduce the dynamic budget allocation strategy and how we tune the configurations for videos in the knowledge base.

F.1 Dynamic Budget Allocation

Given that different MCs may exhibit heterogeneous value and budget sensitivity, determining their priority and allocating trials effectively is crucial under a limited tuning budget. To this end, we design an adaptive budget allocation strategy (Algorithm 2) to distribute the tuning budget among promising MCs dynamically. Throughout the tuning process, we continuously record the best average performance achieved across the entire tuning set. After each trial, we estimate the Expected Improvement (EI) of the remaining MCs by randomly sampling multiple configurations within each MC and averaging their EIs calculated by Eq. (3). If the estimated EI of any remaining MC exceeds that of the current one and all of its modalities have been previously explored, the remaining budget of

Algorithm 1 Two-Stage Configuration Tuning

Require: video v, query set Q_v, total tuning budget B, knowledge base KB, configuration space $\mathcal{C}$.
Ensure: Tuning history h_v.

1: Initialize h_v as empty list, TPE tuner T, GA tuner G, $cur_stage \leftarrow 1$.
2: Calculate each MC's value based on Eq. (1) and KB.
3: Assign budget $b_i \leftarrow round\Big(B \cdot \frac{val_i^v}{\sum_j val_j^v}\Big)$ to the i-th mc.
4: $mc_queue \leftarrow sort_MC_by_budget()$.
5: **while** $B > 0$ **do**
6: **if** $cur_stage = 1$ **then**
7: $mc_i, b_i \leftarrow mc_queue.pop_max_budget()$
8: $\mathcal{C}' \leftarrow restrict_space(\mathcal{C}, mc_i)$
9: **while** $b_i > 0$ & $B > 0$ & $cur_stage = 1$ **do**
10: $h \leftarrow T.optimize(v, Q_v, \mathcal{C}')$, $h_v.add(h)$
11: $cur_stage \leftarrow 2$ if $early_stopping(h_v)$
12: $b_i \leftarrow b_i - 1$, $B \leftarrow B - 1$
13: $dynamic_budget_allocation(h_v, mc_i, mc_queue, \{b_0, \dots, b_i, \dots\})$
14: **end while**
15: **else**
16: $Q'_v \leftarrow poorly_answered_queries(Q_v, h_v)$
17: $G.init_population(h_v, Q'_v)$
18: $h \leftarrow G.optimize(v, Q'_v, \mathcal{C})$, $h_v.add(h)$
19: $B \leftarrow B - |Q'_v|/|Q_v|$
20: **end if**
21: **end while**
22: **Return** h_v.

Algorithm 2 Dynamic Budget Allocation

Require: mc_queue, current MC mc_i, tuning history so far h_v, budget allocation $\{b_0, \dots, b_i, \dots\}$.
Ensure: Updated budget allocation.

1: Update global best average performance on the tuning set in h_v
2: Estimate EI_j for each remaining $mc_j \in mc_queue$
3: $j^\star \leftarrow \arg\max_{mc_j \in mc_queue} EI_j$
4: **if** $EI_{j^\star} > EI_i$ **and** $all_modalities_explored(mc_{j^\star}, h_v)$ **then**
5: $b_{j^\star} \leftarrow b_{j^\star} + b_i$; $b_i \leftarrow 0$
6: Return
7: **end if**
8: **if** $b_i = 0$ and $has_improvement(mc_i, h_v)$ **then**
9: $k^\star \leftarrow \arg\min_{mc_k \in mc_queue}\ \mathrm{val}_k^v$
10: $\Delta b \leftarrow b_{k^\star}$; $b_{k^\star} \leftarrow 0$
11: $b_i \leftarrow b_i + \Delta b$
12: **end if**

the current MC is recycled and reallocated to the more promising MC (lines 4–7). For each MC, we check whether we have encountered the included modalities in the previous trial. If not, we use the best configuration within the same MC from the most similar video, together with Latin Hypercube Sampling (LHS) for initialization. Otherwise, we let the tuner itself decide the next configurations for evaluation.

When the budget allocated to current MC is exhausted, we check whether any improvement has been achieved during the tuning of the current MC. If improvement is observed since the latest budget update of the current MC, we consider this MC worth more trials and thus further recycle the allocated budget of the MC with the lowest value and reassign it to the current MC (lines 8–12). Otherwise, we proceed to the next MC in the mc_queue with the largest remaining budget. This dynamic budget allocation strategy allows us to both consider the interaction between different modalities and focus on the optimization within a small subspace.

Algorithm 3 Latency-Aware Configuration Pruning

Require: Configuration space $\mathcal{C}$; TTFT SLO τ.
Ensure: Monotonic knob set $\mathcal{K}_{\uparrow}$; banned configuration set $\mathcal{B}$.
1: Profile the relationship between input token count $N(c)$ and $\mathrm{TTFT}_{\mathrm{LLM}}(c)$ (LLM inference TTFT).
2: Derive a token budget function $B_\tau(c)$.
3: Estimate a feasible bound for VT knobs (e.g., $f_{\max}$ for `num-frame`) and prune all configurations whose VT knob exceeds the bound.
4: Identify latency-monotonic knobs $\mathcal{K}_{\uparrow}$ via measurement.
5: $\mathcal{B} \leftarrow \emptyset$.
6: **function** TOKENBUDGETPRUNING(c)
7: **if** $N_{lb}(c) > B_\tau(c)$ **then**
8: $U(c) \leftarrow \{\, c' \in \mathcal{C} \mid c \preceq c' \,\}$
9: $\mathcal{B} \leftarrow \mathcal{B} \cup U(c)$
10: **end if**
11: **end function**
12: **function** SLOVIOLATIONPRUNING(($c, ttft$))
13: **if** $ttft > \tau$ **then**
14: $U(c) \leftarrow \{\, c' \in \mathcal{C} \mid c \preceq c' \,\}$
15: $\mathcal{B} \leftarrow \mathcal{B} \cup U(c)$
16: **end if**
17: **end function**

F.2 Tuning Videos in Knowledge Base

For those videos used to initialize the knowledge base, we skip the process of scoring MCs and directly perform tuning over the entire configuration space (lines 5–22 in Algorithm 1) under the same tuning budget (i.e., 30 full-set trials). Although this strategy may slightly compromise the performance on a small subset of videos, it significantly enhances the tuning efficiency for the majority of videos.

