RESADAPT: ADAPTIVE RESOLUTION FOR EFFICIENT VIDEO REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

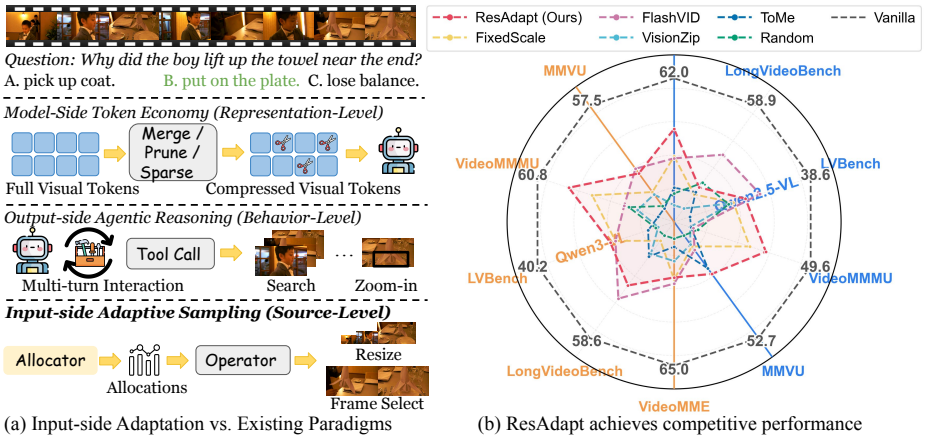
Multimodal Large Language Models (MLLMs) typically encode video frames at uniform spatial resolution, forcing fine visual detail and temporal sampling density into a rigid trade-off under constrained token budgets. Yet, visual relevance in video is temporally sparse and query-dependent: critical evidence concentrates in brief moments, while repetitive background motion carries little useful signal. We introduce **ResAdapt**, an input-side framework that treats per-frame resolution as an adaptive variable determined *prior to* visual feature extraction. A lightweight Allocator pairs coarse preview features with the user query to predict continuous per-frame scales, coarsening redundant intervals while allocating denser token capacity to answer-critical moments. To optimize allocation without frame-level supervision under non-differentiable processor token collation, we formulate the task as a continuous-action contextual bandit. We train the Allocator via **Cost-Aware Policy Optimization (CAPO)**, combining group-relative advantage with dynamic-pivot asymmetric cost shaping, temporal-similarity regularization, and concentration penalties. Across diverse video benchmarks, ResAdapt improves the accuracy–budget frontier: on Qwen2.5-VL at 128 frames, it matches dense full-resolution accuracy on the six-benchmark average (56.28 vs. 55.85) at only 22.9% token retention, while exceeding dense performance on VideoMME and VideoMMMU. Furthermore, spatial savings are fungible with temporal coverage—reinvesting tokens into $16\times$ denser sampling of the same video raises VideoMME by 9.7 points under a matched token budget. Permutation controls show that deliberate resolution placement across frames contributes to these reasoning gains. Finally, the frozen Allocator transfers zero-shot across diverse backbones and cuts 128-frame latency by 59.5% at 28.2% retention on Qwen2.5-VL. Code is available at `https://anonymous.4open.science/r/ResAdapt`.

1 INTRODUCTION

Long-video reasoning requires a Multimodal Large Language Model (MLLM) to follow narratives across extended horizons while discerning fine visual evidence that decides the answer (Wang et al., 2025; Hu et al., 2025). Standard video MLLMs inherit an image-centric paradigm: they treat video as concatenated independent images (An et al., 2026; Yang et al., 2026) and encode every sampled frame at an identical spatial resolution and token quota (Wang et al., 2024a; Bai et al., 2025b). This uniform allocation overlooks temporal sparsity. Natural video exhibits substantial inter-frame redundancy; static stretches carry little incremental signal, whereas critical clues concentrate in brief moments (Tang et al., 2026; Yang et al., 2026). Moreover, visual relevance is query-dependent (Zhang et al., 2025): a fleeting subtitle or small object may resolve the query while prominent camera motion is uninformative. Rigidly encoding each frame at uniform resolution forces spatial fidelity and temporal sampling density into a severe trade-off under a constrained visual-token budget. Consequently, an MLLM requires a query-conditioned policy to dynamically allocate spatial fidelity along the timeline.

Existing efficiency methods intervene at different stages of this pipeline (Figure 1a). Post-encoding pruning (Yang et al., 2025a) shortens the sequence only after full-resolution feature extraction. In-encoder merging (Bolya et al., 2023) reduces later-layer compute but still ingests full-resolution pixels. Discrete frame selection (Tang et al., 2025b; Zhang et al., 2025) acts earlier, yet hard keep-or-drop decisions break temporal continuity and omit transient cues. Iterative zooming agents (Yang et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Input-side Adaptation vs. Existing Paradigms (b) ResAdapt achieves competitive performance

Figure 1: Pre-encoding allocation and video-QA performance. (a) Comparison of efficiency paradigms: model-side compression, iterative agentic search, and input-side allocation (ResAdapt). **(b)** Radar comparison across four benchmarks on Qwen2.5-VL and Qwen3-VL with 32-frame inputs.

2025c; Shen et al., 2025) re-inspect candidate regions on demand with multi-turn latency. While static image methods like CARES (Kimhi et al., 2026) predict resolution from supervised sufficiency labels, video lacks frame-level labels and must allocate a shared budget across the timeline. Yet, none of these methods dynamically allocate continuous spatial fidelity in a single pass before visual encoding begins.

We introduce **ResAdapt**, an input-side framework that predicts continuous per-frame scales ($0.2\times$–$1.8\times$) prior to visual feature extraction. Ahead of an unmodified MLLM, a lightweight Allocator pairs coarse SmolVLM preview features (Marafioti et al., 2025) with the user query in a single forward pass. Redundant intervals are coarsened, while answer-critical moments receive denser token capacity above native resolution ($s_t > 1.0$). Adjacent similar frames can take asymmetric scales, retaining sharp detail in one while preserving a compact trace in the other. Resized frames enter the downstream MLLM through its native vision processor, preserving timeline continuity while cutting encoder FLOPs and sequence length without model modifications.

Because frame-level resolution labels are unavailable and vision processors discretize scales onto piecewise-constant token grids, we formulate allocation as a continuous-action contextual bandit. Naive linear cost penalties either collapse allocations toward minimum scale or let resolution drift (Section 4.4). We propose **Cost-Aware Policy Optimization (CAPO)**, which stabilizes learning through asymmetric outcome shaping around an anchor-mixed dynamic cost pivot, rewarding cheap successes and penalizing costly failures. CAPO also incorporates temporal-similarity and concentration regularizers to penalize adjacent redundancy and overly peaked policies.

The resulting policy improves the accuracy–budget frontier without collapsing to uniform downscaling. On Qwen2.5-VL at 128 frames, ResAdapt matches dense accuracy on the six-benchmark mean at 22.9% retention (56.28 vs. 55.85), leads dense input on VideoMME (65.6 vs. 65.3) and VideoMMMU (51.1 vs. 47.9), and recovers 96%–98% at 32 frames (23.8%). Spatial savings can also fund temporal coverage: expanding sampling $16\times$ ($8 \rightarrow 128$ frames) under a matched token budget raises VideoMME by 9.7 points ($+17.9\%$). Permuting scale assignments at comparable retention drops accuracy by 1.3–2.4 points (mean 1.7), confirming the importance of learned fidelity placement along the timeline. The frozen Allocator transfers zero-shot across unseen backbones (Bai et al., 2025a; Kimi Team, 2025; Xiaomi LLM-Core Team, 2025) and cuts 128-frame latency by 59.5% at 28.2% retention ($2.47\times$). Our primary contributions are threefold:

1. **Pre-encoding continuous allocation:** Reframing resolution as an adaptive variable allocated prior to feature extraction preserves temporal continuity while cutting encoder compute and sequence length without modifying the downstream MLLM.
2. **Cost-Aware Policy Optimization (CAPO):** A continuous-action contextual bandit that stabilizes policy learning under non-differentiable token discretization via dynamic-pivot asymmetric shaping and temporal regularization.
3. **Fungibility, efficiency, and generalization:** We demonstrate that spatial fidelity and temporal coverage are interchangeable under fixed budgets, establish Pareto-dominant accuracy across diverse video benchmarks, achieve substantial wall-clock speedups, and show zero-shot policy transfer across unseen backbones.

2 Background and Problem Formulation

2.1 Why Allocate Before Backbone Encoding?

In video MLLMs, computational cost compounds across two distinct stages: visual feature extraction, whose token-wise work scales with input area ($\sim \sum_{t=1}^{T} s_t^2$) while full spatial attention scales even faster (Dehghani et al., 2023; Karamcheti et al., 2024), and language prefill self-attention, which scales quadratically with sequence length $\left(\sum_{t=1}^{T} n_t\right)^2$ (Wang et al., 2024a; Bai et al., 2025b). Where an efficiency method intervenes determines which of these terms it can cut. Post-encoding compression (Chen et al., 2024; Yang et al., 2025a; Tao et al., 2025) prunes tokens only after paying the full extraction bill. In-encoder merging (Bolya et al., 2023) alleviates deeper-layer compute but still processes full-resolution pixels in early layers. Iterative zooming agents (Yang et al., 2025c; Shen et al., 2025; Zou et al., 2025) re-invoke the visual encoder over candidate regions, adding substantial multi-turn latency at inference.

By contrast, pre-encoding allocation resizes frame f_t by a continuous scale s_t before visual extraction, simultaneously reducing pixel-level encoder work and downstream sequence length in a single pass. The same continuous action space also goes beyond subtraction: it coarsens redundant background intervals while selectively allocating denser token capacity to answer-critical frames ($s_t > 1.0$). Resized frames enter the downstream MLLM through its native vision processor, preserving timeline continuity without architectural changes.

2.2 Preliminaries and Budget Accounting

Given a text query $\boldsymbol{q}$ and a sampled video $\mathcal{V} = (f_t)_{t=1}^{T}$, let $\boldsymbol{x} = (\boldsymbol{q}, \mathcal{V})$. The Allocator policy π_θ predicts one continuous scale per frame:

$$\boldsymbol{s} \sim \pi_\theta(\cdot \mid \boldsymbol{x}), \qquad 0 < s_{\min} \le s_t \le s_{\max}. \tag{1}$$

We set $s_{\min} = 0.2$ and $s_{\max} = 1.8 > 1.0$, spanning aggressive downscaling and selective upscaling within one action space. The continuous action s_t is realized by $\tilde{f}_t = \mathrm{Resize}(f_t, s_t)$, forming the adapted input $\tilde{\boldsymbol{x}}(\boldsymbol{s}) = (\boldsymbol{q}, (\tilde{f}_t)_{t=1}^{T})$ and response $\boldsymbol{y} \sim \pi_\phi(\cdot \mid \tilde{\boldsymbol{x}}(\boldsymbol{s}))$. Token-wise encoder work scales with area ($\sim s_t^2$); Appendix C separates this from quadratic spatial attention.

Let $n_t(\boldsymbol{s})$ denote the visual tokens attributed to frame t after processor collation of the full scale vector (Appendix C). We define the sequence-level *visual-token retention ratio* $R(\boldsymbol{s})$ as:

$$R(\boldsymbol{s}) = \frac{\sum_{t=1}^{T} n_t(\boldsymbol{s})}{\sum_{t=1}^{T} n_t(\mathbf{1})}. \tag{2}$$

$R = 1$ denotes the dense full-resolution reference. Retention is sequence-level, so individual frames may take $s_t > 1$ inside a compressed video. Benchmark tables report the mean over evaluation samples of this realized ratio after processor collation; wall-clock latency is measured separately. Appendix C provides the closed-form area approximation used in complexity analysis, and Appendix B derives its relationship to the linear training surrogate c_m.

2.3 Problem Formulation

Because the Allocator selects all frame scales $\boldsymbol{s}$ in one step before backbone encoding, we formulate fidelity allocation as a continuous-action contextual bandit with context $\boldsymbol{x}$ and action $\boldsymbol{s}$.

For inputs from $\mathcal{D}$ and task quality $Q(\boldsymbol{x}, \boldsymbol{y})$, the budgeted objective is

$$\max_{\theta} \; \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \mathbb{E}_{\boldsymbol{s}\sim\pi_\theta(\cdot|\boldsymbol{x})} \mathbb{E}_{\boldsymbol{y}\sim\pi_\phi(\cdot|\tilde{\boldsymbol{x}}(\boldsymbol{s}))} \left[Q(\boldsymbol{x}, \boldsymbol{y}) - \lambda_{\mathrm{bud}} R(\boldsymbol{s}) \right], \tag{3}$$

where $\lambda_{\mathrm{bud}} \ge 0$ balances task performance against visual-token cost. In standard ResAdapt, the downstream backbone ϕ remains frozen while only the Allocator θ is trained; the joint ResAdapt-RL variant optimizes both over $\mathcal{J}(\theta, \phi)$ (Appendix B). However, directly optimizing this objective via standard reinforcement learning is non-trivial: vision processor rounding renders $R(\boldsymbol{s})$ piecewise-constant and non-differentiable, while trajectory-level rewards lack frame-level supervision. Section 3 details how our Allocator architecture and Cost-Aware Policy Optimization (CAPO) resolve these optimization challenges.

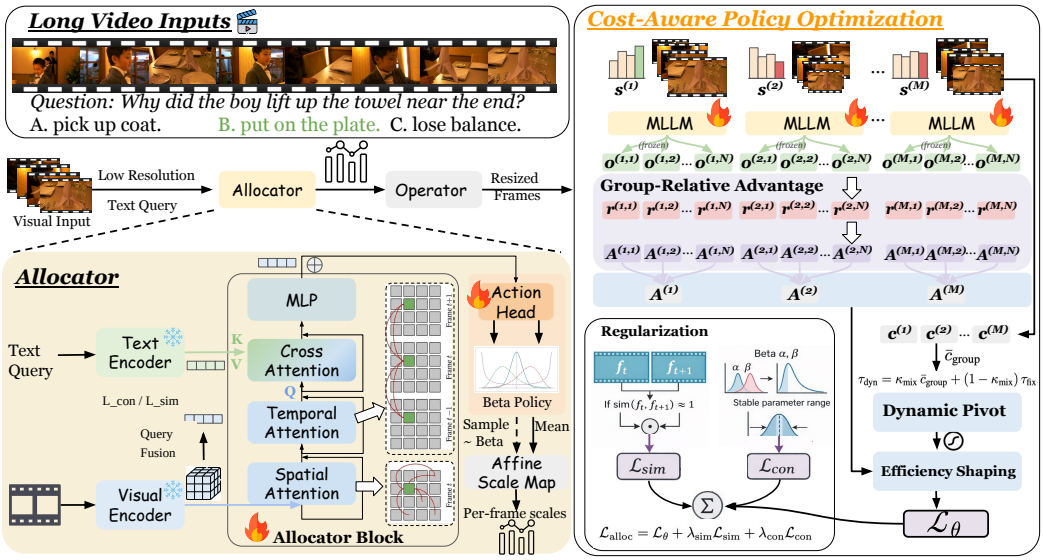


Figure 2: Overview of ResAdapt. Left: A query-conditioned Allocator predicts continuous spatial scales ($0.2\times$ to $1.8\times$) before visual encoding. The adapted sequence enters the downstream MLLM in a single pass; the backbone is frozen in standard ResAdapt and updated only in the optional ResAdapt-RL variant. **Right:** CAPO shapes group-relative task rewards with a dynamic cost pivot and applies $\mathcal{L}_{\text{sim}}$ and $\mathcal{L}_{\text{con}}$ to penalize redundant high-resolution pairs and overly concentrated Beta distributions.

3 METHOD

ResAdapt instantiates the contextual bandit formulation of Section 2.3 as a lightweight policy operating ahead of an unmodified MLLM (Figure 2), trained directly from task feedback via Cost-Aware Policy Optimization (CAPO).

3.1 QUERY-CONDITIONED ALLOCATOR

Ahead of the encoder, the Allocator sees only a coarse preview: frozen SmolVLM features (Marafioti et al., 2025) supply K patch embeddings, projected to width $D = 1{,}024$. Four decoder blocks apply spatial attention within each frame, temporal attention across frames, query cross-attention, and a feed-forward layer (Figure 2). Spatial pooling yields frame states $\boldsymbol{h}_t$ for the policy heads; pooled preview features $\boldsymbol{z}_t$ separately gauge adjacent similarity.

Two projection heads map $\boldsymbol{h}_t$ to Beta parameters $\alpha_t, \beta_t > 1$ via $\text{softplus}(\cdot) + 1$. Conditioned on the full video and query, the policy samples independent latents $a_t \sim \text{Beta}(\alpha_t, \beta_t)$ and maps them to continuous scales:

$$s_t = s_{\min} + a_t(s_{\max} - s_{\min}). \tag{4}$$

The Beta distribution provides continuous support over $(0, 1)$, accommodating bounded scale factors while avoiding boundary truncation. At inference, we use the analytical mean $a_t = \alpha_t/(\alpha_t + \beta_t)$ to predict deterministic scales.

3.2 COST-AWARE POLICY OPTIMIZATION

For each video–query context we sample M allocations $\boldsymbol{s}_m$ with $N{=}1$ rollout each. Given task score $r^{\text{base}}_{m,n}$ and binary success $u_{m,n} \in \{0, 1\}$, group-relative advantage is $A^{\text{base}}_{m,n} = (r^{\text{base}}_{m,n} - \bar{r})/\sigma_r$ when $\sigma_r > 0$ and zero otherwise (Shao et al., 2024), measuring whether a candidate allocation outperformed the other attempts collected for the same prompt.

This signal ignores cost. In our ablations, a direct linear penalty lowers the mean scale, whereas removing explicit cost terms allows resolution to rise (Section 4.4). CAPO conditions cost pressure on outcome, using the normalized mean scale $c_m = (\bar{s}_m - s_{\min})/(s_{\max} - s_{\min})$ with $\bar{s}_m = T^{-1}\sum_t s_{m,t}$. This coordinate distinguishes allocations that processor rounding maps to the same token count,

providing graded cost feedback to the score-function estimator. Appendix E.3.3 defines the controls; Appendix C details token accounting.

Dynamic cost pivot. A fixed reference τ_{fix} supplies an absolute cost anchor. Mixing it with the group mean $\bar{c}_{\text{group}} = M^{-1}\sum_m c_m$ yields $\tau_{\text{dyn}} = \kappa_{\text{mix}}\bar{c}_{\text{group}} + (1-\kappa_{\text{mix}})\tau_{\text{fix}}$, with $\kappa_{\text{mix}} \in [0,1]$, dynamically positioning the center of the shaping sigmoid to balance global target efficiency against empirical batch costs.