G Details for Latency-aware Configuration Pruner

The offline modality orchestrator operates under a strict TTFT SLO τ and a limited tuning budget. A major source of inefficiency is that naive exploration frequently evaluates configurations that are guaranteed to violate the SLO, wasting expensive modality extraction and LLM inference. To mitigate this, we introduce a latency-aware configuration pruner that eliminates SLO-violating regions of the search space before and during tuning by exploiting a task-specific monotonicity property of LLM-based VQA. A (modality) configuration c is defined as a pair of (i) a modality combination (MC) $M(c)$ and (ii) modality-specific knob settings $\mathbf{k}(c)$. Let $\mathrm{TTFT}(c)$ denote the Time-to-First-Token latency when answering a query using c. The pseudocode of our pruning algorithm is provided in Algorithm 3.

Step 1: Profile token count-latency relationship (lines 1–2). We first profile the relationship between the input token count $N(c)$ and the *LLM inference* TTFT, denoted as $\mathrm{TTFT}_{\mathrm{LLM}}(c)$. Empirically, $\mathrm{TTFT}_{\mathrm{LLM}}$ is a stable, monotone increasing function of the prompt length, but its exact form can vary across models and serving stacks. Concretely, we run multiple samples that induce different input token counts and construct a mapping from prompt length n to TTFT:

$$\mathrm{TTFT}_{\mathrm{LLM}}(n) : \mathbb{N} \to \mathbb{R}_{+}, \tag{5}$$

which can be instantiated as a lookup table over token-count buckets, or a simple parametric fit (e.g., a linear or quadratic model [9]). The end-to-end TTFT can be decomposed as

$$\mathrm{TTFT}(c) = \mathrm{TTFT}_{\mathrm{LLM}}(N(c)) + O(c), \tag{6}$$

where $O(c)$ captures retrieval overheads in modern LLM-based VQA pipelines [20, 28] under configuration c. This overhead is empirically nearly constant given the specific modality combinations $M(c)$ and the retrieval model that determines how queries are rewritten/expanded for retrieval. Thus,

we maintain a simple lookup table for each MC and retrieval-model. Given the profiled mapping $\mathrm{TTFT}_{\mathrm{LLM}}(\cdot)$, we define a per-configuration token budget as the largest token count whose predicted inference TTFT fits the budget:

$$B_\tau(c) = \max\Big\{ n \in \mathbb{N} \;\Big|\; \mathrm{TTFT}_{\mathrm{LLM}}(n) \le \tau - O(c) \Big\}. \tag{7}$$

Any configuration with $N(c) > B_\tau(c)$ is deemed SLO-violating and can be pruned without execution.

Step 2: Offline pre-prune (line 3). Before tuning, we start with the VT modality, as VT typically dominates the input length due to the large number of tokens generated per frame, and its per-frame token contribution is consistent and easy to estimate. Let f denote the VT knob `num-frame` and $N_{\mathrm{VT}}(f)$ denote the estimated VT token contribution under f. We compute the maximum feasible frame count and prune all configurations whose VT knob exceeds this bound. This one-shot pruning removes a large portion of the space that would inevitably violate the TTFT SLO.

Step 3: Identify latency-monotonic knobs (line 4). To enable stronger pruning, we identify a set of latency-monotonic knobs $\mathcal{K}_\uparrow$. For each candidate knob k, we fix all other parameters and sweep ordered values $v_1 < \cdots < v_T$ for k, then measure the resulting token counts $N(c_{k,v_t})$. If $N(c_{k,v_{t+1}}) \ge N(c_{k,v_t})$ holds for all t, we mark k as monotonic increasing and include it in $\mathcal{K}_\uparrow$. Note that monotonicity can hold in either direction: for some knobs, increasing the knob value may decrease the extracted content and thus reduce the token count. For simplicity of exposition, we focus on monotone increasing knobs and collect them into $\mathcal{K}_\uparrow$; monotone-decreasing knobs can be handled analogously by reversing the inequality in the dominance relation and the derived infeasible set.

Step 4: Online dynamic pruning (lines 6–17). We perform online pruning after each tuning trial finishes. Given the newly observed extraction statistics and the end-to-end TTFT outcome, it updates the candidate set by pruning configurations that are provably infeasible under the TTFT SLO. We initialize a banned set $\mathcal{B}$ that contains all SLO-violating configurations and define a partial order $\preceq$ over configurations:

$$\begin{aligned} c \preceq c' \;\Leftrightarrow\; \big(M(c) \subseteq M(c')\big) \;\wedge\; \big(\forall k \in \mathcal{K}_\uparrow(M(c)) : k(c) \le k(c')\big) \\ \wedge\; \big(\forall k \notin \mathcal{K}_\uparrow(M(c)) : k(c) = k(c')\big). \end{aligned} \tag{8}$$