Accuracy-conditioned shaping. Outcome-conditioned shaping is formulated as:

$$S_{m,n} = \omega_{m,n}\lambda_+\sigma\left(\frac{\tau_{\text{dyn}} - c_m}{\tau_{\text{s}}}\right) - (1-\omega_{m,n})\lambda_-\sigma\left(\frac{c_m - \tau_{\text{dyn}}}{\tau_{\text{s}}}\right), \tag{5}$$

where $\sigma(z) = (1+e^{-z})^{-1}$, τ_{s} sets transition steepness, and $\omega_{m,n} = u_{m,n} + (1-u_{m,n})\omega_0$ with $0 < \omega_0 \ll 1$. For successful rollouts ($u_{m,n} = 1$), shaping awards an efficiency bonus that scales down as cost increases. For failures ($u_{m,n} = 0$), shaping applies a penalty that escalates as cost rises, while ω_0 preserves a small positive reward branch for low-cost attempts. We set $\lambda_- > \lambda_+ > 0$ so penalties for costly failures exceed bonuses for efficient successes. Task reward, shaping, and global cost combine as

$$\tilde{A}_{m,n} = A^{\text{base}}_{m,n} + \lambda_{\text{capo}} S_{m,n} - \gamma c_m, \tag{6}$$

$$A_{m,n} = u_{m,n}\max(\tilde{A}_{m,n}, \varepsilon_+) + (1-u_{m,n})\tilde{A}_{m,n}. \tag{7}$$

The term $-\gamma c_m$ provides baseline compression pressure, while $\lambda_{\text{capo}} S_{m,n}$ modulates it based on task success. The threshold $\varepsilon_+ > 0$ acts as a protective floor that prevents successful yet cost-heavy rollouts from receiving negative reinforcement during early exploration (Appendix B.4).

3.3 Temporal Structure and Optimization

Because $A_{m,n}$ is a global scalar across frames, we add two inductive biases that guide frame-level allocation and penalize overly concentrated Beta distributions.

For $T \geq 2$, we penalize high fidelity on visually redundant adjacent frames:

$$\mathcal{L}_{\text{sim}} = \frac{1}{T-1}\sum_{t=1}^{T-1} w_t \max(0, \log\mu_t + \log\mu_{t+1} + \eta_{\text{sim}}), \tag{8}$$

where $\mu_t = s_{\min} + p_t(s_{\max} - s_{\min})$ and $p_t = \alpha_t/(\alpha_t+\beta_t)$. The detached gate $w_t = \sigma((\text{sim}_{\cos}(\boldsymbol{z}_t, \boldsymbol{z}_{t+1}) - \tau_{\text{sim}})/\gamma_{\text{sim}})$ weights near-duplicates. The loss is active when $\mu_t\mu_{t+1} > e^{-\eta_{\text{sim}}}$. With $\eta_{\text{sim}} = 1$, two equal scales must lie below ≈ 0.607 to incur zero penalty, so the regularizer also lowers overall budget; task rewards break symmetry between the two frames. Gradients flow through the analytical means μ_t to both heads; $\mathcal{L}_{\text{sim}} = 0$ for $T = 1$.

To discourage peaked policies, we penalize excess concentration:

$$\mathcal{L}_{\text{con}} = \frac{1}{T}\sum_{t=1}^{T}[\alpha_t + \beta_t - \kappa_{\max}]_+, \tag{9}$$

where $[v]_+ = \max(0, v)$. Since $\text{Var}(s_t) = (s_{\max} - s_{\min})^2 p_t(1-p_t)/(\alpha_t+\beta_t+1)$, the hinge penalizes $\alpha_t + \beta_t$ above $\kappa_{\max}$ to discourage overly concentrated distributions (Appendix B.4).

The Allocator objective $\mathcal{L}_{\text{alloc}} = \mathcal{L}_\theta + \lambda_{\text{sim}}\mathcal{L}_{\text{sim}} + \lambda_{\text{con}}\mathcal{L}_{\text{con}}$ optimizes coordinate-wise clipped ratios on latent Beta densities: $\mathcal{L}_\theta = -\frac{1}{MT}\sum_{m,t}\min\big(r^{(m)}_{\theta,t}A_{m,1},\ \text{clip}(r^{(m)}_{\theta,t}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})A_{m,1}\big)$, where $r^{(m)}_{\theta,t} = q_{\theta,t}(a_{m,t} \mid \boldsymbol{x})/q_{\theta_{\text{old}},t}(a_{m,t} \mid \boldsymbol{x})$ is the per-frame density ratio with asymmetric clipping margins $(\varepsilon_{\text{low}}, \varepsilon_{\text{high}}) = (0.2, 0.25)$ (Yu et al., 2025). The constant affine Jacobian $\mathrm{d}s_t/\mathrm{d}a_t = s_{\max} - s_{\min}$ cancels identically from the ratio. For joint training (ResAdapt-RL), Allocator updates alternate with token-level clipped backbone updates (Schulman et al., 2017; Yu et al., 2025) (Appendix B). Hyperparameters and training details are in Appendix D.2.

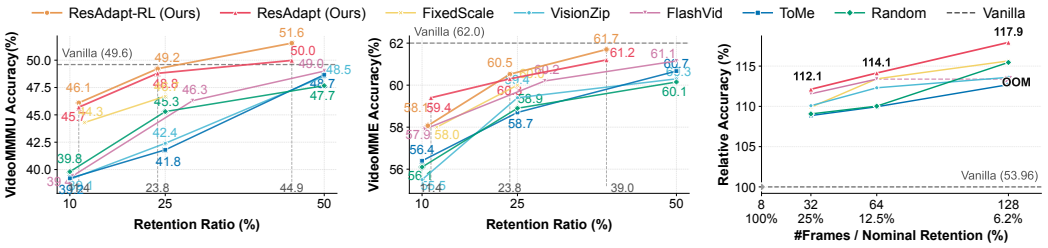


Figure 3: Accuracy–retention frontiers and temporal reallocation. (a,b) Pareto curves on VideoMMMU and VideoMME versus mean visual-token retention R. **(c)** VideoMME accuracy relative to dense 8-frame input as coverage expands from 8 to 128 frames under a matched token budget. Ticks show frame count; FlashVid is OOM at 128 frames.

4 Experiments

4.1 Setup

Implementation and training setup. The Allocator π_θ reads coarse SmolVLM previews (Marafioti et al., 2025) and emits continuous per-frame scales ($0.2\times$–$1.8\times$). Training uses Qwen2.5-VL-7B-Instruct (Bai et al., 2025b) on 93.4K filtered image–video examples from VideoAuto-R1 (Liu et al., 2026) and Video-R1 (Feng et al., 2025). Standard ResAdapt freezes the downstream backbone, while ResAdapt-RL updates it jointly; complete training configurations, hyperparameter choices, and reward specifications are detailed in Appendix D.

Baselines, benchmarks, and evaluation protocol. We evaluate performance on the training backbone and assess zero-shot transfer to Qwen3-VL-8B-Instruct (Bai et al., 2025a), MoE Kimi-VL (Kimi Team, 2025), and RL-tuned MiMo-VL (Xiaomi LLM-Core Team, 2025). Budgets R report mean realized visual-token retention relative to dense input after processor collation; latency includes all front-end overhead. Baselines span Random Drop, FixedScale, ToMe (Bolya et al., 2023), FlashVid (Fan et al., 2026), VisionZip (Yang et al., 2025a), QFrame (Zhang et al., 2025), VideoTree (Wang et al., 2024b), and SlowFast (Xu et al., 2024). Video-QA evaluates six benchmarks (VideoMME, MMVU, LongVideoBench, MLVU, VideoMMMU, LVBench) via unweighted mean; temporal grounding evaluates Charades-STA, ActivityNet, and NExT-GQA (Appendix Table 7).

4.2 Main Results

Pareto efficiency across video benchmarks. Pre-encoding allocation shifts the accuracy–budget frontier well beyond uniform downscaling. On Qwen2.5-VL at 128 frames and 22.9% retention, ResAdapt achieves a six-benchmark mean of 56.28 vs. 55.85 dense, matching or exceeding dense accuracy on VideoMME (65.6 vs. 65.3) and VideoMMMU (51.1 vs. 47.9) while trailing slightly on three perception tasks (Table 1). ResAdapt attains the top compressed score in seven of eight backbone–horizon–budget blocks (Appendix Table 6), leading FlashVid by 2.5–8.3 points on VideoMMMU at comparable budgets (Figure 3). This demonstrates that continuous allocation effectively preserves complex reasoning under tight token budgets.

Fungibility between spatial fidelity and temporal coverage. Spatial savings can be directly reinvested into denser temporal sampling of the same video. Figure 3c fixes the visual budget at an 8-frame-equivalent token limit while expanding sampling from 8 to 128 frames ($16\times$ density). Accuracy rises monotonically, reaching 63.64 on VideoMME at 128 frames, a 9.7-point absolute gain over dense 8-frame input (53.96, $+17.9\%$ relative) that also surpasses dense 32-frame input (62.0). At this operating point, denser temporal sampling improves VideoMME accuracy within the fixed visual-token budget.

Continuous spatial scaling vs. discrete frame selection. Hard frame selection drops entire intervals, whereas continuous spatial scaling degrades gracefully and preserves a visual trace across the timeline. Under 128-frame inputs at a 32-frame-equivalent budget (Table 2), ResAdapt outperforms QFrame by 11.5 points and SlowFast by 8.7 points on VideoMMMU, while also leading on

Table 1: Video-QA results across two backbones and 32/128-frame inputs. R is mean per-sample realized visual-token retention, averaged over tasks in each row. *CoT* marks chain-of-thought inference. **Bold**: best compressed result without CoT in each budget block. Grounding results appear in Table 7.

Backbone	Method	Retention Ratio R	CoT	Video Perception				Video Reasoning	
				VideoMME	LongVideoBench	MMVU	MLVU	VideoMMMU	LVBench
	32 Frames								
Qwen2.5-VL-7B	Vanilla	100%	✗	62.0	58.9	52.7	63.1	49.6	38.6
	Random Drop	25.0%	✗	58.9	57.8	49.6	58.3	45.3	36.7
	ToMe	25.0%	✗	58.7	58.0	51.0	58.7	41.8	37.7
	VisionZip	25.0%	✗	59.4	57.1	49.8	57.9	42.4	36.5
	FlashVid	29.3%	✗	60.2	**58.6**	51.1	59.2	46.3	36.9
	FixedScale	25.0%	✗	60.0	56.8	51.2	59.8	46.7	37.3
	ResAdapt (Ours)	23.8%	✗	**60.3**	58.2	**51.9**	**60.1**	**48.8**	**37.9**
	Random Drop	10.0%	✗	56.1	55.6	47.1	56.5	39.8	35.2
	ToMe	10.0%	✗	56.4	55.2	48.9	58.0	39.2	33.6
	VisionZip	10.0%	✗	55.5	54.5	47.6	57.3	39.1	35.3
	FlashVid	10.4%	✗	57.9	**56.8**	47.9	57.7	39.4	**36.5**
	FixedScale	12.3%	✗	58.0	55.1	47.7	57.5	44.3	35.4
	ResAdapt (Ours)	11.4%	✗	**59.4**	55.4	**49.2**	**58.4**	**45.7**	35.9
	VideoAuto-R1	100%	✓	63.2	58.9	55.0	60.0	53.6	41.5
	+ **ResAdapt (Ours)**	23.8%	✓	60.4	57.1	53.2	61.1	51.2	38.7
	+ **ResAdapt (Ours)**	11.4%	✓	59.3	56.3	51.8	59.3	49.1	36.7
	128 Frames								
	Vanilla	100%	✗	65.3	60.3	53.1	66.5	47.9	42.0
	Random Drop	25.0%	✗	64.9	61.2	50.8	64.8	48.1	41.3
	ToMe	25.0%	✗	65.1	**61.6**	51.9	63.1	46.6	**42.1**
	VisionZip	25.0%	✗	64.8	61.3	51.1	64.5	47.3	41.5
	ResAdapt (Ours)	22.9%	✗	**65.6**	60.2	**52.8**	**65.9**	**51.1**	**42.1**
	Random Drop	10.0%	✗	63.0	59.0	45.8	63.4	46.7	38.0
	ToMe	10.0%	✗	60.6	56.3	44.2	63.5	41.8	39.5
	VisionZip	10.0%	✗	61.8	56.1	44.8	63.2	42.1	38.2
	FixedScale	12.3%	✗	**64.1**	**60.9**	**49.6**	**64.5**	46.9	**40.3**
	ResAdapt (Ours)	11.1%	✗	63.8	58.6	49.0	64.3	**49.2**	39.9
	VideoAuto-R1	100%	✓	64.7	59.1	56.7	65.1	52.2	41.2
	+ **ResAdapt (Ours)**	22.9%	✓	66.2	60.2	53.5	66.0	52.6	41.8
	+ **ResAdapt (Ours)**	11.1%	✓	64.7	57.8	52.4	64.6	51.3	39.5
	32 Frames								
Qwen3-VL-8B	Vanilla	100%	✗	65.0	58.6	57.5	64.0	60.8	40.2
	Random Drop	25.0%	✗	61.3	58.4	**57.1**	60.2	53.4	37.8
	ToMe	25.0%	✗	62.4	57.4	56.0	60.8	49.1	36.4
	VisionZip	25.0%	✗	61.8	57.2	54.4	60.6	51.5	37.3
	FlashVid	30.0%	✗	**63.9**	**59.0**	54.8	**61.9**	55.1	**38.5**
	ResAdapt (Ours)	23.8%	✗	62.6	57.5	55.3	61.0	**58.4**	**38.5**
	Random Drop	10.0%	✗	58.8	54.7	53.2	56.6	47.1	35.5
	ToMe	10.0%	✗	59.2	55.5	53.1	58.5	42.7	35.8
	VisionZip	10.0%	✗	59.9	55.4	53.7	58.8	45.8	35.4
	FlashVid	12.2%	✗	**61.0**	**57.1**	**54.8**	59.1	47.8	37.1
	FixedScale	12.3%	✗	60.8	54.9	53.8	58.4	52.6	37.1
	ResAdapt (Ours)	11.4%	✗	60.7	56.6	54.6	**59.6**	**56.1**	**37.3**
	128 Frames								
	Vanilla	100%	✗	69.4	64.3	58.5	72.7	63.0	45.7
	Random Drop	25.0%	✗	67.2	61.3	**56.8**	67.4	55.3	42.4
	ToMe	25.0%	✗	67.2	**62.0**	55.9	70.4	53.5	43.1
	VisionZip	25.0%	✗	67.1	61.3	55.7	69.2	56.8	41.2
	ResAdapt (Ours)	22.9%	✗	**67.4**	61.9	56.3	**70.8**	**59.6**	**43.3**
	Random Drop	10.0%	✗	64.1	58.3	**55.4**	62.4	55.5	38.8
	ToMe	10.0%	✗	64.7	58.6	55.1	67.3	46.3	40.5
	VisionZip	10.0%	✗	64.2	59.1	54.2	66.8	47.6	39.4
	FixedScale	12.3%	✗	66.7	59.5	54.4	67.7	56.3	41.7
	ResAdapt (Ours)	11.1%	✗	**66.8**	**60.2**	**55.4**	**69.4**	**58.2**	**42.6**

VideoMME and MMVU. The largest margin is on VideoMMMU, supporting the value of retaining more temporal evidence for multi-frame reasoning.

Zero-shot generalization across MLLM backbones. To evaluate architecture portability, we test the frozen Qwen2.5-VL policy on unseen model families without retraining (Table 1, Appendix Table 16). At 32 frames and $\sim$23% retention, ResAdapt recovers 96.3% of the dense mean on Qwen3-VL and 98.2% on Kimi-VL and MiMo-VL. At 128 frames, it exceeds the dense four-benchmark

Table 2: Adaptive video representations on Qwen2.5-VL-7B. 128-frame inputs at a 32-frame-equivalent visual budget. **Bold**: best compressed result in each column.

Method	LongVideoBench	LVBench	VideoMMMU	VideoMME	MMVU
Vanilla (128f, 100%)	60.3	42.0	47.9	65.3	53.1
QFrame (Zhang et al., 2025) (128→32)	**61.5**	**43.6**	39.6	63.2	50.3
VideoTree (Wang et al., 2024b) (128→32)	60.6	42.0	38.9	62.4	49.8
SlowFast (Xu et al., 2024) (32-frame budget)	58.5	37.6	42.4	59.8	50.4
ResAdapt (Ours), 128f @ 22.9%	60.2	42.1	**51.1**	**65.6**	**52.8**

Table 3: Latency (ms, ↓) on Qwen2.5-VL-7B. The Allocator uses one H100; vLLM 0.8.1 uses four, with bf16, batch size 1, and at most 256 output tokens. Values average 200 runs after 5 warm-ups. *Backbone Time* includes prefill and decode. E2E = Scale Time + Backbone Time.

#Frames	Method	Retention Ratio R	Scale						Inference		Total
			TFLOPs	Text Enc.	Visual Enc.	Scale Pred.	Scale Apply	Scale Time	TFLOPs	Backbone Time	E2E Time
32	Vanilla	100%	–	–	–	–	–	–	222.5	881.9	881.9
	ResAdapt	74.4%	2.9	19.9	204.1	97.4	14.4	335.9	153.9 (↓30.8%)	627.6 (↓28.8%)	963.5 (↑9.2%)
	ResAdapt	51.5%	2.9	20.0	193.2	92.0	16.2	321.4	102.4 (↓54.0%)	467.1 (↓47.0%)	788.5 (↓10.6%)
	ResAdapt	28.2%	2.9	20.3	190.4	90.3	17.3	318.3	61.4 (↓72.4%)	332.6 (↓62.3%)	650.9 (↓26.2%)
128	Vanilla	100%	–	–	–	–	–	–	888.9	4877.0	4877.0
	ResAdapt	74.2%	11.6	20.1	766.3	95.0	53.1	934.5	614.1 (↓30.9%)	2323.6 (↓52.4%)	3258.1 (↓33.2%)
	ResAdapt	51.4%	11.6	20.2	755.3	93.8	59.4	928.7	408.0 (↓54.1%)	1496.0 (↓69.3%)	2424.7 (↓50.3%)
	ResAdapt	28.2%	11.6	20.4	734.5	92.0	68.6	915.5	243.9 (↓72.6%)	1061.8 (↓78.2%)	1977.3 (↓59.5%)

mean by $+0.79$ points on Kimi-VL and MiMo-VL. These results show that the learned allocation policy transfers across the evaluated architectures.