Intuitively, $c \preceq c'$ means that c' uses a superset of modalities and is no smaller on all latency-monotonic knobs and shares identical values for all non-monotonic knobs associated with their shared modalities; hence c' is expected to have no fewer input tokens and no lower $\mathrm{TTFT}_{\mathrm{LLM}}$ than c. For any configuration c, we denote its *upper set* (i.e., dominated supersets) as

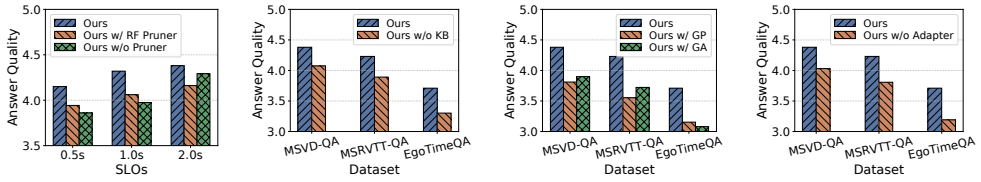
$$U(c) \;=\; \{\, c' \in \mathcal{C} \mid c \preceq c' \,\}. \tag{9}$$

With the above definition, we design two complementary strategies for efficient online pruning:

(i) Token-budget pruning (lines 6–10). After each trial, we incorporate the token count of the newly extracted configuration to tighten token-count estimates for other candidates. For any candidate c, we compute a conservative lower bound $N_{\mathrm{lb}}(c)$ using token statistics from extracted configurations and the monotonicity of knobs in $\mathcal{K}_\uparrow$. If $N_{lb}(c) > B_\tau(c)$, then c is budget-infeasible at the LLM stage and can be pruned without running further extraction or inference. Moreover, by dominance monotonicity, every configuration in $U(c)$ is also budget-infeasible; therefore we prune $U(c)$ as well, i.e., $\mathcal{B} \leftarrow \mathcal{B} \cup U(c)$.

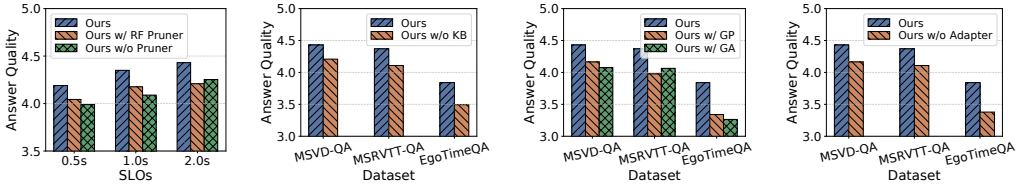
(ii) SLO-violation pruning (lines 11–17). If the current trial evaluates a configuration c and observes $ttft > \tau$, then c is end-to-end SLO-violating. By the same dominance argument, every configuration in $U(c)$ cannot be faster than c and is also SLO-violating. Thus, upon an SLO violation, we additionally prune $U(c)$ and update the banned set accordingly.

Discussion and benefits. This pruner is training-free: it does not require learning a latency predictor and thus avoids prediction-induced over-pruning. Instead, it relies on the stable token–TTFT trend and monotonic dominance relations. Empirically, our lightweight pruner can efficiently reduce the proportion of SLO-violating trials under the same budget by 22%–65%, depending on different SLOs and videos.



(a) (b) (c) (d)

Figure 9: Ablation study on (a) configuration pruner, (b) knowledge base, (c) two-stage tuner, (d) semantic adapter for LLaVA-Video-7B.



(a) (b) (c) (d)

Figure 10: Ablation study on (a) configuration pruner, (b) knowledge base, (c) two-stage tuner, (d) semantic adapter for VideoLLaMA2-7B.

H Evaluation Setting

For MSVD-QA and MSRVTT-QA, we select 1k videos with the largest number of queries, while for EgoTimeQA, we use the whole dataset (about 300 videos). Among them, 10% of videos are used to construct the knowledge base. The cluster number is chosen automatically by maximizing the Silhouette Score. To better reflect real-world scenarios, we select a subset of the queries as the training set for tuning and report the answer quality on the entire query set. This evaluation strategy captures both the tuning effect on seen queries and the generalization performance on unseen queries. Meanwhile, the performance on seen queries remains meaningful, as industrial LLM inference services often reuse responses for semantically similar inputs to reduce inference costs [4, 10, 31]. For MSVD-QA and MSRVTT-QA, where the number of queries per video is relatively limited, we use 40% of the queries as the training set, while for EgoTimeQA, we use 30%. During tuning, we include the modalities and their knobs listed in Table 1, along with two retrieval-related knobs: *retrieval-thresh* $\in [0, 1]$, which controls whether retrieved candidates are selected into the prompt, and *retrieval-model* $\in \{\text{BERT}, \text{LLM}, \text{None}\}$, which determines how queries are rewritten/expanded for retrieval. These knobs are included to account for the substantial cost of SOTA LLM-based retrieval methods [20, 28].

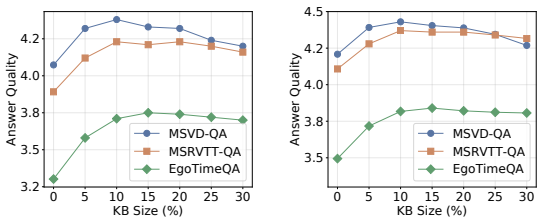
I Experiment Results

I.1 Ablation Study

In this ablation study, we evaluate the four key components of Prysma on both LLaVA-Video-7B and VideoLLaMA2-7B in Fig. 9 and 10. We use LLaVA-Video-7B as an illustrative example below, while similar results can also be observed on VideoLLaMA2-7B.