Fidelity allocation under temporal grounding. Temporal grounding tests whether spatial coarsening preserves fine-grained boundary timestamps. At 128 frames and 16.1% retention, ResAdapt attains 42.0 mIoU on Charades-STA, doubling Random Drop and ToMe at 25.0% budget (Appendix Table 7). Paired with reasoning models like VideoAuto-R1, ResAdapt surpasses dense baselines on ActivityNet (46.1) and NExT-GQA (37.3) at one-sixth of the tokens, confirming that semantic coarsening preserves fine temporal boundaries across downstream tasks.

4.3 End-to-End Latency

Table 3 profiles batch-1 end-to-end latency on Qwen2.5-VL-7B using vLLM on NVIDIA H100 GPUs across horizons and retentions. The Allocator requires only 2.9 TFLOPs at 32 frames (1.3% of the backbone's 222.5 TFLOPs), with scale prediction taking ≈ 90 ms across horizons. At 128 frames ($R \approx 28.2\%$), ResAdapt cuts end-to-end latency by 59.5% ($4{,}877.0 \rightarrow 1{,}977.3$ ms, a $2.47\times$ speedup): the 78.2% backbone inference saving ($4{,}877.0 \rightarrow 1{,}061.8$ ms) decisively outweighs the 915.5 ms front-end overhead. Front-end allocation amortizes on extended timelines and deep compression, whereas on short 16-frame clips with conservative retention, overhead can exceed modest backbone savings (Table 10).

4.4 Analysis and Ablation

Contribution of fidelity placement. Permuting learned frame-scale assignments within each video while preserving the scale multiset (at matched mean retention $R = 23.8\%$) (Krojer et al., 2025) drops accuracy by 1.3–2.4 points across benchmarks (mean 1.7; Appendix Table 20), supporting the contribution of learned resolution placement across the video timeline.

Query conditioning vs. content-only allocation. We examine whether allocation requires language guidance. Pairing test videos with mismatched queries drops six-benchmark accuracy by 1.10 points (VideoMMMU by 2.4), while training without language conditioning drops accuracy by 0.44 points despite using more tokens (25.4% vs. 23.8%; Appendix Table 12). These ablations support the contribution of query conditioning to allocation quality.

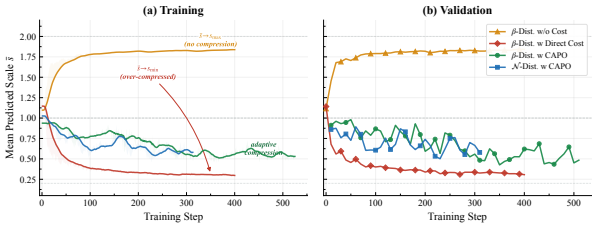


Figure 4: Effect of cost shaping. Train/val mean predicted scale $\bar{s}$. CAPO stays interior; no-cost and direct-cost curves drift. Test retention R is in Table 4.

Table 4: CAPO objectives. Qwen2.5-VL-7B, 32 frames. Native and calibrated retention and six-benchmark means; calibration targets $R{\approx}23.8\%$.

Objective	Native		Calibrated	
	R (%)	Mean	R (%)	Mean
Full CAPO	23.8	52.87	23.8	**52.87**
w/o $\mathcal{L}_{\text{sim}}$	22.9	52.07	23.7	52.23
Direct cost	9.4	49.38	22.8	51.57
No cost	67.2	53.60	23.5	52.02

Ablation of cost-aware policy optimization. CAPO achieves higher mean accuracy than the tested direct-cost and no-cost objectives after validation-based budget calibration (Table 4, Figure 4). At native operating points, removing explicit cost terms (while retaining $\mathcal{L}_{\text{sim}}$) lets retention drift to 67.2% for a marginal 0.73-point gain, whereas a direct linear penalty collapses retention to 9.4% with a 3.49-point loss. Full CAPO stably anchors an intermediate regime (23.8% retention, 52.87 mean). When calibrated toward comparable budgets ($R \approx 23.8\%$), CAPO leads the no-cost and direct-cost variants by 0.85 and 1.30 points, while removing $\mathcal{L}_{\text{sim}}$ incurs a 0.64-point drop (Appendix Table 14). Allocation profiles separately confirm the regularizer's role in promoting scale diversity (Appendix Figures 11–13).

5 Related Work

Adaptive video representations. Prior input-side methods reduce frames via discrete selection (Wu et al., 2019; Meng et al., 2020; Tang et al., 2025b; Chen et al., 2026), tiered presets (Zhang et al., 2025; Kim et al., 2026; An & Grauman, 2026), or encoder redundancy reduction (Tang et al., 2026; An et al., 2026; Yang et al., 2026). However, hard frame dropping disrupts narrative continuity, while discrete tiers offer limited flexibility. In contrast, ResAdapt preserves timeline continuity by predicting continuous per-frame scales ($s_t \in [0.2, 1.8]$) in a single feed-forward pass.

In-encoder and post-encoding compression. Post-encoding token pruning (Chen et al., 2024; Yang et al., 2025a; Fan et al., 2026; Xing et al., 2024; Xu et al., 2024) and dynamic sequence routing (Li et al., 2025; Qi et al., 2026; Khaki et al., 2025; Tao et al., 2025) shorten sequences but pay full feature extraction costs. In-encoder merging (Bolya et al., 2023) processes full-resolution pixels in early layers, while iterative zooming agents (Yang et al., 2025c; Shen et al., 2025) incur multi-turn latency. Pre-encoding allocation intervenes at the source, simultaneously reducing visual encoder FLOPs and downstream language prefill length.

Policy optimization for visual efficiency. Group-relative policy optimization (Shao et al., 2024; Guo et al., 2025) has advanced multimodal reasoning (Feng et al., 2025; Liu et al., 2026) and discrete efficiency choices such as temporal sampling (Tang et al., 2025a), token budgeting (Yang et al., 2025b), and memory compression (Wu et al., 2025). Continuous pre-encoding allocation presents unique challenges under non-differentiable processor grids, which CAPO resolves via dynamic-pivot asymmetric shaping and temporal regularization.

6 Conclusion

ResAdapt reframes spatial resolution from a static preset into an adaptive variable allocated along the video timeline prior to visual encoding. We formulate frame-fidelity allocation as a contextual bandit and train the Allocator with CAPO directly from task feedback, without requiring frame-level supervision or architectural modifications. Spatial fidelity and temporal sampling density act as interchangeable resources under fixed budgets: tokens spared on redundant intervals fund $16\times$ denser temporal sampling, unlocking a $+9.7$-point VideoMME gain under matched budgets. Permutation controls indicate that deliberate resolution placement across frames contributes to performance beyond aggregate retention. Furthermore, the Allocator cuts 128-frame latency by 59.5% at 28.2% retention on Qwen2.5-VL and transfers zero-shot across diverse backbones, establishing spatial allocation as an adaptive degree of freedom for long-horizon video reasoning.

AI USE STATEMENT

Generative AI tools assisted with selected coding tasks, methodological review, and manuscript polishing. The authors reviewed all AI-assisted material and remain responsible for the paper and code.

ETHICS STATEMENT

This study involved no human annotation or new human-subject data collection. All datasets are publicly available and were used under their applicable open-source licenses. The implementation combines open-source frameworks with code developed by the authors.

REPRODUCIBILITY STATEMENT

Appendix D documents data construction, training configuration, prompts, reward definitions, and evaluation protocols. Benchmark tables report mean per-sample realized visual-token retention R after processor collation; latency tables report measured wall-clock time. Appendix C states the area approximation used in complexity analysis, while Appendix B derives its relationship to the linear training surrogate c_m.

REFERENCES

Joungbin An and Kristen Grauman. AdaAlloc: Adaptive visual token allocation for long-video question answering. Project page, 2026. URL https://vision.cs.utexas.edu/projects/adaalloc/.

Xiang An, Yin Xie, Feilong Tang, Yunyao Yan, Huajie Tan, Didi Zhu, Changrui Chen, Xiuwei Zhao, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Kaichen Zhang, Wenkang Zhang, Zheng Cheng, Nansen Zhang, Chunsheng Wu, Chunjiang Ge, Zimin Ran, Dehua Song, Chunyuan Li, Shikun Feng, Ming Hu, Zhangquan Chen, Junbo Niu, Bo Li, Ziyong Feng, Ziwei Liu, Zongyuan Ge, and Jiankang Deng. Llava-onevision-2: Towards next-generation perceptual intelligence, 2026. URL https://arxiv.org/abs/2605.25979.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2024.

Zeyuan Chen, Kai Zhang, Zhuowen Tu, and Yuanjun Xiong. Soft tail-dropping for adaptive visual tokenization. *arXiv preprint arXiv:2601.14246*, 2026.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8016–8026, 2023.

Ziyang Fan, Keyu Chen, Ruilong Xing, Yulin Li, Li Jiang, and Zhuotao Tian. Flashvid: Efficient video large language models via training-free tree-based spatiotemporal token merging. *arXiv preprint arXiv:2602.08024*, 2026.

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*, 2025.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 24108–24118, 2025.

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

Kairui Hu, Penghao Wu, Fanyi Pu, Wang Xiao, Yuanhan Zhang, Xiang Yue, Bo Li, and Ziwei Liu. Video-mmmu: Evaluating knowledge acquisition from multi-discipline professional videos. *arXiv preprint arXiv:2501.13826*, 2025.

Siddharth Karamcheti, Suraj Nair, Ashwin Chen, Ilya Kostrikov, Chelsea Finn, Percy Liang, and Tete Xiao. Prismatic vlms: Investigating the design space of modularly trained vision-language models. *arXiv preprint arXiv:2402.07865*, 2024.

Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N Plataniotis, Yao Lu, Song Han, and Zhijian Liu. Sparsevila: Decoupling visual sparsity for efficient vlm inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23784–23794, 2025.

Minkuk Kim, Suyong Yun, Young Tae Kim, Jinyoung Moon, Jinwoo Choi, and Seong Tae Kim. ReQuest: Rethinking-based question-aware frame selection for long-form video QA, 2026. URL https://arxiv.org/abs/2607.01737.

Moshe Kimhi, Nimrod Shabtay, Raja Giryes, Chaim Baskin, and Eli Schwartz. CARES: Context-aware resolution selector for VLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2243–2256. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.102. URL https://aclanthology.org/2026.acl-long.102/.

Kimi Team. Kimi-VL technical report, 2025. URL https://arxiv.org/abs/2504.07491.

Benno Krojer, Mojtaba Komeili, Candace Ross, Quentin Garrido, Koustuv Sinha, Nicolas Ballas, and Mahmoud Assran. A shortcut-aware video-qa benchmark for physical understanding via minimal video pairs. *arXiv preprint arXiv:2506.09987*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Yulin Li, Haokun Gui, Ziyang Fan, Junjie Wang, Bin Kang, Bin Chen, and Zhuotao Tian. Less is more, but where? dynamic token compression via llm-guided keyframe prior. *arXiv preprint arXiv:2512.06866*, 2025.

Huanxuan Liao, Shizhu He, Yupu Hao, Xiang Li, Yuanzhe Zhang, Jun Zhao, and Kang Liu. SKIntern: Internalizing symbolic knowledge for distilling better CoT capabilities into small language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 3203–3221. Association for Computational Linguistics, 2025a. URL https://aclanthology.org/2025.coling-main.215/.

Huanxuan Liao, Wen Hu, Yao Xu, Shizhu He, Jun Zhao, and Kang Liu. Beyond hard and soft: Hybrid context compression for balancing local and global information retention. *arXiv preprint arXiv:2505.15774*, 2025b. URL https://arxiv.org/abs/2505.15774.

Huanxuan Liao, Yixing Xu, Shizhu He, Guanchen Li, Xuanwu Yin, Dong Li, Emad Barsoum, Jun Zhao, and Kang Liu. Spark: Query-aware unstructured sparsity with recoverable kv cache channel pruning. *arXiv preprint arXiv:2508.15212*, 2025c. URL https://arxiv.org/abs/2508.15212.

Shuming Liu, Mingchen Zhuge, Changsheng Zhao, Jun Chen, Lemeng Wu, Zechun Liu, Chenchen Zhu, Zhipeng Cai, Chong Zhou, Haozhe Liu, Ernie Chang, Saksham Suri, Hongyu Xu, Qi Qian, Wei Wen, Balakrishnan Varadarajan, Zhuang Liu, Hu Xu, Florian Bordes, Raghuraman Krishnamoorthi, Bernard Ghanem, Vikas Chandra, and Yunyang Xiong. Videoauto-r1: Video auto reasoning via thinking once, answering twice. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 32829–32839, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.

Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and Rogerio Feris. AR-Net: Adaptive frame resolution for efficient action recognition. In *Computer Vision – ECCV 2020*, pp. 86–104, 2020. doi: 10.1007/978-3-030-58571-6_6. URL https://arxiv.org/abs/2007.15796.

Haozhe Qi, Kevin Qu, Mahdi Rad, Rui Wang, Alexander Mathis, and Marc Pollefeys. AdaptToken: Entropy-based adaptive token selection for MLLM long video understanding, 2026. URL https://arxiv.org/abs/2603.28696.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3505–3506, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

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

Xiaoqian Shen, Min-Hung Chen, Yu-Chiang Frank Wang, Mohamed Elhoseiny, and Ryo Hachiuma. Zoom-zero: Reinforced coarse-to-fine video understanding via temporal zoom-in. *arXiv preprint arXiv:2512.14273*, 2025.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

Canhui Tang, Zifan Han, Hongbo Sun, Sanping Zhou, Xuchong Zhang, Xin Wei, Ye Yuan, Huayu Zhang, Jinglin Xu, and Hao Sun. Tspo: Temporal sampling policy optimization for long-form video language understanding. *arXiv preprint arXiv:2508.04369*, 2025a.

Feilong Tang, Xiang An, Yunyao Yan, Yin Xie, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Chunyuan Li, Shikun Feng, Changrui Chen, Huajie Tan, Ming Hu, Manyuan Zhang, Bo Li, Ziyong Feng, Ziwei Liu, Zongyuan Ge, and Jiankang Deng. Onevision-encoder: Codec-aligned sparsity as a foundational principle for multimodal intelligence, 2026. URL https://arxiv.org/abs/2602.08683.

Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. *arXiv preprint arXiv:2502.21271*, 2025b.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. Dycoke: Dynamic compression of tokens for fast video large language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 18992–19001, 2025.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Ming Ding, Xiaotao Gu, Shiyu Huang, Bin Xu, et al. Lvbench: An extreme long video understanding benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22958–22967, 2025.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. *arXiv preprint arXiv:2405.19209*, 2024b.

Peiran Wu, Zhuorui Yu, Yunze Liu, Chi-Hao Wu, Enmin Zhou, and Junxiao Shen. MARC: Memory-augmented RL token compression for efficient video understanding, 2025. URL https://arxiv.org/abs/2510.07915.

Zuxuan Wu, Caiming Xiong, Chih-Yao Ma, Richard Socher, and Larry S. Davis. AdaFrame: Adaptive frame selection for fast video recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1278–1287, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Wu_AdaFrame_Adaptive_Frame_Selection_for_Fast_Video_Recognition_CVPR_2019_paper.html.

Xiaomi LLM-Core Team. MiMo-VL technical report, 2025. URL https://arxiv.org/abs/2506.03569.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024.

Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*, 2024.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19792–19802, 2025a.

Senqiao Yang, Junyi Li, Xin Lai, Jinming Wu, Wei Li, Zejun Ma, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Visionthink: Smart and efficient vision language model via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. doi: 10.52202/085713-3182.

Senqiao Yang, Kaichen Zhang, Zhaoyang Jia, Jinghao Guo, Yifei Shen, Xinjie Zhang, Xiaoyi Zhang, Haoqing Wang, Xiao Li, Peng Zhang, Xiang An, Yin Xie, Zhening Liu, Xun Guo, Jiahao Li, Shicheng Zheng, Jinglu Wang, Zongyu Guo, Wenxuan Xie, Zihan Zheng, Yuxuan Luo, Bin Li, and Yan Lu. Mage-vl: An efficient codec-native streaming multimodal foundation model, 2026. URL https://arxiv.org/abs/2607.24904.

Zuhao Yang, Sudong Wang, Kaichen Zhang, Keming Wu, Sicong Leng, Yifan Zhang, Bo Li, Chengwei Qin, Shijian Lu, Xingxuan Li, and Lidong Bing. Longvt: Incentivizing "thinking with long videos" via native tool calling. *arXiv preprint arXiv:2511.20785*, 2025c.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Kaichen Zhang, Bo Li, Peiyuan Zhang, Fanyi Pu, Joshua Adrian Cahyono, Kairui Hu, Shuai Liu, Yuanhan Zhang, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Lmms-eval: Reality check on the evaluation of large multimodal models, 2024a. URL https://arxiv.org/abs/2407.12772.

Shaojie Zhang, Jiahui Yang, Jianqin Yin, Zhenbo Luo, and Jian Luan. Q-frame: Query-aware frame selection and multi-resolution adaptation for video-llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22056–22065, 2025.

Tianyu Zhang, Suyuchen Wang, Lu Li, Ge Zhang, Perouz Taslakian, Sai Rajeswar, Jie Fu, Bang Liu, and Yoshua Bengio. Vcr: A task for pixel-level complex reasoning in vision language models via restoring occluded text. *arXiv preprint arXiv:2406.06462*, 2024b. URL https://arxiv.org/abs/2406.06462.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

Yuanhao Zou, Shengji Jin, Andong Deng, Youpeng Zhao, Jun Wang, and Chen Chen. A.I.R.: Enabling adaptive, iterative, and reasoning-based frame selection for video question answering. *arXiv preprint arXiv:2510.04428*, 2025.

A Overview and Organization of Appendix

This supplementary document provides mathematical derivations, complexity analyses, implementation details, extended experimental evaluations, qualitative case studies, and operating boundaries for ResAdapt. It is organized into the following sections:

- **Appendix B (Derivation of Joint RL Formulation):** Formulates the allocator-level contextual bandit problem, derives the score-function gradients for joint policy optimization, and details the clipped surrogate objectives.
- **Appendix C (Complexity Analysis):** Analyzes visual-token counts, quadratic attention scaling, Allocator decoder overhead, and architecture-agnostic deployment properties.
- **Appendix D (Implementation Details):** Details training data composition, hyperparameter configurations, reward designs, prompt templates, and evaluation protocols.
- **Appendix E (Supplementary Experiments and Analysis):** Presents full benchmark tables, temporal grounding, static image transfer, extended latency breakdowns, behavioral allocation distributions, and ablation studies.
- **Appendix F (Qualitative Case Studies and Visualizations):** Visualizes representative successes on complex video reasoning tasks and analyzes fine-detail failure cases.
- **Appendix G (Operating Boundaries, Limitations, and Future Work):** Discusses runtime operating regimes, perceptual resolution limits, and avenues for closed-loop extensions.

B Derivation of Joint RL Formulation

This appendix connects the allocator-level contextual bandit of Section 2.3 to the clipped surrogates we actually optimize, and marks where each step is exact and where it is an approximation. Everything is derived for one video–query context; the training objective averages over $\mathcal{D}$.

Notation. The context is $\boldsymbol{x} = (\boldsymbol{q}, \mathcal{V})$, where $\mathcal{V} = (f_t)_{t=1}^{T}$ is an ordered frame sequence. The Allocator policy $\pi_\theta(\boldsymbol{s} \mid \boldsymbol{x})$ emits a continuous allocation vector $\boldsymbol{s} = (s_1, \ldots, s_T)$, which the deterministic map $\tilde{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{s}) = (\boldsymbol{q}, (\mathrm{Resize}(f_t, s_t))_{t=1}^{T})$ applies to the frames; Resize is bilinear resizing in all experiments. The MLLM policy $\pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})$ then samples $\boldsymbol{y} = (\boldsymbol{r}, \boldsymbol{o})$, a reasoning trace followed by an answer.

B.1 Contextual Bandit and the Joint Objective

At the Allocator level the problem is a contextual bandit with a single vector-valued action $\boldsymbol{s}$, and the autoregressive response is the stochastic downstream outcome of that action. Temporal state transitions therefore never enter the Allocator gradient, while the backbone keeps its token-level factorization. The two policies compose conditionally,

$$p_{\theta,\phi}(\boldsymbol{s}, \boldsymbol{y} \mid \boldsymbol{x}) = \pi_\theta(\boldsymbol{s} \mid \boldsymbol{x})\, \pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{s})), \tag{10}$$

so for a context with ground-truth answer $\boldsymbol{o}^\star$ the marginal answer probability under the transformed input is

$$p_{\theta,\phi}(\boldsymbol{o}^\star \mid \boldsymbol{x}) = \mathbb{E}_{\pi_\theta(\boldsymbol{s}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi(\boldsymbol{r}|\tilde{\boldsymbol{x}})}[\pi_\phi(\boldsymbol{o}^\star \mid \tilde{\boldsymbol{x}}, \boldsymbol{r})]\right]. \tag{11}$$

We optimize expected task utility rather than answer log-likelihood. For fixed $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{y})$, the task score $Q(\boldsymbol{x}, \boldsymbol{y})$ and processor cost have no explicit dependence on policy parameters. Charging the allocation its budget cost gives the ideal rollout reward

$$r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}} = Q(\boldsymbol{x}, \boldsymbol{y}) - \lambda_{\text{bud}}\, R(\boldsymbol{s}), \tag{12}$$

with budget-penalty coefficient $\lambda_{\text{bud}} \geq 0$, and the one-step expected return we maximize is

$$\max_{\theta,\phi}\ \mathcal{J}(\theta, \phi) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\pi_\theta(\boldsymbol{s}|\boldsymbol{x})}\left[\mathbb{E}_{\pi_\phi(\boldsymbol{y}|\tilde{\boldsymbol{x}})}\left[r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}}\right]\right]. \tag{13}$$

B.2 Score-Function Gradients and Alternating Updates

Both policies enter Eq. (13) only through sampling, so both gradients are score-function estimators taken over the joint sampling process:

$$\nabla_\phi \mathcal{J}(\theta,\phi) = \mathbb{E}\big[r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}} \nabla_\phi \log \pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{s}))\big], \tag{14}$$

$$\nabla_\theta \mathcal{J}(\theta,\phi) = \mathbb{E}\big[r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}} \nabla_\theta \log \pi_\theta(\boldsymbol{s} \mid \boldsymbol{x})\big]. \tag{15}$$

These identities assume fixed scale bounds, differentiable normalized policy densities, integrable rewards and score functions, and interchange of differentiation and integration. The expectations include $\boldsymbol{x} \sim \mathcal{D}$ and both policies. For fixed $\boldsymbol{x}, \boldsymbol{s}$, the input cost is independent of the generated response, so

$$\mathbb{E}_{\boldsymbol{y}\sim\pi_\phi}[R(\boldsymbol{s})\nabla_\phi \log \pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})] = R(\boldsymbol{s})\nabla_\phi \sum_{\boldsymbol{y}} \pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}) = 0. \tag{16}$$

Thus the exact backbone gradient can use Q alone. An allocation-independent baseline $b(\boldsymbol{x})$ can similarly be subtracted in Eq. (15); a response-independent baseline $b(\boldsymbol{x},\boldsymbol{s})$ can be subtracted in Eq. (14). Group normalization, outcome shaping, and clipping define the practical surrogate below and need not preserve these exact-gradient identities.

For behavior policies $\pi_{\theta_{\text{old}}}, \pi_{\phi_{\text{old}}}$ with support covering the target policies, exact off-policy correction uses the joint weight $r_\theta r_\phi = (\pi_\theta/\pi_{\theta_{\text{old}}})(\pi_\phi/\pi_{\phi_{\text{old}}})$. Both ratios are products over their respective coordinates. Alternating block updates avoid multiplying the two policy ratios: the inactive block remains at the policy that generated the current data. After a block changes, the next block uses fresh rollouts under that updated policy; reusing older data would require the omitted ratio as well.

Updating the backbone fixes $\pi_\theta = \pi_{\theta_{\text{old}}}$ and leaves the off-policy surrogate gradient

$$\nabla_\phi \mathcal{J}_{\text{surr}}(\phi) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\pi_{\theta_{\text{old}}}}\mathbb{E}_{\pi_{\phi_{\text{old}}}}\left[\frac{\pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})}{\pi_{\phi_{\text{old}}}(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})} r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}} \nabla_\phi \log \pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})\right]. \tag{17}$$

Substituting $\nabla_\phi r_\phi = r_\phi \nabla_\phi \log \pi_\phi$ with $r_\phi = \pi_\phi/\pi_{\phi_{\text{old}}}$ identifies the objective this gradient belongs to,

$$\mathcal{J}^{\text{ideal}}_\phi = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\pi_{\theta_{\text{old}}}}\mathbb{E}_{\pi_{\phi_{\text{old}}}}\big[r_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}}) r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}}\big]. \tag{18}$$

Updating the Allocator instead freezes the backbone at $\pi_\phi = \pi_{\phi_{\text{old}}}$ and marginalizes the response:

$$\mathcal{J}^{\text{ideal}}_\theta = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\mathbb{E}_{\pi_{\theta_{\text{old}}}}\big[r_\theta(\boldsymbol{s} \mid \boldsymbol{x})\, r^{\text{ideal}}_{\boldsymbol{s}}\big], \qquad r_\theta(\boldsymbol{s} \mid \boldsymbol{x}) = \frac{\pi_\theta(\boldsymbol{s} \mid \boldsymbol{x})}{\pi_{\theta_{\text{old}}}(\boldsymbol{s} \mid \boldsymbol{x})}, \tag{19}$$

where $r^{\text{ideal}}_{\boldsymbol{s}} = \mathbb{E}_{\pi_{\phi_{\text{old}}}(\boldsymbol{y}|\tilde{\boldsymbol{x}})}[r^{\text{ideal}}_{\boldsymbol{s},\boldsymbol{y}}]$ is estimated by the rollouts sampled under the frozen backbone. With $N = 1$ rollout per allocation, that estimate is unbiased but high-variance, which is why CAPO compares allocations inside a group rather than trusting any single rollout reward.

B.3 From Ideal Rewards to the CAPO Surrogates

CAPO replaces r^{ideal} with the advantage $A_{m,n}$ of Eq. (7). For a fixed context, let $G = MN$, $\bar{r} = G^{-1}\sum_{m,n} r^{\text{base}}_{m,n}$, and $\sigma_r^2 = G^{-1}\sum_{m,n}(r^{\text{base}}_{m,n} - \bar{r})^2$. The task advantage is $(r^{\text{base}}_{m,n} - \bar{r})/\sigma_r$ for $\sigma_r > 0$ and zero otherwise. Advantages, group statistics, and sampled actions are held fixed during the policy update. Because the group baseline includes each sample and the standard deviation is sample-dependent, this normalized update is a surrogate rather than an unbiased estimator of Eq. (13).

The first change concerns how cost is charged. The cost penalty alone favors assigning the minimum scale $s_{\min}$ to every frame. CAPO instead conditions cost pressure on task success through the advantage $A_{m,n}$, using the smooth scale coordinate c_m rather than the discrete retention R. The two coordinates are related but not equivalent: under the uniform-resolution approximation $R(\boldsymbol{s}) \approx T^{-1}\sum_t s_t^2$ derived in Section C,

$$R(\boldsymbol{s}_m) \approx \big[s_{\min} + (s_{\max} - s_{\min})c_m\big]^2 + \text{Var}_t(s_{m,t}), \qquad \text{Var}_t(s_{m,t}) = \frac{1}{T}\sum_{t=1}^{T}(s_{m,t} - \bar{s}_m)^2, \tag{20}$$

Thus c_m controls mean scale, while the second moment determines the equal-area token approximation. The global term $-\gamma c_m$ supplies compression pressure without being algebraically equal to $-\lambda_{\text{bud}} R$.

The practical losses make two further choices: they replace products of likelihood ratios by averages of coordinate-wise surrogates, then clip each coordinate. For M allocations with N rollouts each, the sequence-level backbone surrogate is

$$\mathcal{L}_\phi^{\text{seq}} = -\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\min\Big(r_\phi^{(m,n)}\,A_{m,n},\ \text{clip}(r_\phi^{(m,n)}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}})\,A_{m,n}\Big), \tag{21}$$

and factorizing $\pi_\phi(\boldsymbol{y} \mid \tilde{\boldsymbol{x}})$ into token conditionals while reusing the rollout-level advantage across tokens gives the loss we optimize,

$$\mathcal{L}_\phi = -\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\frac{1}{L_{m,n}}\sum_{j=1}^{L_{m,n}}\min\Big(r_{\phi,j}^{(m,n)}\,A_{m,n},\ \text{clip}\big(r_{\phi,j}^{(m,n)}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big)\,A_{m,n}\Big), \tag{22}$$

where $L_{m,n}$ is the rollout length, $0 < \varepsilon_{\text{low}} < 1$ and $\varepsilon_{\text{high}} > 0$ are the lower and upper clipping margins, and the token-level importance ratio is

$$r_{\phi,j}^{(m,n)} = \frac{\pi_\phi(y_j^{(m,n)} \mid y_{<j}^{(m,n)}, \tilde{\boldsymbol{x}}^{(m)})}{\pi_{\phi_{\text{old}}}(y_j^{(m,n)} \mid y_{<j}^{(m,n)}, \tilde{\boldsymbol{x}}^{(m)})}. \tag{23}$$

Here $r_\phi^{(m,n)} = \prod_{j=1}^{L_{m,n}} r_{\phi,j}^{(m,n)}$, including the termination token when present. Equation (22) uses a token-wise surrogate with a shared rollout advantage. At the behavior policy, its unclipped per-rollout gradient is $1/L_{m,n}$ times the sequence score-function gradient; variable output lengths therefore change the relative weighting of rollouts. In joint training, the shared CAPO advantage also couples response learning to allocation quality. Equation (16) applies to the pure input-cost term, not to the nonlinear success-gated shaping. A task-only backbone advantage separates these signals, but defines a different training variant.

The Allocator side follows the same pattern one level up. With $\Delta s = s_{\max} - s_{\min} > 0$, $q_{\theta,t}(a_t \mid \boldsymbol{x}) = \text{Beta}(a_t; \alpha_t, \beta_t)$, and $a_t = (s_t - s_{\min})/\Delta s$, the change of variables of Section 3.1 gives

$$\log \pi_\theta(\boldsymbol{s} \mid \boldsymbol{x}) = \sum_{t=1}^{T}\log \text{Beta}(a_t; \alpha_t, \beta_t) - T\log \Delta s. \tag{24}$$

The Jacobian term is constant in θ and cancels from the new-to-old ratio because both policies share the same fixed bounds, so Allocator ratios are computed directly from the latent Beta densities and the score function is a sum of per-frame terms,

$$\nabla_\theta \log \pi_\theta(\boldsymbol{s}^{(m)} \mid \boldsymbol{x}) = \sum_{t=1}^{T}\nabla_\theta \log \text{Beta}(a_t^{(m)};\, \alpha_t, \beta_t). \tag{25}$$

Clipping each per-frame ratio $r_{\theta,t}^{(m)} = q_{\theta,t}(a_t^{(m)} \mid \boldsymbol{x})/q_{\theta_{\text{old}},t}(a_t^{(m)} \mid \boldsymbol{x})$ and charging it the allocation-level advantage $A_m^{\text{CAPO}} = N^{-1}\sum_n A_{m,n}$ (which simplifies to $A_m^{\text{CAPO}} = A_{m,1}$ when $N{=}1$) yields

$$\mathcal{L}_\theta = -\frac{1}{MT}\sum_{m=1}^{M}\sum_{t=1}^{T}\min\Big(r_{\theta,t}^{(m)}A_m^{\text{CAPO}}, \text{clip}\big(r_{\theta,t}^{(m)}, 1-\varepsilon_{\text{low}}, 1+\varepsilon_{\text{high}}\big)A_m^{\text{CAPO}}\Big). \tag{26}$$

The factorization is conditional on the full context; frame representations can interact through attention while the sampled latents remain conditionally independent. In particular,

$$r_\theta(\boldsymbol{s}_m \mid \boldsymbol{x}) = \prod_{t=1}^{T} r_{\theta,t}^{(m)}, \qquad \prod_t(1+\delta_t) = 1 + \sum_t \delta_t + \mathcal{O}(\|\boldsymbol{\delta}\|^2), \quad \delta_t = r_{\theta,t}^{(m)} - 1. \tag{27}$$

The sum $\sum_t r_{\theta,t}^{(m)} - (T-1)$ matches this first-order expansion. Equation (26) instead averages the coordinate surrogates, giving, at $\theta = \theta_{\text{old}}$,

$$\nabla_\theta\left[\frac{A_m^{\text{CAPO}}}{T}\sum_t r_{\theta,t}^{(m)}\right] = \frac{A_m^{\text{CAPO}}}{T}\sum_t \nabla_\theta \log q_{\theta,t}(a_{m,t} \mid \boldsymbol{x}) = \frac{A_m^{\text{CAPO}}}{T}\nabla_\theta \log \pi_\theta(\boldsymbol{s}_m \mid \boldsymbol{x}). \tag{28}$$

This is a scaled local first-order relation before clipping, not an equality between the objectives away from the behavior policy. Training uses $T = 128$ and the frame-mean reduction in Eq. (26); changing only the inference horizon does not alter this training normalization. Changing the training horizon changes the balance between policy and auxiliary gradients unless their scaling is adjusted.

For either policy, define $\ell(r, A) = \min(rA, \mathrm{clip}(r, 1 - \varepsilon_{\mathrm{low}}, 1 + \varepsilon_{\mathrm{high}})A)$. For $A > 0$, the objective becomes flat above the upper threshold; for $A < 0$, it becomes flat below the lower threshold. Clipping limits the incentive from that sample in one direction, rather than imposing a hard bound on r. Even if every marginal ratio stayed inside the clipping interval, their product could range from $(1 - \varepsilon_{\mathrm{low}})^T$ to $(1 + \varepsilon_{\mathrm{high}})^T$. Coordinate clipping therefore provides no hard joint-ratio or KL guarantee.

B.4 Cost Shaping and Regularizer Properties

Success floor. For an all-success group with equal base rewards (including the format bonus), $A^{\mathrm{base}}_{m,n} = 0$ and, holding the detached pivot fixed, the raw advantage is

$$\widetilde{A}(c) = \lambda_{\mathrm{capo}}\lambda_+\sigma((\tau_{\mathrm{dyn}} - c)/\tau_{\mathrm{s}}) - \gamma c, \qquad \frac{\mathrm{d}\widetilde{A}}{\mathrm{d}c} = -\frac{\lambda_{\mathrm{capo}}\lambda_+}{\tau_{\mathrm{s}}}\sigma(z)(1 - \sigma(z)) - \gamma \le 0, \quad (29)$$

where $z = (\tau_{\mathrm{dyn}} - c)/\tau_{\mathrm{s}}$. Successful samples can still have different task or format rewards; their base advantages then remain in $\widetilde{A}$. At fixed task advantage, the floor $A(c) = \max(\widetilde{A}(c), \varepsilon_+)$ preserves cost ordering only above the floor and assigns equal weight to saturated successes. A constant advantage ε_+ over the full action distribution has zero expected on-policy score gradient, since $\mathbb{E}[\nabla_\theta \log \pi_\theta] = 0$. This cancellation need not hold for only the successful subset. Finite-sample updates, multiple PPO epochs, and nonconstant advantages can behave differently. The floor protects successful samples from negative surrogate weights; convergence to the pivot is not a consequence of the floor alone.

Adjacent allocation. With $h_t = \log \mu_t + \log \mu_{t+1} + \eta_{\mathrm{sim}}$, an active pair contributes derivatives $w_t/((T-1)\mu_t)$ and $w_t/((T-1)\mu_{t+1})$ to its two mean scales. Both derivatives are nonnegative and are unchanged by exchanging the two frames. Chain-rule derivatives follow from

$$\frac{\partial \mu_t}{\partial \alpha_t} = \frac{(s_{\max} - s_{\min})\beta_t}{(\alpha_t + \beta_t)^2}, \qquad \frac{\partial \mu_t}{\partial \beta_t} = -\frac{(s_{\max} - s_{\min})\alpha_t}{(\alpha_t + \beta_t)^2}. \quad (30)$$

The zero-loss region is $\mu_t\mu_{t+1} \le e^{-\eta_{\mathrm{sim}}}$. With $\eta_{\mathrm{sim}} = 1$, this allows a mean scale of 1.79 beside 0.201 (product $0.360 < e^{-1}$), while two native-resolution frames incur a penalty. Finite $\alpha_t, \beta_t > 1$ keep means strictly inside $(s_{\min}, s_{\max})$. This is a soft preference for asymmetric fidelity on redundant pairs, and equal low scales also satisfy the product bound, so the regularizer can reduce overall budget as well as discourage simultaneous high-resolution pairs. It neither forces asymmetry by itself nor enforces a hard exclusion rule; task feedback supplies the query-dependent selection pressure. For padded batches, each video's loss averages over valid adjacent pairs before averaging over videos; empty pair sets contribute zero. The feature gate is detached and the mean scales remain differentiable.