Latency-Aware Configuration Pruner. We compare the proposed pruner with two variants: *Ours w/ RF Pruner* (SCOOT-based random forest) and *Ours w/o Pruner*. On MSVD-QA under three SLOs, our approach outperforms *Ours w/ RF Pruner* by 6.8%, 4.1%, and 6.2%, and *Ours w/o Pruner* by 10.6%, 7.3%, and 3.1%, respectively. The RF Pruner performs worse under looser SLOs due to inaccurate predictions, while our training-free pruner leverages task-specific insights from LLM inference and multimodal VQA.

Video Knowledge Base. A baseline without the knowledge base (*Ours w/o KB*) is evaluated across three datasets under a 2.0s SLO. Incorporating the knowledge base improves answer quality by



(a) LLaVA-Video-7B (b) VideoLLaMA2-7B

Figure 11: Sensitivity analysis on the size of video knowledge base.

7.5%, 8.6%, and 12.3%, and enhances tuning efficiency by warm-starting with high-value MCs and reducing exploration in low-performing regions.

Two-Stage Tuner. We compare our two-stage tuner with *Ours w/ GP* (GP-based BO) and *Ours w/ GA* (Genetic Algorithm). Our tuner outperforms GP by 14.9%, 19.1%, and 17.6%, and GA by 12.4%, 13.7%, and 20.5%, on three datasets under a 2.0s SLO. This module bridges offline tuning and online adaptation by collecting valuable tuning history.

Semantic Adapter. Compared to *Ours w/o Adapter* (using video-level optimal configurations), the semantic adapter boosts answer quality by 8.7%, 11.3%, and 16.2% across the datasets. Without it, video-level configurations are often suboptimal; this lightweight module enables finer-grained, query-aware optimization.

In Fig. 10, we report the ablation results on VideoLLaMA2-7B. The ablation results on VideoLLaMA2-7B exhibit trends highly consistent with those on LLaVA-Video-7B. In particular, each proposed component contributes positively to the overall performance, and removing any module leads to noticeable degradation. The gains are generally more pronounced under stricter TTFT SLOs and on more challenging datasets (e.g., EgoTimeQA), indicating that the benefits of our latency-aware pruning, knowledge reuse, multi-stage tuning, and query-level adaptation are not model-specific but generalize well across different video LLM backbones.

I.2 Sensitivity Analysis

I.2.1 Video Knowledge Base Size

One key hyperparameter of Prysma is the proportion of videos used to construct the video knowledge base. To assess its sensitivity, we vary the proportion of videos used to initialize the knowledge base and evaluate performance on three datasets (Fig. 11). The results on both video LLMs show that performance improves as the knowledge base grows, peaking when about 10–15% of the videos are used. A moderate number of representative videos thus provides sufficient diversity for effective knowledge transfer. When the knowledge base becomes too large, performance can decline as more videos end up being tuned from scratch under the same budget (reducing search efficiency), while fewer remaining videos can benefit from the accumulated knowledge. In conclusion, these results suggest constructing the video knowledge base with an appropriate proportion of the videos to balance diversity and efficiency.

I.2.2 Modality Pool

We further evaluate the sensitivity of Prysma to the available modality pool by explicitly restricting the set of modalities that can be selected during adaptation. The experiments are conducted on EgoTimeQA with LLaVA-Video-7B under a moderate 1.0s TTFT SLO. For each modality pool, we compare Prysma with the strongest baseline under the same pool, so that the comparison isolates the benefit of adaptive modality configuration rather than the benefit of access to additional modalities.

As shown in Table 2, Prysma consistently outperforms the strongest baseline across all modality pools. Even with only two modalities, Prysma improves answer quality from 2.25 to 2.80 under VT+C and from 2.32 to 2.98 under VT+OBJ, corresponding to relative gains of 24.3% and 28.5%, respectively. When the pool is expanded to three modalities, Prysma achieves larger gains: 33.4% for VT+C+OBJ and 31.8% for VT+C+OCR. These results indicate that Prysma remains effective

Table 2: Performance comparison across different modality pools.

Modality Pool	Strongest Baseline	Prysma	Improvement
VT + C	2.25	2.80	24.3%
VT + OBJ	2.32	2.98	28.5%
VT + C + OBJ	2.44	3.25	33.4%
VT + C + OCR	2.38	3.13	31.8%
VT + C + OCR + OBJ + ASR	2.62	3.62	38.1%

Table 3: Overhead of major modules in Prysma.

Dataset	Two-stage Tuner	Video Knowledge Base	Config Pruner	Semantic Adapter
MSVD	2.3s	154s	42.3ms	9.3ms
MSRVTT	2.2s	109s	43.1ms	8.1ms
EgoTimeQA	2.4s	162s	44.5ms	12.2ms

even when the system is deployed with a restricted set of extractors, as it can still adaptively choose suitable modality combinations and knob settings within the reduced search space.

The gains further increase when the full modality pool is available. With VT+C+OCR+OBJ+ASR, Prysma improves answer quality from 2.62 to 3.62, yielding the largest relative improvement of 38.1%. This trend suggests that richer modality pools provide more opportunities for query- and video-specific adaptation, especially on long and challenging videos where complementary cues such as captions, OCR text, object records, and ASR transcripts can capture different aspects of the content. Overall, the results demonstrate that Prysma is robust to restricted deployment settings while benefiting further from a more comprehensive modality pool.

I.3 Overhead Analysis

I.3.1 Runtime Overhead

In Table 3, we report the runtime overhead of Prysma's four major components, namely the two-stage configuration tuner, video knowledge base, latency-aware configuration pruner, and online semantic adapter. Since their operations are generally model-agnostic, we report the overhead on three evaluated datasets.