Concentration. Write $p_t = \mathbb{E}[a_t]$ and $\kappa_t = \alpha_t + \beta_t$. Then $\mathrm{Var}(a_t) = p_t(1 - p_t)/(\kappa_t + 1)$ and $\mathrm{Var}(s_t) = (s_{\max} - s_{\min})^2 \mathrm{Var}(a_t)$. At fixed p_t, a hard cap $\kappa_t \le \kappa_{\max}$ would imply the corresponding variance lower bound. Equation (9) is a soft hinge penalty, so it encourages rather than guarantees that cap. Unlike penalizing κ_t everywhere, the hinge has zero gradient below the threshold and does not continually push both Beta parameters toward their minimum. Above the threshold its gradients can still change the mean through the shared heads; it is not a mean-preserving projection.

C Complexity Analysis

We analyze visual-token counts and the quadratic attention terms in the backbone and Allocator decoder. We write T for the frame count and $H \times W$ for the native resolution; heterogeneous inputs replace HW with frame-specific H_tW_t.

Long-context efficiency also depends on text representations and cache dimensions. HyCo$_2$ combines global context compression with local token retention (Liao et al., 2025b), while SparK applies query-aware KV-channel pruning with dynamic recovery during attention computation (Liao et al., 2025c). SKIntern reduces inference context by progressively internalizing symbolic knowledge during training (Liao et al., 2025a). Input resizing acts on the visual sequence length entering the backbone.

Baseline cost. Let P denote the ViT patch size and let $n_t(\boldsymbol{s})$ be the visual-token count attributed to frame t after the backbone processor collates the full scale vector. The exact full-resolution baseline is $N_0 := \sum_{t=1}^{T} n_t(\mathbf{1})$. Under direct patching with uniform native resolution, this count reduces to

$$N_0 \;=\; T \cdot \left\lceil \frac{H}{P} \right\rceil \left\lceil \frac{W}{P} \right\rceil \;\approx\; \frac{THW}{P^2}. \tag{31}$$

Adaptive cost and Retention Ratio. For resizing, the exact token-count *Retention Ratio* is

$$N^{\text{adapt}} = \sum_{t=1}^{T} n_t(\boldsymbol{s}), \qquad R(\boldsymbol{s}) \;\triangleq\; \frac{N^{\text{adapt}}}{N_0}. \tag{32}$$

For independently collated frames, $n_t(s_t) \approx s_t^2 H_t W_t / P^2$. With baseline-token weights $b_t = n_t(\mathbf{1}) / \sum_u n_u(\mathbf{1})$, the area approximation and the mean-scale estimate are

$$R_{\text{area}} = \sum_t b_t s_t^2 = \bar{s}_b^2 + \text{Var}_b(s), \qquad R_{\text{mean}} = \bar{s}_b^2, \qquad \bar{s}_b = \sum_t b_t s_t. \tag{33}$$

Here $\text{Var}_b(s) = \sum_t b_t (s_t - \bar{s}_b)^2$. For equal-sized frames, $b_t = 1/T$. Squaring a logged mean scale yields R_{mean}, whereas averaging squared scales yields R_{area}. Their difference is the scale variance. Processor rounding, pixel limits, spatial merging, and temporal grouping determine the realized token ratio R. Benchmark tables report the mean over evaluation samples of this processor-counted ratio. Eq. (33) provides the closed-form area approximation used in complexity analysis, whereas the training surrogate c_m in CAPO operates on the normalized arithmetic mean scale $\bar{s}_m$ as a smooth linear proxy to avoid processor rounding plateaus (with their quadratic relationship derived in Eq. (20)); mean-only estimates are identified as R_{mean}.

Quadratic attention reduction. Let N_{text} denote the nonvisual prefill length. For an L_{mllm}-layer MLLM with hidden dimension D_{mllm}, the quadratic self-attention terms scale as

$$\Phi_{\text{base}} = \mathcal{O}\big(L_{\text{mllm}} (N_{\text{text}} + N_0)^2 D_{\text{mllm}}\big)\,, \tag{34}$$

$$\Phi_{\text{adapt}} = \mathcal{O}\big(L_{\text{mllm}} (N_{\text{text}} + R N_0)^2 D_{\text{mllm}}\big)\,. \tag{35}$$

Their ratio approaches R^2 only when visual tokens dominate prefill. At $R = 0.11$ that limit is 1.2% of dense attention cost (83×), and end-to-end FLOPs fall by considerably less.

Allocator computation. Let K be the number of preview patch tokens per frame, ℓ_q the query length, $D = D_{\text{pred}}$, and L_{pred} the number of prediction layers. The prediction blocks operate on $T \times K$ states. Spatial attention costs $\mathcal{O}(TK^2D)$, patch-wise temporal attention $\mathcal{O}(KT^2D)$, and query cross-attention $\mathcal{O}(TK\ell_q D)$ per layer. Projections and feed-forward layers add $\mathcal{O}((TK + \ell_q)D^2)$. Writing C_{prev} for the frozen encoder cost per frame and $C_{\text{text}}(\ell_q)$ for query encoding, the full front-end cost is

$$\begin{aligned} \Phi_{\text{front}} = \mathcal{O}\Big(& T C_{\text{prev}} + C_{\text{text}}(\ell_q) + \sum_t s_t^2 H_t W_t \\ & + L_{\text{pred}} [TK^2D + KT^2D + TK\ell_q D + (TK + \ell_q)D^2] \Big). \end{aligned} \tag{36}$$

Pooling after the decoder adds $\mathcal{O}(TKD)$. Frozen encoders still contribute their full forward cost: pooling complexity alone does not describe feature extraction. At 128 frames, preview encoding accounts for 734.5 of 915.5 ms front-end time (80.2%); query encoding takes 20.4 ms, prediction 92.0 ms, and resizing 68.6 ms (Table 3). Across the measured horizons, prediction takes 85.6–97.4 ms.

Downstream visual encoding. For L_v visual layers of width D_v and $k_t(s_t)$ encoder-input patches per frame, full spatial attention has cost $\mathcal{O}(L_v \sum_t [k_t(s_t)^2 D_v + k_t(s_t) D_v^2])$, so the quadratic term

scales as s_t^4 when $k_t \propto s_t^2$. Windowed attention replaces the quadratic term by $\mathcal{O}(k_t(s_t)wD_v)$ for window size w; temporal grouping is accounted for in the processor's effective patch grids. With $k_t(s_t) \approx s_t^2 k_t(1)$, resizing reduces both attention and token-wise visual computation before language prefill. Encoder patch counts and language visual-token counts can differ because of spatial merging. Area $\sim s_t^2$ is therefore the right scaling for token-wise work and for windowed attention, not a universal law for every encoder layer.

Attention-only speedup. For $0 < R \leq 1$, the idealized speedup of the quadratic prefill term is

$$\text{Speedup}_{\text{attn}} = \frac{(N_{\text{text}} + N_0)^2}{(N_{\text{text}} + RN_0)^2} \leq \frac{1}{R^2}, \tag{37}$$

reaching the $1/R^2$ upper bound in the asymptotic limit where visual tokens dominate text tokens ($N_{\text{text}}/N_0 \to 0$). This estimate concerns the quadratic attention term rather than total runtime. Table 3 reports measured end-to-end speedups, including the Allocator.

Temporal context scaling. Under a fixed token budget B_{tok}, a dense model admits $T_0 = B_{\text{tok}}P^2/(HW)$ full-resolution frames, whereas average retention R admits a nominal horizon of T_0/R. Our 8-frame-equivalent protocol spends that headroom on 32–128 frames, a $16\times$ nominal horizon over the dense 8-frame reference, and Figure 3c reports what the exchange buys in accuracy.

Remark (Vision Processor and Temporal Tubelet Compatibility). ResAdapt integrates the Allocator into the video processor: predicted scales determine frame sizes before patch extraction, and the processor assembles the resized inputs with their temporal metadata. Qwen2.5-VL uses temporal tubelets of size two and a spatial merge granularity of 28 pixels. Spatial divisibility and temporal collation are separate requirements: frames within a tubelet must have compatible grids, while differently sized units are represented by separate grid entries. Consequently $n_t(\boldsymbol{s})$ is defined after collation: when two frames share a tubelet, the token count depends on the pair of scales rather than on s_t alone. Table budgets average this processor-counted per-sample R. The Qwen2.5-VL serving adapter aligns mRoPE positions with the processor's resized sequence and timestamps. This model-specific runtime adaptation leaves the learned vision architecture and attention kernels unchanged.

Remark (acceleration transparency). After processor collation, the backbone receives standard visual-token tensors and uses its existing attention kernels. Our vLLM measurements (Kwon et al., 2023) include front-end processing. The same separation between preprocessing and attention permits integration with other serving stacks, including SGLang (Zheng et al., 2024), through their multimodal input adapters.

D Implementation Details

The implementation is available in our anonymous repository: `https://anonymous.4open.science/r/ResAdapt`.

D.1 Training Data

Data Composition. We build the training set by filtering the VideoAuto-R1 (Liu et al., 2026) corpus by difficulty, keeping image and video samples and removing text-only instances. To cover visually demanding subdomains more densely, we add 16,500 video instances sampled from Video-R1 (Feng et al., 2025) that target OCR, free-form QA, and regression tasks. The 93.4K-sample training set is filtered from VideoAuto-R1 and Video-R1 and is disjoint from the reported evaluation benchmarks.

D.2 Training Configuration

The Allocator adds L_{pred}=4 decoder layers of width D_{pred}=1,024 on top of SmolVLM's frozen visual tower. Unless otherwise specified, we train for a single epoch with AdamW (Loshchilov & Hutter, 2017) at a global batch size of 128, weight decay 0.01, and gradient-norm clipping at 1.0. In standard ResAdapt, the downstream MLLM backbone is completely frozen (lr $= 0$, parameters fixed); the Allocator is trained with learning rate 2×10^{-5}. For the joint training variant (ResAdapt-RL), the backbone is additionally updated with learning rate 1×10^{-6}. The maximum video token budget is

8,192, training samples T=128 frames, and scaling factors are restricted to $[s_{\min}, s_{\max}] = [0.2, 1.8]$, which permits aggressive downscaling and selective upscaling within one action space. CAPO draws M=16 allocation trajectories per prompt with N=1 rollout each. Training runs on 32 H100 GPUs with VeRL (Sheng et al., 2025), DeepSpeed (Rasley et al., 2020), and vLLM (Kwon et al., 2023); evaluation runs in lmms-eval (Zhang et al., 2024a) with responses capped at 256 tokens, or 4,096 for reasoning-heavy models.

CAPO and regularization. We use $\kappa_{\text{mix}} = 0.5$, $\tau_{\text{fix}} = 0.4$, $\tau_{\text{s}} = 0.15$, $\lambda_{\text{capo}} = 0.25$, $(\lambda_+, \lambda_-) = (0.9, 1.2)$, $\omega_0 = 0.05$, $\gamma = 0.05$, and $\varepsilon_+ = 10^{-3}$. The temporal-similarity parameters are $(\tau_{\text{sim}}, \gamma_{\text{sim}}, \eta_{\text{sim}}) = (0.5, 0.05, 1.0)$ with $\lambda_{\text{sim}} = 10^{-3}$, and the concentration penalty threshold is $\kappa_{\max} = 20$ with $\lambda_{\text{con}} = 5 \times 10^{-4}$. Following asymmetric clipping (Yu et al., 2025), we set $(\varepsilon_{\text{low}}, \varepsilon_{\text{high}}) = (0.2, 0.25)$, giving clipping thresholds $[0.8, 1.25]$. These values define the main CAPO configuration.

D.3 Evaluation Protocol

Budget accounting. Each experiment uses the frame count stated in its table, with same-backbone dense input defining the reference budget. Reported visual budgets R in the benchmark tables represent the mean per-sample realized visual-token retention measured directly through processor collation relative to dense input (with Table 1 explicitly reporting processor-counted ratios). Budgets can also be estimated from scale statistics via Eq. (33) (R_{area} or R_{mean}), which provides a closed-form area approximation when per-token processor tallies are omitted. Retained-token or frame fractions describe merging and selection baselines. Table-wide budgets average over the included evaluation tasks. Because resizing reduces visual-encoder input at the source, token retention reflects downstream language sequence length, while pixel-level encoder work scales with input area. Discrete baselines use their nearest feasible operating points. Table 10 reports end-to-end time. The OCR floor comparison and agentic composition study explicitly report R_{mean} where costs are estimated from mean scales (Tables 9 and 18).

Video QA. Scoring follows the benchmark implementations in lmms-eval (Zhang et al., 2024a). The six video-QA datasets report their native accuracy-based scores, and aggregate tables take the unweighted mean of those six values. Unless a row is labeled ResAdapt-RL, the downstream backbone is frozen for both Allocator training and evaluation.

Temporal grounding. Charades-STA and ActivityNet report recall at temporal IoU thresholds 0.3, 0.5, and 0.7 alongside mIoU; NExT-GQA reports answer accuracy and grounding mIoU; the main table quotes mIoU only. Predictions from which no valid, non-degenerate interval can be parsed score zero, a convention applied identically to every method. Because ResAdapt rescales frames instead of removing them, all T sampled time points still reach the backbone and predicted timestamps stay in the time base of the full-resolution reference. Baselines that change the frame count do not share this property: the FixedScale row at 6.3% halves the frame count, buying per-frame resolution at the price of a coarser temporal grid, and its NExT-GQA entries should be read in that light.

Table 7 reports Retention Ratio R: ResAdapt uses 16.2% and 6.8% retention, while the corresponding baselines use 25.0%, 12.3%, or 6.3%. The reasoning-augmented comparison pairs ResAdapt with VideoAuto-R1 at low retention and uses full-retention VideoAuto-R1 as the reference. On NExT-GQA, the non-reasoning 32-frame Qwen2.5-VL results span 74.1–78.9 in answer accuracy and 12.3–28.0 in grounding mIoU, showing greater variation in localization quality. At 6.8% retention, ResAdapt reaches 74.3 answer accuracy versus FlashVid's 77.4 at 12.6%, while leading grounding mIoU by 4.3 points in the same budget block.

D.4 Reward Design

This section completes the reward formulation of Sec. 3.2. The base scalar reward $r^{\text{task}}_{m,n}$ measures task performance; efficiency enters through CAPO's advantage shaping, never as a hard budget constraint. These parsers define the *training* signal only; the reported benchmarks are scored by the lmms-eval implementations described above.

Base Task Reward ($r^{\text{task}}_{m,n}$). Training feedback is computed across four primary task types:

- *Question Answering.* For math problems, we extract the numeric solution and check it against the ground truth within a 10^{-2} tolerance. For multiple-choice queries, we parse the predicted option letter. For the remaining QA tasks, we compare normalized strings (*e.g.*, case-folded, whitespace-stripped). Each case yields the binary reward

$$r_{\text{QA}}(\hat{o}, o) \in \{0, 1\}.$$

- *Free-form Generation.* For open-ended tasks, we score the generated answer $\hat{o}$ against the reference o with ROUGE-L:

$$r_{\text{Gen}}(\hat{o}, o) = \text{ROUGE-L}(\hat{o}, o) \in [0, 1].$$

- *Temporal Grounding.* The evaluator parses a single predicted interval $\hat{g} = [\hat{b}, \hat{e}]$ from the model output, computes temporal IoU against each of the valid ground-truth intervals $\mathcal{G} = \{[b_j, e_j]\}_j$, and keeps the best match:

$$r_{\text{TG}}(\hat{g}, \mathcal{G}) \;=\; \max_{[b,e]\in\mathcal{G}} \text{tIoU}(\hat{g}, [b, e]) \in [0, 1].$$

  The parser returns exactly one numeric start–end pair, reorders reversed endpoints, and resolves outputs containing several interval-like patterns by a deterministic precedence; when no valid, non-degenerate interval survives, the reward is zero.

- *Grounding QA.* We extract the textual answer and one temporal interval from the same output, score them independently, and add the two terms:

$$r_{\text{GQA}}(\hat{o}, \hat{g}; o, \mathcal{G}) \;=\; r_{\text{QA}}(\hat{o}, o) + r_{\text{TG}}(\hat{g}, \mathcal{G}) \;\in [0, 2].$$

These metrics define $r^{\text{task}}_{m,n}$. The binary success gate has a distinct role: it selects the success branch of cost shaping. Exact-match QA supplies $u_{m,n}$ directly; single-metric generation and grounding use a threshold of 0.35. For Grounding-QA,

$$u_{m,n} = \mathbf{1}\{r_{\text{QA}} = 1\}\, \mathbf{1}\{\hat{g} \text{ is valid}\}\, \mathbf{1}\{r_{\text{TG}} \geq 0.35\}. \tag{38}$$

The additive reward retains partial credit for either component, while the conjunctive gate reserves the success branch for jointly correct answering and localization. With format validation, group normalization uses $r^{\text{base}}_{m,n} = r^{\text{task}}_{m,n} + 0.2 r_{\text{fmt}}$; the success gate remains independent of the format bonus.

Format Reward. A binary response-format reward $r_{\text{fmt}}(\boldsymbol{y}) \in \{0, 1\}$ is assigned by regular-expression matching. The output must contain exactly one `<think>...</think>` block and one `<answer>...</answer>` block, with the final answer in `\boxed{...}` strictly inside the `<answer>` tags:

$$r_{\text{fmt}}(\boldsymbol{y}) = \begin{cases} 1 & \text{if format matches regex,} \\ 0 & \text{otherwise.} \end{cases}$$

Valid outputs receive a format bonus of weight 0.2; invalid outputs receive none.

D.5 Prompt Template

Table 5 gives the training prompt, which places the reasoning trace inside `<think> </think>` tags and the final answer inside `\boxed{}`. The Allocator emits no text; the backbone's response format makes training reward extraction deterministic.

E Supplementary Experiments and Analysis

This supplement follows the order of the main experimental claims: complete benchmark and latency tables first, then allocation behavior and the roles of query conditioning, CAPO, and temporal regularization, and finally transfer, composition with other inference strategies, and representative successes and failures. Unless stated otherwise, plots use Qwen2.5-VL-7B and 32 frames.

Table 5: CAPO training prompt. The model receives the video and query, writes its reasoning inside `<think>` tags, and places the boxed answer inside `<answer>` tags for deterministic reward parsing.