The two-stage tuner introduces a small per-trial offline decision overhead (2.2–2.4 s across datasets) to propose the next configuration. This cost is minor compared to the subsequent end-to-end trial evaluation, which includes modality extraction and LLM inference that typically requires more than ten minutes. Since the tuning budget is fixed, the tuner overhead scales linearly with the budget and does not become a bottleneck in practice.

The latency-aware configuration pruner runs alongside tuning and adds only 42–45 ms per step. This overhead is negligible relative to the cost of evaluating a trial, while effectively preventing wasted exploration of configurations that violate the target SLO.

Constructing the video knowledge base is a one-time offline cost comprising video decoding, feature extraction and K-means clustering. We will discuss the tuning overhead in the next part. This cost is about 154s for MSVD-QA, 109s for MSRVTT-QA datasets and 162s for EgoTimeQA with long videos. This cost only occurs once for the whole video set and is negligible compared to the tuning overhead. This cost is worthwhile because it significantly improves tuning efficiency.

At inference time, the semantic adapter performs query filtering and configuration scoring based on semantic similarity, incurring millisecond-level overhead (8.1–12.2 ms), which is negligible under common TTFT SLOs (0.5–2.0 s) and remains imperceptible to end users.

Overall, Prysma introduces minimal additional overhead. Computationally expensive operations are confined to bounded-budget offline tuning and one-time knowledge base construction, while online adaptation remains strictly millisecond-scale relative to interactive latency SLO.

I.3.2 Tuning Overhead

Under a 30-trial budget, the average offline tuning wall-clock time is approximately 9 minutes per short video on MSVD-QA and MSRVTT-QA, and about 14 minutes per long video on EgoTimeQA. Although this reflects the elapsed time per video, the pipeline does not require sustained full-GPU occupancy. Our profiling shows that the average GPU utilization per video remains relatively low (around 15%), as a considerable portion of the time is spent on video decoding, modality extraction, and information retrieval rather than full expensive LLM inference. This indicates that co-scheduling multiple videos in parallel is feasible in practical deployments. While a comprehensive study of parallel and pipelined execution is beyond the scope of this work, it represents an important direction for future research.

We also provide a discussion on whether such a tuning cost is worthwhile. Prysma trades a bounded one-time offline tuning cost for persistent savings in required model scale and serving cost. Our additional result on MSVD-QA suggests that matching Prysma's answer quality with a fixed pipeline requires scaling the backbone to at least 34B (i.e., PLLaVA [30]), which entails at least $5\times$ higher inference latency and GPU memory usage. In our measurement on MSVD-QA, this further translates into more than a $10\times$ reduction in queries per second due to the much longer context input and extra computation resource demand of the 34B model. If we conservatively approximate the 34B model as incurring $10\times$ the per-query serving cost/latency of 7B, then each served query saves roughly 9 times the cost of a 7B query. With our measured offline tuning wall-clock of about 9 minutes, the one-time tuning cost is amortized after roughly $540/(9c)$ queries, where c is the per-query cost of the 7B model—about 60 queries when $c = 1s$, and about 30 queries when $c = 2s$. By comparison, a trending video on YouTube can have thousands of new comments within just one hour. Note that this estimate is conservative because it only accounts for single-query latency. In practice, a 34B model would also require substantially more memory, reduce concurrency, and increase queuing delay, thereby further hurting throughput and making SLO violations more likely. In conclusion, the tuning cost can be effectively amortized in real-world production environments, making Prysma highly practical.

I.4 Human Preference Calibration

To further examine whether the answers optimized by Prysma are genuinely preferred by humans, we conducted a preliminary human preference study. The goal of this study is to verify that the answer optimized by Prysma is consistently better than those produced by competing methods under both LLM-based evaluation and human judgment.

We adopted a pairwise preference protocol, in which annotators were asked to choose their preferred answer given the same video, question, and candidate answers. Compared with absolute rating schemes, pairwise comparison is generally more intuitive for annotators and can reduce annotation noise caused by individual differences in score calibration. It also offers better reproducibility and lower annotation cost.

We sampled 120 questions from 12 videos, with 4 videos from each dataset. For each question, we compared the answer optimized by Prysma against the answer from the strongest baseline. Five human annotators independently selected the better answer based on the video and the question. The annotators were volunteers from our local academic community and none were authors. Answers were presented in a randomized order for each question and annotator; annotators were blinded to the producing method and could select a tie. We asked them to first watch the current video and then gave each annotator the same instruction: Given the same video and question, please decide which answer is better. For comparison with the LLM-based evaluation results, we also converted point-wise LLM scores into pairwise preferences by selecting the answer with the higher score. The study produced 600 pairwise judgments across 120 questions, with 40 questions from each dataset.

The agreement rate between LLM-induced preferences and human judgments is 92%. The inter-annotator agreement is 0.76 as measured by Fleiss' kappa over three categories (win, tie, and loss), indicating substantial agreement among annotators and suggesting that the human judgments are consistent and reliable. As shown in Table 4, per-question majority vote gives 83 wins, 21 ties, and 16 losses for Prysma, with a 69% win rate and a 95% bootstrap confidence interval of 61–78%. Per-annotator win rates range from 66% to 72% (median 68%), and per-dataset win rates are 60%, 70%, and 78% on MSVD-QA, MSRVTT-QA, and EgoTimeQA, respectively. More importantly, both

evaluation protocols lead to the same conclusion: answers optimized by Prysma are consistently preferred over the strongest baseline. This demonstrates that Prysma's optimization yields answers that are not only rated higher by LLM evaluators, but also better aligned with human preferences.