Prompt Template for Training with Thinking

System Prompt:
You are a helpful assistant.
You FIRST think about the reasoning process as an internal monologue and then provide the final answer.
The reasoning process MUST BE enclosed within `<think> </think>` tags and the answer MUST BE enclosed within `<answer> </answer>` tags.
The final answer MUST BE put in `\boxed{}` and the `\boxed{}` expression MUST BE contained entirely within the `<answer> </answer>` tags.
Do not include any reasoning or explanations outside these tags.

E.1 Full Evaluation Results

E.1.1 Aggregate Video-QA Consistency

Table 6 averages the six video-QA columns of Table 1 within comparable retention bands. ResAdapt has the highest compressed mean in seven of the eight blocks, including a 0.02-point margin over FlashVid on Qwen3-VL at 32 frames and ∼25% retention (55.55 vs. 55.53). In the remaining block (Qwen2.5-VL at 128 frames and ∼10% retention), FixedScale reaches 54.38 against 54.13 at a larger budget (12.3% vs. 11.1%).

Table 6: Six-benchmark mean video-QA accuracy at comparable budgets. Means follow Table 1. **Bold**: best compressed result. –: not evaluated in that block.

Method	32 Frames		128 Frames	
	∼25%	∼10%	∼25%	∼10%
Qwen2.5-VL-7B				
Vanilla (100%)	54.15	54.15	55.85	55.85
Random Drop	51.10	48.38	55.18	52.65
ToMe (Bolya et al., 2023)	50.98	48.55	55.07	50.98
VisionZip (Yang et al., 2025a)	50.52	48.22	55.08	51.03
FlashVid (Fan et al., 2026)	52.05	49.37	–	–
FixedScale	51.97	49.67	–	**54.38**
ResAdapt (Ours)	**52.87**	**50.67**	**56.28**	54.13
Qwen3-VL-8B				
Vanilla (100%)	57.68	57.68	62.27	62.27
Random Drop	54.70	50.98	58.40	55.75
ToMe (Bolya et al., 2023)	53.68	50.80	58.68	55.42
VisionZip (Yang et al., 2025a)	53.80	51.50	58.55	55.22
FlashVid (Fan et al., 2026)	55.53	52.82	–	–
FixedScale	–	52.93	–	57.72
ResAdapt (Ours)	**55.55**	**54.15**	**59.88**	**58.77**

E.1.2 Temporal Grounding Evaluation

Table 7 reports temporal-grounding results under the protocol in Appendix D.3.

Table 7: Temporal grounding results. R is mean realized visual-token retention over tasks in this table. †: FixedScale at 6.3% halves the frame count, raising per-frame resolution for NExT-GQA answer reading. *CoT* marks chain-of-thought inference models. **Bold**: best compressed result without CoT in each frame-and-budget block.

Backbone	Method	Retention Ratio R	CoT	Temporal Grounding Benchmark									
				Charades-STA				ActivityNet				NExT-GQA	
				0.3	0.5	0.7	mIoU	0.3	0.5	0.7	mIoU	Acc	mIoU
Qwen2.5-VL-7B	*32 Frames*												
	Vanilla	100%	✗	71.0	51.4	26.0	47.3	30.4	18.0	8.9	22.6	78.9	28.0
	Random Drop	25.0%	✗	39.4	23.2	11.0	25.7	15.2	8.1	3.7	11.7	77.5	16.6
	ToMe (Bolya et al., 2023)	25.0%	✗	39.5	23.9	11.4	26.0	16.0	8.4	4.0	12.1	77.8	16.3
	FlashVid (Fan et al., 2026)	31.3%	✗	40.7	24.2	11.3	26.6	15.8	8.4	3.8	12.0	**78.1**	16.5
	FixedScale	25.0%	✗	36.7	24.7	12.3	24.9	18.6	9.4	4.3	14.1	77.7	12.3
	ResAdapt (Ours)	16.2%	✗	**53.8**	**34.8**	**17.0**	**35.6**	**19.8**	**10.8**	**5.2**	**15.3**	76.6	**23.2**
	Random Drop	10.0%	✗	36.9	23.2	11.6	24.6	14.3	7.5	3.6	11.1	76.3	15.4
	ToMe (Bolya et al., 2023)	10.0%	✗	41.3	26.9	14.1	27.4	16.0	8.4	**4.0**	12.2	77.3	15.7
	FlashVid (Fan et al., 2026)	12.6%	✗	38.2	22.9	11.1	25.1	15.4	8.1	3.7	11.8	**77.4**	16.1
	FixedScale	12.3%	✗	**48.0**	**31.5**	**15.4**	**32.0**	**17.5**	**8.9**	**4.0**	**13.3**	76.1	13.7
	FixedScale	6.3%	✗	39.9	26.8	13.3	26.7	15.2	8.1	3.9	11.9	74.1	15.4
	ResAdapt (Ours)	6.8%	✗	41.0	27.8	14.0	27.2	16.3	8.5	3.9	12.5	74.3	**20.4**
	VideoAuto-R1 (Liu et al., 2026)	100%	✓	60.0	48.3	27.2	41.5	50.8	34.1	17.4	34.4	73.6	33.8
	+ **ResAdapt (Ours)**	23.8%	✓	72.8	53.0	27.5	49.1	65.8	44.9	23.8	44.7	79.3	35.3
	+ **ResAdapt (Ours)**	6.8%	✓	43.5	30.1	15.8	30.0	35.4	21.5	10.0	24.4	74.7	24.7
	128 Frames												
	Vanilla	100%	✗	77.5	60.3	34.1	52.8	47.9	30.9	17.5	34.4	79.8	29.9
	Random Drop	25.0%	✗	32.3	19.6	7.9	20.7	26.7	13.9	6.3	18.8	**80.3**	10.7
	ToMe (Bolya et al., 2023)	25.0%	✗	32.4	19.8	7.9	20.7	27.2	14.4	6.4	19.1	**80.3**	10.9
	ResAdapt (Ours)	16.1%	✗	**63.5**	**43.6**	**21.3**	**42.0**	**33.1**	**19.3**	**10.2**	**24.3**	78.1	**27.2**
	Random Drop	10.0%	✗	37.8	23.8	11.2	24.7	23.8	12.0	5.3	17.0	**79.4**	12.8
	ToMe (Bolya et al., 2023)	10.0%	✗	27.9	16.2	7.3	17.9	22.9	11.8	5.5	16.4	79.1	11.1
	FixedScale	12.3%	✗	34.7	22.3	10.5	22.7	**25.0**	**13.8**	5.9	**18.3**	77.9	11.3
	FixedScale	6.3%	✗	42.6	28.4	14.3	28.3	22.8	12.8	5.7	17.1	75.7	12.9
	ResAdapt (Ours)	6.8%	✗	**43.5**	**29.8**	**15.0**	**28.9**	23.5	12.9	**6.1**	17.2	76.2	**23.9**
	VideoAuto-R1 (Liu et al., 2026)	100%	✓	40.3	33.7	22.1	28.9	49.4	34.3	18.5	33.5	68.0	31.0
	+ **ResAdapt (Ours)**	16.1%	✓	76.2	55.5	28.8	51.4	67.9	46.3	24.5	46.1	81.2	37.3
	+ **ResAdapt (Ours)**	6.8%	✓	50.1	33.2	16.6	34.2	53.4	34.0	16.4	35.7	76.6	29.4
Qwen3-VL-8B	*32 Frames*												
	Vanilla	100%	✗	73.0	49.0	21.4	46.4	44.6	28.3	15.5	31.8	78.7	34.2
	Random Drop	25.0%	✗	16.2	8.6	3.8	12.1	12.4	6.7	3.2	10.0	77.2	15.6
	ToMe (Bolya et al., 2023)	25.0%	✗	68.7	42.1	17.6	43.1	45.9	28.8	15.6	32.6	77.1	31.7
	FlashVid (Fan et al., 2026)	31.3%	✗	**72.9**	**52.3**	**25.1**	**47.7**	**51.9**	**33.4**	**19.0**	**36.8**	**77.8**	**33.9**
	ResAdapt (Ours)	16.2%	✗	64.4	37.3	16.3	39.9	40.0	24.4	13.0	28.5	75.1	30.2
	Random Drop	10.0%	✗	4.1	1.8	0.7	4.4	4.7	2.4	1.0	5.0	74.3	11.3
	ToMe (Bolya et al., 2023)	10.0%	✗	67.6	39.3	16.6	41.8	46.3	31.0	**19.2**	34.1	**79.2**	**34.0**
	FlashVid (Fan et al., 2026)	12.6%	✗	**68.8**	**46.9**	**22.9**	**44.6**	**49.9**	**31.5**	17.4	**35.2**	75.6	31.8
	FixedScale	12.3%	✗	61.3	34.3	14.6	37.9	39.6	24.2	13.1	28.4	74.2	29.9
	FixedScale	6.3%	✗	52.7	28.2	11.3	33.2	37.0	22.3	12.0	27.0	71.5	28.0
	ResAdapt (Ours)	6.8%	✗	53.6	29.0	11.8	33.6	37.5	22.5	12.3	27.2	71.8	28.2
	128 Frames												
	Vanilla	100%	✗	72.8	46.0	20.1	45.6	45.8	31.1	19.2	33.9	81.1	36.6
	Random Drop	25.0%	✗	41.6	25.2	10.6	27.4	36.1	21.1	12.7	26.3	**79.3**	22.4
	ResAdapt (Ours)	16.1%	✗	**64.4**	**37.0**	**15.9**	**39.8**	**40.6**	**26.7**	**15.7**	**30.0**	76.8	**33.3**
	Random Drop	10.0%	✗	32.6	19.0	7.8	21.9	33.5	18.6	11.5	24.8	76.9	19.9
	ToMe (Bolya et al., 2023)	10.0%	✗	61.6	33.8	13.3	**38.1**	**42.4**	**27.6**	**16.6**	**31.4**	**77.4**	31.5
	FixedScale	12.3%	✗	**61.7**	**34.9**	**14.7**	**38.1**	39.9	26.2	15.3	29.5	75.4	32.6
	FixedScale	6.3%	✗	53.7	28.2	11.8	33.6	37.9	24.3	14.3	28.1	73.0	39.1†
	ResAdapt (Ours)	6.8%	✗	54.3	28.0	11.7	33.7	38.3	24.5	14.4	28.4	73.2	**43.9**

Table 8: Zero-shot transfer to image benchmarks. Parenthetical values denote per-task retention ratio R, and ResAdapt-RL additionally fine-tunes the MLLM via RL.

Model	MathVista testmini	MMMU val	OCRBench	ChartQA	AI2D	TextVQA val
Qwen2.5-VL-7B	49.1(100%)	50.9(100%)	84.2(100%)	83.9(100%)	82.5(100%)	82.9(100%)
Random Drop	44.8(50%)	49.0(50%)	74.8(50%)	71.6(50%)	80.3(50%)	78.1(50%)
ToMe (Bolya et al., 2023)	46.2(50%)	49.6(50%)	79.3(50%)	78.1(50%)	81.9(50%)	81.2(50%)
VisionZip (Yang et al., 2025a)	47.2(50%)	48.6(50%)	79.6(50%)	77.9(50%)	81.9(50%)	81.3(50%)
ResAdapt(Qwen2.5-VL-7B)	45.5(42%)	51.0(29%)	80.0(64%)	85.9(105%)	81.4(41%)	69.6(30%)
ResAdapt-RL(Qwen2.5-VL-7B)	46.7(42%)	50.9(29%)	80.8(64%)	86.6(105%)	81.1(41%)	70.1(30%)
Qwen3-VL-8B	56.1(100%)	53.4(100%)	85.0(100%)	84.0(100%)	83.5(100%)	82.1(100%)
Random Drop	47.3(50%)	48.7(50%)	62.9(50%)	70.2(50%)	79.7(50%)	76.6(50%)
VisionZip (Yang et al., 2025a)	47.8(50%)	50.3(50%)	70.5(50%)	75.0(50%)	80.5(50%)	79.3(50%)
ToMe (Bolya et al., 2023)	49.6(50%)	50.6(50%)	70.3(50%)	75.2(50%)	80.5(50%)	79.4(50%)
ResAdapt(Qwen3-VL-8B)	52.5(42%)	50.9(29%)	82.7(64%)	83.2(105%)	81.2(41%)	67.8(30%)

E.1.3 Transfer Beyond Video

Table 8 applies the learned video policy to static image benchmarks without retraining. The policy separates images that warrant extra fidelity, spending 105% retention on ChartQA, from those it compresses to 29–42%; text-dense tasks transfer only at operating points that preserve fine detail.

$s_{\min}$ controls the fidelity–budget trade-off. Table 9 compares two operating points for each OCR task. The default video-trained policy supplies the accuracy and realized retention R reported in Table 8; the raised-floor policy supplies a second accuracy and its mean-scale budget proxy R_{mean}. As noted in Table 9, for configurations where $s_{\min}$ is raised to 0.50 or 0.70, the average predicted scale exceeds native resolution ($\bar{s} \geq 1.0$), so $R_{\text{mean}} = \bar{s}^2 \gtrsim 100\%$, indicating an expanded visual token allocation. Raising $s_{\min}$ recovers text-dense accuracy: on Qwen3-VL, TextVQA rises from 67.8 to 83.1 at $s_{\min}$=0.85.

Table 9: Scale-floor trade-off on OCR-class tasks. Each row compares the default video-trained policy with a raised-floor operating point. The default columns report accuracy and realized retention R from Table 8; the raised-floor columns report accuracy and the logged mean-scale budget proxy $R_{\text{mean}} = \bar{s}^2$ (Eq. 33).

Model	Task	Default policy		Raised scale floor		
		Acc.	R	$s_{\min}$	Acc.	R_{mean}
Qwen2.5-VL-7B	OCRBench	80.0	64%	0.70	**87.70**	$\gtrsim 100\%$
Qwen2.5-VL-7B	TextVQA	69.6	30%	0.70	**83.80**	$\gtrsim 100\%$
Qwen3-VL-8B	OCRBench	82.7	64%	0.50	**87.60**	$\gtrsim 100\%$
Qwen3-VL-8B	TextVQA	67.8	30%	0.85	**83.10**	$\approx 180\%$ ($\bar{s} \approx 1.34$)

E.1.4 Extended Latency Breakdown

Latency measurements use the hardware configuration in Table 3, with batch size 1 and at most 256 output tokens per response. Scale Time includes query and frame encoding, scale prediction, and resizing. Backbone Time covers the subsequent backbone inference, including visual encoding, prefill, and decoding. Reported end-to-end time is the sum of these two sequential stages. Table 10 breaks latency down across 16, 32, 64, and 128 frames on Qwen2.5-VL-7B. Scale prediction itself is nearly frame-independent at 86–97 ms, while the Allocator's own visual encoding grows with frame count and the backbone's generation cost grows faster still. That asymmetry sets where the front end pays for itself: at conservative retention it adds 9.2% end-to-end at 16 frames but removes 33.2% at 128 frames.

Table 10: Latency across 16–128 frames on Qwen2.5-VL-7B (ms, ↓). The protocol matches Table 3: H100 GPUs, bf16, batch size 1, vLLM 0.8.1, and 200 runs after 5 warm-ups. E2E = Scale Time + Backbone Time.

#Frames	Method	Retention Ratio R	Scale						Inference		Total
			TFLOPs	Text Enc.	Visual Enc.	Scale Pred.	Scale Apply	Scale Time	TFLOPs	Backbone Time	E2E Time
16	Vanilla	100%	–	–	–	–	–	–	111.4	527.9	527.9
	ResAdapt	76.3%	1.5	19.8	94.1	85.6	6.3	205.8	77.2 (↓30.7%)	370.7 (↓29.8%)	576.5 (↑9.2%)
	ResAdapt	52.8%	1.5	19.9	102.9	94.5	8.4	225.7	51.5 (↓53.8%)	313.1 (↓40.7%)	538.8 (↑2.1%)
	ResAdapt	28.9%	1.5	20.4	103.4	92.2	9.0	225.0	31.0 (↓72.2%)	237.9 (↓54.9%)	462.9 (↓12.3%)
32	Vanilla	100%	–	–	–	–	–	–	222.5	881.9	881.9
	ResAdapt	74.4%	2.9	19.9	204.1	97.4	14.4	335.9	153.9 (↓30.8%)	627.6 (↓28.8%)	963.5 (↑9.2%)
	ResAdapt	51.5%	2.9	20.0	193.2	92.0	16.2	321.4	102.4 (↓54.0%)	467.1 (↓47.0%)	788.5 (↓10.6%)
	ResAdapt	28.2%	2.9	20.3	190.4	90.3	17.3	318.3	61.4 (↓72.4%)	332.6 (↓62.3%)	650.9 (↓26.2%)
64	Vanilla	100%	–	–	–	–	–	–	444.6	2059.6	2059.6
	ResAdapt	73.2%	5.8	19.8	389.5	95.8	26.4	531.5	307.3 (↓30.9%)	1327.0 (↓35.6%)	1858.5 (↓9.8%)
	ResAdapt	50.7%	5.8	20.1	382.1	94.9	29.9	527.0	204.3 (↓54.0%)	740.5 (↓64.0%)	1267.5 (↓38.5%)
	ResAdapt	27.8%	5.8	20.0	371.6	90.2	34.8	516.6	122.2 (↓72.5%)	511.4 (↓75.2%)	1028.0 (↓50.1%)
128	Vanilla	100%	–	–	–	–	–	–	888.9	4877.0	4877.0
	ResAdapt	74.2%	11.6	20.1	766.3	95.0	53.1	934.5	614.1 (↓30.9%)	2323.6 (↓52.4%)	3258.1 (↓33.2%)
	ResAdapt	51.4%	11.6	20.2	755.3	93.8	59.4	928.7	408.0 (↓54.1%)	1496.0 (↓69.3%)	2424.7 (↓50.3%)
	ResAdapt	28.2%	11.6	20.4	734.5	92.0	68.6	915.5	243.9 (↓72.6%)	1061.8 (↓78.2%)	1977.3 (↓59.5%)

E.2 Behavioral Analysis of the Learned Policy

Figure 5 shows representative per-frame profiles. The rest of this subsection aggregates the same behavior across benchmarks, clip durations, and task types.

E.2.1 Global Allocation Statistics

Figure 6 shows similar aggregate scales across sampled positions, while the heatmap reveals localized high-resolution frames within individual videos. The policy varies fidelity within clips rather than consistently favoring one position on the timeline.

E.2.2 Benchmark-Level Budget Allocation

Figure 7 reveals a benchmark-level hierarchy that the policy was never told about, since benchmark labels are absent during training: VideoMMMU Adaptation occupies the highest-scale region and VideoMME the lowest. The operating point is thus a function of the input and the query, not a fixed compression rule.

E.2.3 Long-Context and Semantic Structure

The duration analysis in Figure 8 uses an earlier checkpoint with mean scale $\bar{s} \approx 0.337$ and VideoMME accuracy 57.96%; Table 1 reports the calibrated checkpoint at 60.3%. Within the duration analysis, mean scale decreases from 0.342 to 0.336 to 0.332 across short, medium, and long clips, while within-video variation rises from 0.085 to approximately 0.095. Longer clips thus receive lower average fidelity and more heterogeneous allocation at the same sampled-frame count.

Figure 9 shows the same effect at the level of semantics rather than duration. Within a single benchmark, the policy grants larger budgets to *Sports Competition* than to *Artistic Performance*, adding category-level structure to the aggregate retention ratios in the main table.

E.2.4 Selectivity and Success

The Gini coefficient measures how unevenly scales are distributed across frames. Figure 10 reports two-sided Mann–Whitney U tests for VideoMME ($p = 0.2762$), MMMU-P ($p = 0.02979$), and LongVideoBench ($p = 0.1046$). MMMU-P shows a nominal association between higher selectivity and correct predictions at the unadjusted 0.05 threshold. The permutation experiment separately tests the effect of fidelity placement.

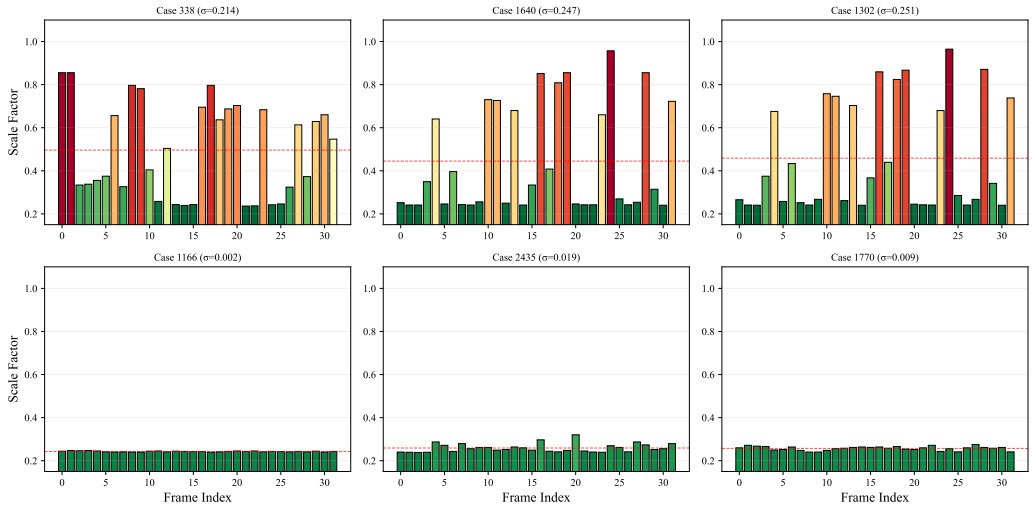


Figure 5: Representative VideoMME allocation profiles. High-dispersion examples (top) concentrate fidelity on a few frames, whereas low-dispersion examples (bottom) remain nearly uniform. Red dashes mark the mean scale for each video.

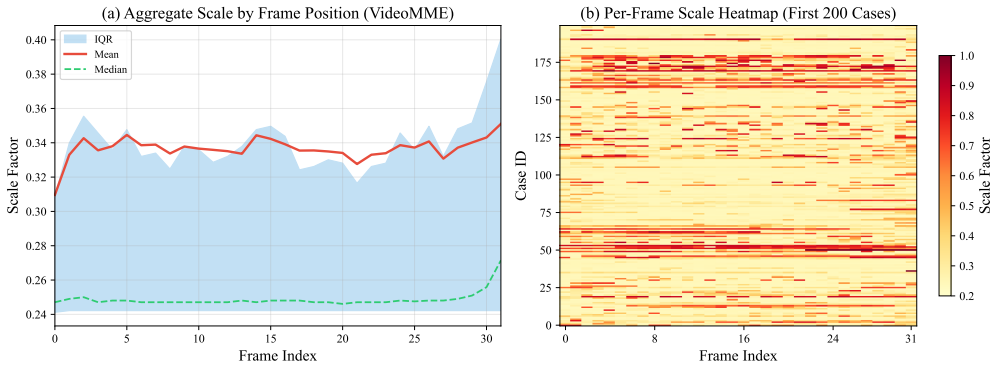


Figure 6: Allocation across frame positions on VideoMME. Left: mean, median, and interquartile range of scales at each sampled position. Right: per-frame scales for the first 200 examples.

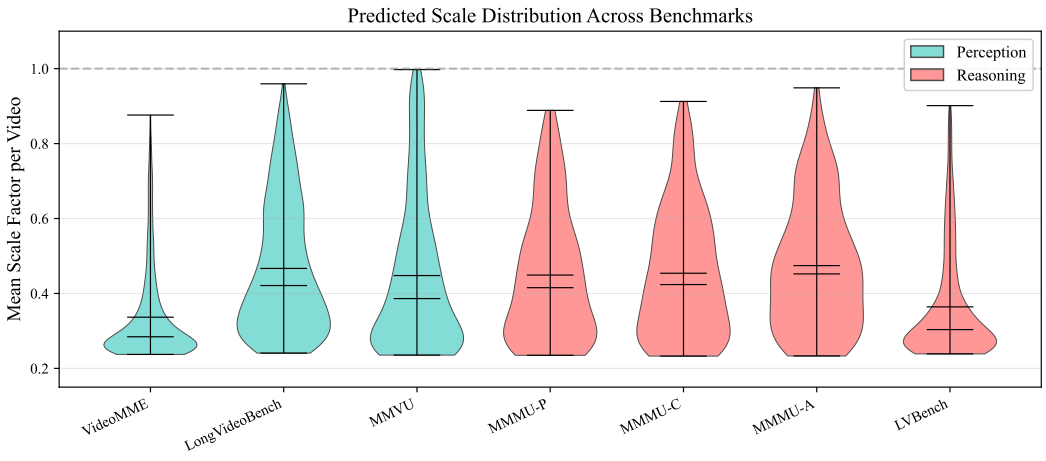


Figure 7: Per-video mean scale across benchmarks. Kernel density estimates trace the per-video mean scale $\bar{s}$ and reveal benchmark-dependent operating regions.

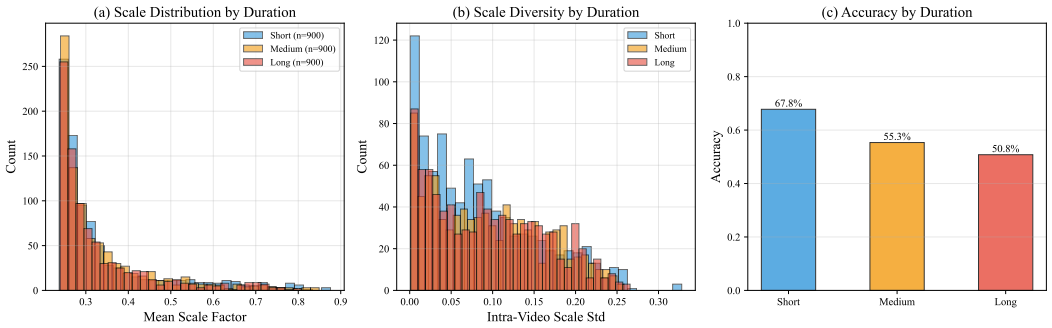


Figure 8: Allocation by VideoMME clip duration. Longer clips receive a lower mean scale and greater within-video scale variation, alongside lower accuracy.

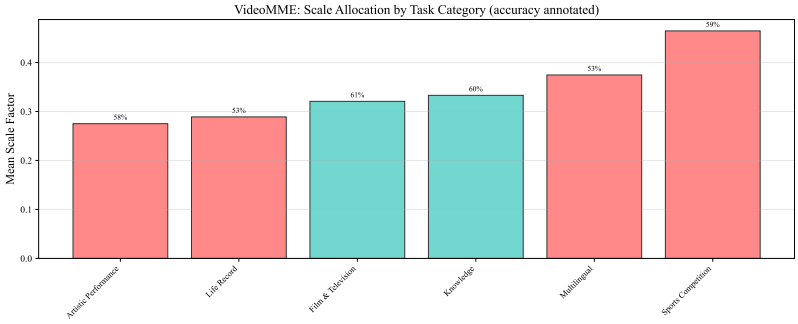


Figure 9: Scale allocation by VideoMME task category. Mean $\bar{s}$ varies across categories, with Sports Competition receiving higher allocations than Artistic Performance.

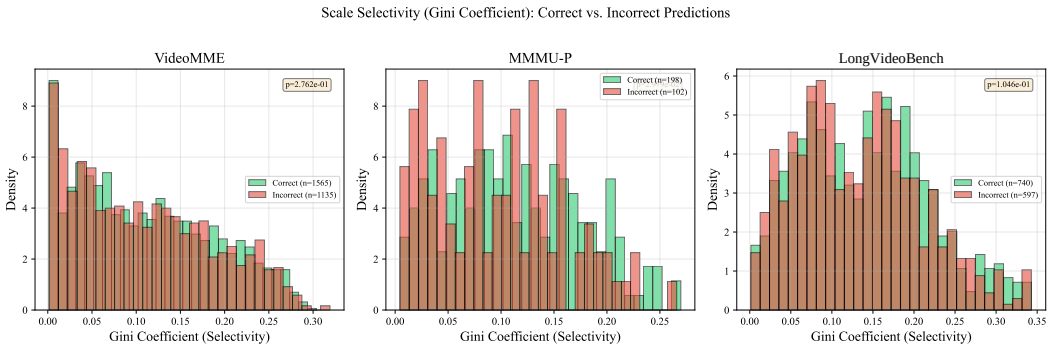


Figure 10: Selectivity and prediction correctness. Gini coefficients of predicted frame scales measure allocation selectivity on VideoMME, MMMU-P, and LongVideoBench. Embedded p-values use two-sided Mann–Whitney U tests.

E.3 Ablation Studies

Table 11: CAPO latent-family ablation. This study uses ablation checkpoints and a dataset distinct from those used for Table 1. Replacing the Beta policy with a Gaussian changes the four aggregate scores by at most 0.8 points; VideoMMMU category scores vary more. $\bar{s}$ is the unweighted mean predicted scale on the ablation dataset. **P/C/A**: VideoMMMU perception, comprehension, and adaptation.

Variant	$\bar{s}$	V-MME	LongVB	MMVU	P	C	A	LVBench
Beta-CAPO	0.54	60.3	**58.2**	**51.9**	65.0	**54.3**	28.7	**37.9**
Gaussian-CAPO	0.60	**61.0**	57.4	51.8	**66.0**	50.0	**30.3**	37.2

E.3.1 Query-Conditioning Ablation

Table 12 separates query-conditioned allocation from content-only adaptation. The no-query variant replaces query tokens with a fixed null embedding during training and evaluation, while the shuffled-query control keeps the trained model fixed and pairs each test video with another video's query. Crucially, the shuffled-query control only substitutes the query fed into the Allocator for scale prediction, while the downstream backbone still receives the original, correct query alongside the adapted video frames; this isolates the effect of misallocating spatial fidelity without confounding downstream language comprehension. Every variant is calibrated toward the same validation retention, after which its scale multiplier is frozen for test evaluation. The no-query policy ends up spending a slightly larger test budget (25.4% versus 23.8%) yet loses accuracy, dropping the six-benchmark mean from 52.87 to 52.43; shuffled queries realize 22.3% and drop it to 51.77. Query conditioning improves accuracy over content-only adaptation.

Table 12: Query-conditioning ablation on Qwen2.5-VL-7B with 32 frames. Variants share the Allocator architecture and are calibrated on validation data toward $R{\approx}23.8\%$.

Variant	Retention Ratio R	VideoMME	MMVU	VideoMMMU	LongVideoBench	LVBench	MLVU	Avg
ResAdapt (Ours)	23.8%	60.3	51.9	48.8	58.2	37.9	60.1	52.87
w/o query (train and test)	25.4%	60.2	51.6	47.5	57.6	37.7	60.0	52.43
shuffled query (test only)	22.3%	59.8	51.2	46.4	56.5	37.2	59.5	51.77

E.3.2 Temporal Similarity Ablation

We next isolate how temporal-similarity regularization reshapes allocation structure and accuracy.

Without $\mathcal{L}_{\text{sim}}$, the mean within-video $\text{Std}_t(s_t)$ lies between roughly 0 and 0.045 depending on the benchmark; adding the regularizer raises it to 0.091–0.133 (Figures 11 and 12). Native retention is 23.8% with the regularizer and 22.9% without it (Table 13); Table 14 compares accuracy after bringing their budgets closer.

Figure 13 shows the same change in the global histogram, per-video range, adjacent-frame variation, and Gini coefficient.

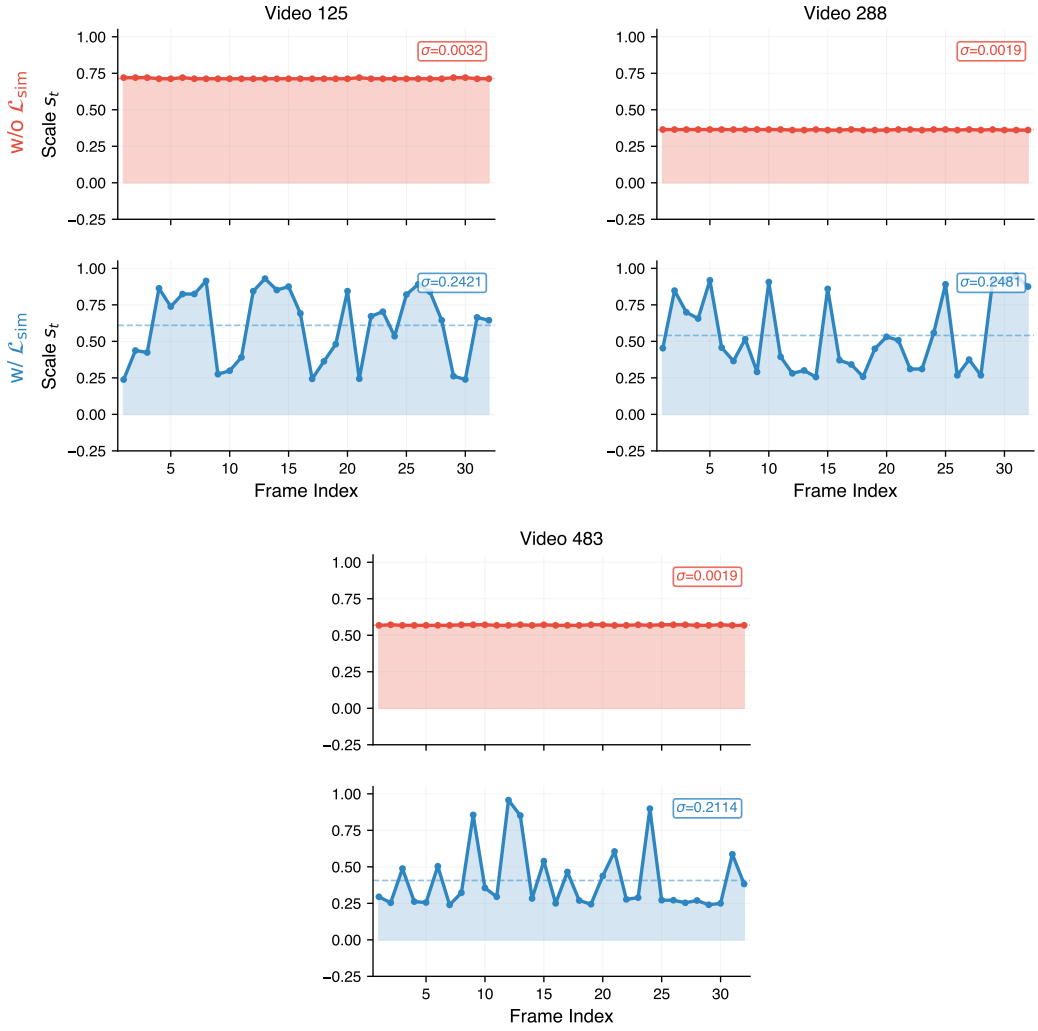


Figure 11: Per-frame effect of $\mathcal{L}_{\text{sim}}$. Each video shows scales without regularization (red, top) and with regularization (blue, bottom). Regularization replaces nearly uniform scales with frame-specific allocations.

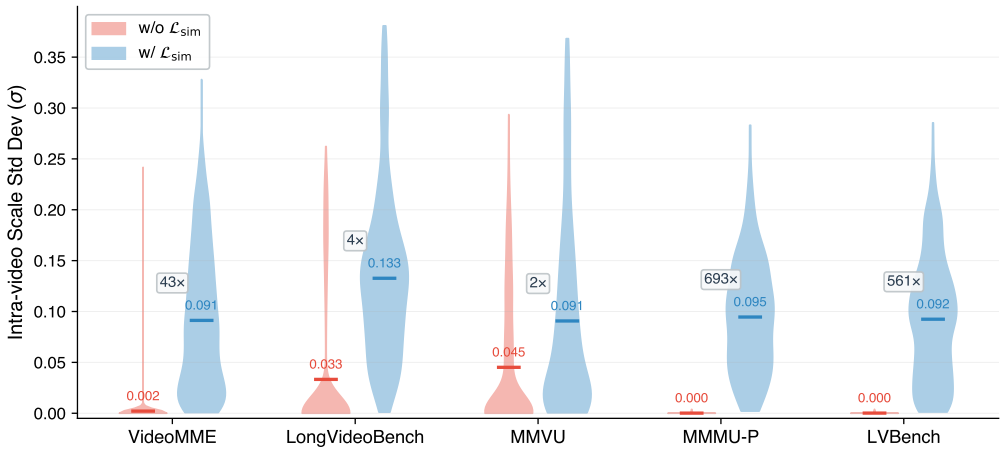


Figure 12: Scale diversity with and without $\mathcal{L}_{\text{sim}}$. Across five benchmarks, temporal regularization increases the mean per-video standard deviation $\text{Std}_t(s_t)$.