Participation was voluntary and uncompensated, and no personally identifiable information was collected.

Table 4: Human-preference study (120 questions, 12 videos, five annotators).

Comparison	Win	Tie	Loss	Win rate (95% CI)	Fleiss' κ
Prysma vs. strongest baseline	83	21	16	69% (61–78%)	0.76

I.5 Held-out Evaluation

We evaluate Prysma's ability to answer new queries by separately measuring answer quality on queries not used during tuning. For MSVD-QA and MSRVTT-QA, 40% of queries are used for tuning and the remaining 60% form the held-out set; for EgoTimeQA, the split is 30%/70%. All methods are evaluated on the same held-out queries using the same scoring protocol. Tables 5 and 6 compare Prysma with the strongest baseline in each setting across three datasets, two video LLMs, and three TTFT SLOs.

Table 5: Held-out quality: best baseline / Prysma (relative gain), LLaVA-Video-7B.

Dataset	0.5 s	1.0 s	2.0 s
MSVD-QA	3.46 / 4.05 (+17%)	3.58 / 4.22 (+18%)	3.69 / 4.28 (+16%)
MSRVTT-QA	3.09 / 3.73 (+21%)	3.19 / 4.01 (+26%)	3.25 / 4.14 (+27%)
EgoTimeQA	2.23 / 3.22 (+44%)	2.50 / 3.53 (+41%)	2.76 / 3.62 (+31%)

Table 6: Held-out quality: best baseline / Prysma (relative gain), VideoLLaMA2-7B.

Dataset	0.5 s	1.0 s	2.0 s
MSVD-QA	3.74 / 4.09 (+9.4%)	3.83 / 4.25 (+11%)	3.89 / 4.33 (+11%)
MSRVTT-QA	3.48 / 4.03 (+16%)	3.62 / 4.10 (+13%)	3.81 / 4.28 (+12%)
EgoTimeQA	2.68 / 3.44 (+28%)	2.73 / 3.63 (+33%)	2.92 / 3.75 (+28%)

Prysma consistently outperforms the strongest baseline in all 18 settings, improving held-out answer quality by approximately 9–44%. At the 1.0 s TTFT SLO, the gains on MSVD-QA, MSRVTT-QA, and EgoTimeQA are 18%, 26%, and 41% for LLaVA-Video-7B, and 11%, 13%, and 33% for VideoLLaMA2-7B. The largest improvements occur on EgoTimeQA, where Prysma achieves gains of 31–44% and 28–33% across the three SLOs for the two models, respectively. For example, with LLaVA-Video-7B at 1.0 s, Prysma raises held-out quality from 2.50 to 3.53. These improvements show that Prysma's advantage extends to unseen queries across different models and latency budgets.

Held-out queries yield slightly lower scores than seen queries. At the 1.0 s TTFT SLO, Prysma's seen-query scores on MSVD-QA, MSRVTT-QA, and EgoTimeQA are 4.47/4.23/3.82 for LLaVA-Video-7B and 4.50/4.32/3.92 for VideoLLaMA2-7B, compared with held-out scores of 4.22/4.01/3.53 and 4.25/4.10/3.63, respectively. Relative to evaluation on the mixed set of seen and held-out queries, quality decreases by about 1% for untuned VT, 2.5% for Prysma, and 3.5–5% for tuned baselines. Prysma thus retains its gains despite the change in evaluation queries: for LLaVA-Video-7B at 1.0 s, the relative improvements increase from 16%/22%/38% under mixed-set evaluation to 18%/26%/41% on held-out queries. This generalization is consistent with the design of the online semantic adapter (§ 3.3), which transfers configuration preferences from semantically related tuning queries to each incoming query.

I.6 Judge Robustness and Evaluation Prompt

We evaluate the consistency of Prysma's quality improvements across different LLM judges. We use `gpt-3.5-turbo-0125` as the tuning signal, then re-score the final held-out answers with GPT-4o

and Vicuna-13B-v1.5 while keeping the generated answers fixed. This evaluation includes both a proprietary judge and an open-weight judge independent of the tuning model. Table 7 reports the average answer quality across MSVD-QA, MSRVTT-QA, and EgoTimeQA for LLaVA-Video-7B at a 1.0 s TTFT SLO.

Table 7: Cross-judge held-out evaluation (three-dataset average).

Judge	Best baseline	Prysma	Gain
GPT-3.5-turbo-0125	3.09	3.92	+27%
GPT-4o	2.90	3.70	+28%
Vicuna-13B-v1.5	3.12	3.89	+25%

Prysma outperforms the strongest baseline under all three judges. GPT-4o assigns average scores of 3.70 to Prysma and 2.90 to the strongest baseline, a 28% improvement, while Vicuna-13B-v1.5 gives scores of 3.89 and 3.12, a 25% improvement. These gains are comparable to the 27% improvement measured by the tuning judge. Although the absolute scores vary across judges, Prysma's relative advantage remains consistent. Per-question scores also correlate strongly with those of the tuning judge, with a Spearman correlation of $\rho = 0.83$. Together, these results support the robustness of Prysma's quality improvements to the choice of evaluator.

For answer-quality scoring, we follow the official Video-ChatGPT correctness evaluation template. The system message instructs the judge to assess whether the predicted answer meaningfully matches the reference, accepting synonyms and paraphrases. The user message provides the question, reference answer, and predicted answer, and requests a Python dictionary containing a yes/no correctness decision and an integer score from 0 to 5, where 5 indicates the strongest agreement with the reference. We use temperature 0 for scoring. Across five repeated evaluations of identical answers, the mean absolute score difference is 0.04, indicating low scoring variability.

I.7 Larger Video LLMs

We additionally evaluate PLLaVA-13B and Qwen3-VL-32B-Instruct, both served natively in BF16. We use the 2.0 s TTFT SLO, one of the standard settings in our main evaluation. Table 8 reports the dataset-level results on the three open-ended benchmarks. Prysma outperforms the strongest baseline by 14%, 23%, and 31% with PLLaVA-13B, and by 15%, 22%, and 34% with Qwen3-VL-32B-Instruct on MSVD-QA, MSRVTT-QA, and EgoTimeQA, respectively. The advantage is therefore preserved for both larger backbones and is largest on the more challenging EgoTimeQA dataset, indicating that increasing model size does not remove the benefit of adapting the modality pipeline.

Table 8: Answer quality with larger models at a 2.0 s TTFT SLO.

Setting	VT	Best baseline	Prysma
MSVD-QA, PLLaVA-13B	3.49	3.88	4.43 (+14%)
MSRVTT-QA, PLLaVA-13B	3.29	3.48	4.29 (+23%)
EgoTimeQA, PLLaVA-13B	2.36	2.98	3.90 (+31%)
MSVD-QA, Qwen3-VL-32B-Instruct	3.60	3.94	4.54 (+15%)
MSRVTT-QA, Qwen3-VL-32B-Instruct	3.41	3.55	4.33 (+22%)
EgoTimeQA, Qwen3-VL-32B-Instruct	2.53	3.08	4.12 (+34%)

I.8 Raw Answer Quality and SLO Compliance

To better understand the sources of Prysma's gains, we examine raw answer quality and SLO compliance separately. We also extend the TTFT SLO range beyond the 0.5, 1.0, and 2.0 s settings in the main evaluation to cover 0.05–10 s, assessing performance under both tighter and more relaxed latency budgets. Table 9 reports three metrics separately: raw answer quality, average TTFT, and the fraction of requests that exceed the TTFT SLO. Raw answer quality is computed before applying any penalty to SLO-violating requests. Each entry compares Prysma with the strongest baseline at the same SLO. Section 4 describes the SLO-handling mechanism used by each baseline.

Table 9: Raw answer quality and SLO compliance on MSVD-QA with LLaVA-Video-7B. Each cell reports Prysma / the strongest baseline.

TTFT SLO	Raw quality	Avg. TTFT (s)	Violation rate
0.05 s	2.30 / 2.05	0.040 / 0.044	12.18% / 37.49%
0.1 s	3.13 / 2.66	0.084 / 0.092	4.79% / 22.46%
0.2 s	3.71 / 3.15	0.17 / 0.18	1.87% / 10.97%
0.5 s	4.08 / 3.63	0.42 / 0.45	0.80% / 4.87%
1.0 s	4.24 / 3.69	0.81 / 0.88	0.53% / 3.60%
2.0 s	4.29 / 3.74	1.52 / 1.68	0.14% / 1.68%
3.0 s	4.31 / 3.75	1.71 / 1.95	0.03% / 0.79%
10 s	4.32 / 3.75	1.72 / 1.96	0.00% / 0.00%

Prysma achieves higher raw answer quality and lower average TTFT at every evaluated SLO, with quality improvements of approximately 12–18%. At the 1.0 s SLO, it improves raw quality from 3.69 to 4.24 while reducing average TTFT from 0.88 to 0.81 s and the violation rate from 3.60% to 0.53%. Across the main 0.5–2.0 s SLOs, Prysma keeps the violation rate below 1%, compared with 1.68–4.87% for the strongest baseline. These results show that modality adaptation improves answer quality while delivering more responses within the requested latency budget.

The wider SLO range also reveals how performance changes under very strict and relaxed latency budgets. At 0.05 s, both methods incur frequent violations, but Prysma reduces the violation rate from 37.49% to 12.18% while retaining higher raw quality. As the SLO is relaxed beyond 2–3 s, answer quality begins to saturate: Prysma's score increases only from 4.31 at 3.0 s to 4.32 at 10 s. At the latter setting, its average TTFT remains 1.72 s, reflecting the finite modality pool and context budget. Both methods have zero observed violations at 10 s, while Prysma retains a raw-quality advantage of 4.32 versus 3.75.

I.9 Performance under Batched Serving

We evaluate Prysma under batched serving on EgoTimeQA with LLaVA-Video-7B, using a batch size of four and a 2.0 s TTFT SLO. For each incoming query, the online adapter selects a modality configuration and assembles the prompt, after which the serving engine batches the resulting prompts for inference. Each request therefore retains its own configuration within the batch.

To account for the latency introduced by batching, we calibrate the token-to-TTFT profile at the target batch size before tuning. At matched input length, prefilling four prompts takes approximately $2.43\times$ the time required for a single prompt. The 2.0 s TTFT SLO thus corresponds to an input-token budget derived from a single-request TTFT of approximately 0.82 s. Prysma then uses this budget for configuration pruning and tuning.

Table 10: Batched serving (batch size 4, 2.0 s TTFT SLO; EgoTimeQA, LLaVA-Video-7B).

Method	Answer quality	Avg. TTFT at batch 4 (s)
Best baseline	2.40	1.93
Prysma	3.42 (+43%)	1.82

As shown in Table 10, Prysma achieves an answer quality of 3.42, improving over the strongest baseline's 2.40 by 43%, while reducing average TTFT from 1.93 s to 1.82 s. These results demonstrate that modality adaptation remains effective under batched serving. The calibrated profile applies to the evaluated serving conditions; changes in concurrency, hardware, or scheduling require updating the profile and its corresponding token budget. Extending this calibration to dynamically varying workloads is a direction for future work.

I.10 Cold-start Performance and Transferability of Prysma

In real-world deployments, service providers may encounter out-of-distribution videos or newly uploaded videos with insufficient query data for per-video tuning. In such cold-start scenarios, Prysma

Table 11: Cold-start results.

Dataset	VT-only	Prysma (cold start)	Prysma
MSVD-QA	3.5	4.0 (+14.3%)	4.3 (+22.9%)
MSRVTT-QA	3.2	3.8 (+18.8%)	4.1 (+28.1%)
EgoTimeQA	2.1	3.3 (+57.1%)	3.7 (+76.2%)

can reuse configurations discovered from previously tuned videos by exploiting both inter-video visual similarity and inter-query semantic similarity.

We first evaluate this zero-tuning setting on the three main datasets. For each dataset, we sample an extra 10% of videos for testing. For each sampled video, we retrieve the most visually similar tuned video using CLIP features. For each query, we then select the configuration associated with the most semantically similar tuned query from the retrieved video. As shown in Table 11, this cold-start strategy already yields clear gains over the VT-only baseline, improving answer quality by 14.3%, 18.8%, and 57.1% on MSVD-QA, MSRVTT-QA, and EgoTimeQA, respectively. Although full Prysma tuning achieves the best performance, the cold-start variant closes a substantial fraction of the gap without any additional tuning on the target video, demonstrating that the learned configurations are reusable across visually and semantically related instances.

We further stress-test the transferability of Prysma on Video-MME, a more challenging dataset with a different evaluation protocol and broader video-duration distribution. Video-MME is not well-suited for direct Prysma tuning because each video contains only three questions and the benchmark focuses on multiple-choice evaluation, whereas Prysma is designed for open-ended VQA serving scenarios with multiple queries per video. Nevertheless, transferring Prysma-discovered configurations (following the same steps in cold start) to Video-MME improves accuracy from 46.2 to 57.1 (23.6% relative gain) on VideoLLaMA2, and improves accuracy from 63.2 to 69.4 (9.8% relative gain) on LLaVA-Video-7B. These results suggest that Prysma's configurations generalize beyond the datasets used for tuning, and remain effective even under a different question format and video distribution.

J Broader Impacts

Prysma positively impacts society by improving the efficiency and accuracy of video question answering (VQA) systems. By intelligently adapting multimodal configurations for large language model-based VQA, Prysma enhances answer quality while meeting strict latency requirements. This can benefit educational tools, accessibility services (e.g., automated video summarization or captioning), and research applications by enabling faster, more reliable understanding of diverse video content, reducing computational waste, and supporting scalable deployment across varied video datasets.

At the same time, more efficient VQA systems may be deployed in settings where video content contains sensitive or personal information. Although Prysma does not introduce new data collection mechanisms or new generative capabilities, improving the efficiency of video understanding could make large-scale video analysis easier to deploy. If used without appropriate safeguards, such systems may raise privacy concerns or be applied to surveillance-oriented scenarios. In addition, because Prysma optimizes answer quality under latency constraints, its outputs may still inherit errors, biases, or hallucinations from the underlying video LLMs and modality extractors, especially when videos contain ambiguous scenes, low-quality audio, or underrepresented content. These limitations could lead to misleading interpretations if users over-trust the generated answers.

We believe these risks can be mitigated through standard deployment practices, such as applying Prysma only to videos for which users have appropriate rights or consent, avoiding high-stakes automated decision-making without human review, auditing the underlying VQA models across diverse video domains, and clearly communicating uncertainty or failure cases to end users. Since Prysma is a configuration adaptation framework rather than a new dataset or foundation model, these considerations primarily concern downstream deployment contexts. For deployment, providers should calibrate the latency profile for their hardware and serving regime and monitor per-request TTFT compliance as workloads change.

K Limitations

A limitation of the Prysma framework is its reliance on a certain number of existing queries for effective tuning. This requirement may not always be met in scenarios involving newly uploaded videos with few or no user queries. However, this limitation can be addressed by leveraging mature pipeline tools to generate synthetic queries at a low cost, which has been demonstrated to facilitate modality adaptation. Additionally, experiments have shown that transferring configurations based on video similarity yields comparable results, suggesting that this issue does not significantly impact the overall effectiveness of the system.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly and accurately present the contributions and scope in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of Prysma in § K.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The methods proposed in this paper are based on empirical observations and do not involve theoretical proof.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our experiments and implementation in § 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide detailed descriptions of our experiments and implementation in § 4. Additionally, we submit our code in the supplementary materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide the detailed experimental setting in Appendix H.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The main experiment results in § 4 are accompanied by error bars that denote standard deviation over 5 random seeds.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide detailed descriptions of our experiments and implementation, including computation resources in § 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We follow the guidelines of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impacts of Prysma in § J.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work does not involve the release of new models or data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The models, datasets, and codebase used in our work comply with open-source licenses and can be used for scientific research.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release an anonymized implementation of Prysma as supplementary material. The code package includes documentation, environment setup instructions, and scripts for reproducing the main experiments. We do not release new datasets or model weights.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: We describe the human preference study protocol in Appendix I.4, including the annotation task, the number of annotators, and the exact instruction given to annotators. The study was conducted with volunteer annotators rather than paid crowd workers. Participation was voluntary and uncompensated, and no personally identifiable information was collected.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: This study involved voluntary anonymous questionnaires. Participants were informed about the purpose of the study and consented before participation. No personally

identifiable information was collected, and the study posed minimal risk to participants. According to our institution's policy, formal IRB approval was not required for this type of anonymous survey study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this work does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.