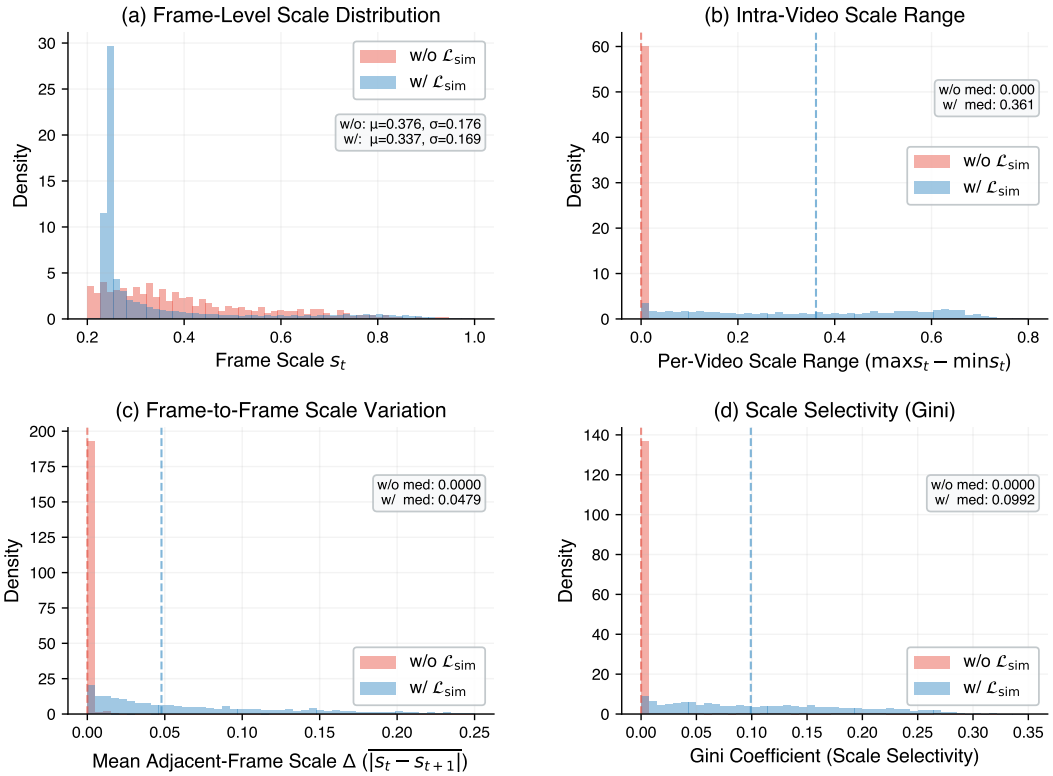


Figure 13: $\mathcal{L}_{\text{sim}}$ diagnostics on VideoMME. Regularization produces a bimodal scale distribution with larger per-video ranges, adjacent-frame variation, and Gini coefficients.

E.3.3 Reward Design Ablation

An allocation objective determines both the global operating budget and its distribution across frames. In Table 11, Beta-mean and Gaussian-sampling inference differ by at most 0.8 points on the four aggregate benchmark scores, although the Gaussian learns a higher mean scale ($\bar{s} = 0.60$ versus 0.54). The following controls examine how cost shaping affects budget selection and accuracy. Figure 14 tracks the per-sample scale range and Figure 15 the operating point each objective converges to, with rolling-mean smoothed curves ($w = 5$) over translucent raw trajectories.

Tables 13 and 14 report different evaluation protocols for the same objectives: native outputs before scale calibration, then evaluation after validation-set budget calibration. The direct-cost baseline replaces CAPO's asymmetric shaping ($S_{m,n}$) and positive floor (ε_+) in Eq. (7) with an unconditioned linear cost penalty: $A_{m,n} = A^{\text{base}}_{m,n} - \lambda_{\text{cost}} c_m$, with $\lambda_{\text{cost}} = 0.5$. Because the penalty suppresses budget indiscriminately regardless of whether an allocation successfully solves the query, it drives the policy gradient to collapse scales toward $s_{\min}$ (R=9.4%, losing 3.49 mean points relative to full CAPO). The no-cost variant removes explicit cost terms from the advantage (λ_{capo}=0, γ=0), optimizing solely on $A_{m,n} = A^{\text{base}}_{m,n}$ while retaining $\mathcal{L}_{\text{sim}}$; with explicit cost penalties removed, it drifts to R=67.2%. Removing $\mathcal{L}_{\text{sim}}$ lowers native retention from 23.8% to 22.9% and the mean score by 0.80 points.

Table 13: CAPO objectives before budget calibration. Qwen2.5-VL-7B with 32 frames, evaluated at each policy's native scale.

Training objective	Retention Ratio R	VideoMME	MMVU	VideoMMMU	LongVideoBench	LVBench	MLVU	Mean
CAPO + $\mathcal{L}_{\text{sim}}$	23.8%	60.3	51.9	48.8	58.2	37.9	60.1	52.87
CAPO w/o $\mathcal{L}_{\text{sim}}$	22.9%	60.0	51.3	46.8	57.1	37.4	59.8	52.07
Direct linear cost + $\mathcal{L}_{\text{sim}}$	9.4%	57.6	47.5	43.8	55.1	35.3	57.0	49.38
No cost + $\mathcal{L}_{\text{sim}}$	67.2%	61.4	52.3	49.1	58.5	38.2	62.1	53.60

Table 14 then calibrates every objective toward the full method's retention ($R \approx 23.8\%$). Budget calibration is conducted on the 500-sample VideoAuto-R1 validation split. We perform a two-stage search over scalar multipliers $k \in [0.5, 3.0]$ applied to predicted scales $\boldsymbol{s} \leftarrow \text{clip}(k\boldsymbol{s}, s_{\min}, s_{\max})$: an initial grid with step size 0.05 identifies candidate ranges, followed by fine refinement with step size 0.01 until validation retention matches 23.8% within $\pm 0.5\%$. The resulting multiplier ($k = 1.65$ for direct-cost, $k = 0.60$ for no-cost, and $k = 1.02$ for w/o $\mathcal{L}_{\text{sim}}$) is then frozen for test evaluation across all six benchmarks. For Gaussian policies (Table 11), scales sampled from $\mathcal{N}(\mu_t, \sigma_t^2)$ are truncated to $[s_{\min}, s_{\max}]$ at test time. At these approximately comparable budgets, full CAPO exceeds the no-$\mathcal{L}_{\text{sim}}$, direct-cost, and cost-free variants by 0.64, 1.30, and 0.85 mean points. The $\mathcal{L}_{\text{sim}}$ gap should be read with the native budget change in Table 13: the regularizer alters both temporal structure and overall retention.

Table 14: CAPO objectives after budget calibration. Qwen2.5-VL-7B, 32 frames. A validation multiplier targets $R \approx 23.8\%$ and is frozen at test time.

Training objective	Retention Ratio R	VideoMME	MMVU	VideoMMMU	LongVideoBench	LVBench	MLVU	Mean
CAPO + $\mathcal{L}_{\text{sim}}$	23.8%	60.3	51.9	48.8	58.2	37.9	60.1	52.87
CAPO w/o $\mathcal{L}_{\text{sim}}$	23.7%	60.1	51.4	47.2	57.3	37.5	59.9	52.23
Direct linear cost + $\mathcal{L}_{\text{sim}}$	22.8%	59.6	50.9	46.2	56.4	37.0	59.3	51.57
No cost + $\mathcal{L}_{\text{sim}}$	23.5%	60.0	51.3	46.9	56.9	37.3	59.7	52.02

Figures 14 and 15 visualize the scale variation and operating points summarized above.

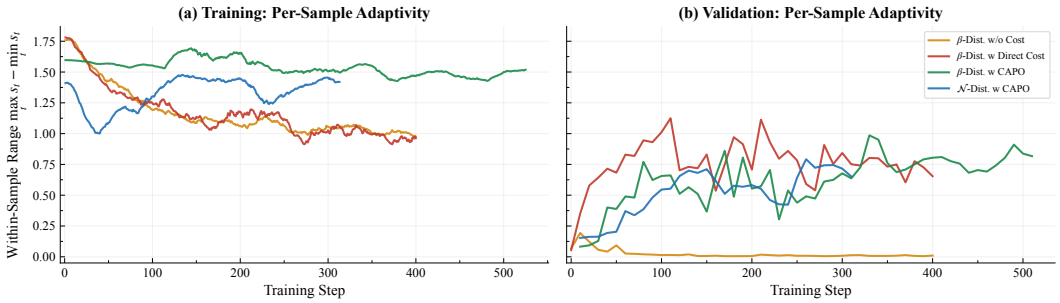


Figure 14: Per-sample adaptivity across reward designs. The within-sample range $\max_t s_t - \min_t s_t$ is tracked on **(a)** training and **(b)** validation data. CAPO maintains variation; direct cost collapses and cost-free training saturates.

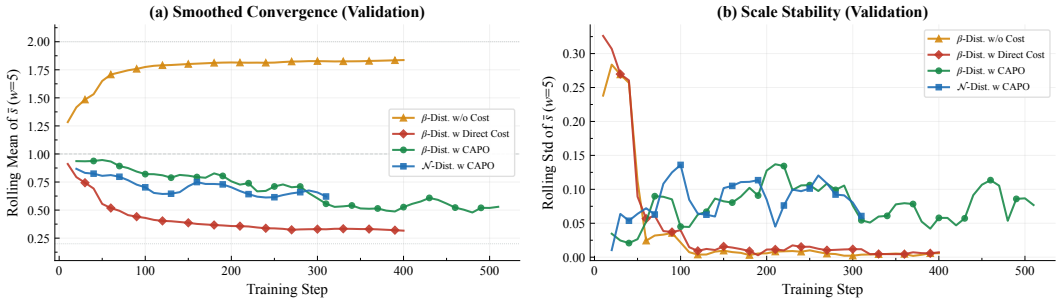


Figure 15: Validation mean predicted scale across reward designs. Curves display rolling means ($w = 5$) over validation steps. CAPO stabilizes at an interior operating point ($\bar{s} \approx 0.54$), whereas cost-free training expands toward the upper scale bound ($s_{\max} = 1.8$) and direct linear cost collapses toward the lower bound ($s_{\min} = 0.2$, reaching $\bar{s} \approx 0.30$). Test retention R is in Tables 4 and 13.

E.3.4 Operator Generalization

We test whether scales learned for resizing can rank frames for an unseen discrete operator. The Allocator scores 128 candidates in every variant. *Top-K Select* keeps the K highest-scoring frames at full resolution; *Threshold Select* keeps every frame above a fixed score, so its frame count varies by video; *Top-$2K$ Select + Resize* keeps twice the target frame count and then applies the predicted scales. Because the adaptive variants retain different numbers of frames, Table 15 reports frame count and mean retention R together.

Hard selection alone falls below uniform sampling at the same nominal frame count. Pairing selection with resizing performs best on two of four benchmarks at both 16 and 32 frames. At 32 frames it reaches 62.5 on VideoMME and 40.0 on LVBench, versus 62.0 and 38.6 for uniform dense input; dense input leads on LongVideoBench and MMVU. Retaining more candidate frames before resizing preserves temporal context while transferring the learned scales to frame ranking.

Table 15: Zero-shot transfer to frame selection. Combining top-K selection with resizing from 128 candidates leads on two of four benchmarks at both 16 and 32 frames. The 32-frame dense reference follows Table 1. Frame-equivalent budgets for adaptive selection are normalized to the 128-frame candidate pool.

Method	VideoMME	LongVideoBench	LVBench	MMVU
Budget: 8 frames				
Vanilla	54.0	53.9	33.3	48.9
Top-8 Select	52.2	51.1	32.0	49.2
Budget: 16 frames				
Vanilla	58.9	56.0	36.1	50.9
Threshold Select	58.0	57.4	36.4	51.0
Avg. Budget (Retention Ratio)	12.2f (9.5%)	23.2f (18.1%)	16.7f (13.0%)	17.2f (13.4%)
Top-32 Select + Resize	60.6	57.2	38.9	50.2
Avg. Budget (Retention Ratio)	11.7f (9.1%)	16.9f (13.2%)	13.7f (10.7%)	14.1f (11.0%)
Budget: 32 frames				
Vanilla	62.0	58.9	38.6	52.7
Top-32 Select	59.7	55.7	37.0	51.2
Top-64 Select + Resize	62.5	58.4	40.0	52.3
Avg. Budget (Retention Ratio)	23.8f (18.6%)	36.2f (28.3%)	24.1f (18.8%)	32.5f (25.4%)

E.4 Additional Comparisons and Robustness

E.4.1 Closest Input-Side Adaptive Baselines

Table 2 in the main text compares query-aware hard frame selection and slow–fast routing with ResAdapt under a 128→32 frame budget.

E.4.2 Extended Cross-Family Transfer

Table 16 transfers a Qwen2.5-VL-trained Allocator, without retraining, to a Mixture-of-Experts backbone (Kimi-VL) and an RL-tuned backbone (MiMo-VL). At 128 frames and ∼23% retention it exceeds full-frame vanilla on the four-benchmark average for both families, raising Kimi-VL from 56.26 to 57.05 and MiMo-VL from 54.94 to 55.73. Table 17 compares ResAdapt with ToMe, Random Drop, and FixedScale near its higher-retention operating points. ResAdapt achieves the strongest compressed average on both backbones, with the largest task-level margin on VideoMMMU for 128-frame MiMo-VL.

Table 16: Zero-shot transfer across MLLM families. A Qwen2.5-VL-trained Allocator is applied to Kimi-VL-A3B-Instruct and MiMo-VL-7B-RL without retraining. **Avg**: four-benchmark mean; **Bold**: compressed average exceeding its same-frame vanilla counterpart.

Backbone	Method	Frames	Retention Ratio R	Video Benchmark				Avg
				LongVideoBench	LVBench	VideoMMMU	VideoMME	
Kimi-VL	Vanilla	32	100%	62.68	44.74	52.33	65.81	56.39
	ResAdapt (Ours)	32	23.8%	62.0	44.0	51.6	64.0	55.40
	ResAdapt (Ours)	32	11.4%	59.2	42.1	48.5	63.2	53.25
	Vanilla	128	100%	62.38	45.84	49.11	67.70	56.26
	ResAdapt (Ours)	128	22.9%	62.3	45.9	52.2	67.8	**57.05**
	ResAdapt (Ours)	128	11.1%	60.7	43.7	50.4	66.2	55.25
MiMo-VL	Vanilla	32	100%	58.71	40.35	49.44	62.85	52.84
	ResAdapt (Ours)	32	23.8%	58.0	39.7	48.7	61.2	51.90
	ResAdapt (Ours)	32	11.4%	55.3	37.6	45.6	60.2	49.68
	Vanilla	128	100%	61.26	45.19	47.67	65.63	54.94
	ResAdapt (Ours)	128	22.9%	61.2	45.3	50.9	65.5	**55.73**
	ResAdapt (Ours)	128	11.1%	59.6	43.1	49.0	64.1	53.95

E.4.3 Agentic Tool-Use Compatibility

ResAdapt predicts scales in a single pass and can also be applied within a multi-turn agentic pipeline. In the LongVT (Yang et al., 2025c) harness (Table 18), adding the Allocator raises answer accuracy

Table 17: Cross-family transfer with approximately budget-matched controls. R is mean realized visual-token retention. The upper block for each backbone compares compressed methods near ResAdapt's operating point; lower-retention rows provide additional reference points. **Bold**: best compressed result in each column of the matched block.

Backbone	Frames	Method	R	LongVideoBench	LVBench	VideoMMMU	VideoMME	Avg.
Kimi-VL	32	Vanilla	1.00	62.68	44.74	52.33	65.81	56.39
		ResAdapt ($s_{\min}$=0.75)	0.83	**63.65**	44.22	**52.44**	**66.00**	**56.58**
		ToMe (Bolya et al., 2023)	0.82	61.88	44.05	51.92	65.28	55.78
		Random Drop	0.81	61.72	43.91	51.48	64.97	55.52
		FixedScale	0.84	62.41	**44.48**	52.18	65.72	56.20
		ToMe	0.50	58.79	42.09	50.89	63.56	53.83
		Random Drop	0.50	59.54	41.77	49.00	62.52	53.21
MiMo-VL	128	Vanilla	1.00	61.26	45.19	47.67	65.63	54.94
		ResAdapt ($s_{\min}$=0.75)	0.80	**62.30**	**46.29**	**51.78**	**65.59**	**56.49**
		ToMe	0.79	60.48	44.62	47.11	65.02	54.31
		Random Drop	0.80	60.18	44.27	47.40	64.70	54.14
		FixedScale	0.81	61.08	45.02	48.15	65.38	54.91
		Random Drop	0.20	54.30	39.25	46.67	59.81	50.01

and format adherence by 2.7 and 12.4 points on LVBench, and by 9.4 and 25.9 points averaged over three benchmarks and five sub-tasks. The table reports the mean-scale area proxy $R_{\text{mean}} = \bar{s}^2 \approx 0.87$, computed from logged average scale $\bar{s} \approx 0.93$ across inspection passes.

Table 18: Compatibility with the LongVT agentic harness (Yang et al., 2025c). The Allocator improves answer accuracy and format adherence at a mean-scale area proxy $R_{\text{mean}} = \bar{s}^2 \approx 0.87$, with $\bar{s}$ averaged across agentic inspection passes. Three-benchmark rows average LongVideoBench, VideoMMMU, and VideoMME over five sub-tasks.

Setting	Task scope	Avg. Acc.	Avg. Format	Mean scale $\bar{s}$	Area proxy R_{mean}
Baseline	LVBench (single)	22.66	54.13	–	–
+ ResAdapt	LVBench (single)	**25.37**	**66.52**	0.933	0.871
Baseline	3 benchmarks (5 sub-tasks)	20.71	35.79	–	–
+ ResAdapt	3 benchmarks (5 sub-tasks)	**30.08**	**61.64**	0.930	0.865

E.4.4 Pixel-Level Detail Stress Test (VCR)

Visual Caption Restoration (VCR) (Zhang et al., 2024b) tests whether a model can recover partially obscured text from visual and linguistic context. We evaluate Qwen2.5-VL-7B using the 32-frame construction in Table 19, representing each target image as a repeated-frame sequence. This probes preservation of the fine visual cues needed to restore text. ResAdapt leads the low-retention block and trails FlashVid by 0.0082 Jaccard at high retention. Raising $s_{\min}$ recovers fidelity; every compressed setting remains below vanilla.

Table 19: Visual Caption Restoration (VCR) (Zhang et al., 2024b) with Qwen2.5-VL-7B under the 32-frame protocol in Appendix E.4.4 (mean Jaccard ↑). Rows are ordered by retention. **Bold**: best compressed result in each budget block.

Method	Retention Ratio R	Mean Jaccard
Random (keep=0.2)	0.20	0.2515
ToMe (Bolya et al., 2023) (keep=0.2)	0.20	0.3772
FlashVid (Fan et al., 2026) (keep=0.2)	0.23	0.4393
ResAdapt (default)	0.24	**0.4851**
Random (keep=0.75)	0.75	0.5367
ToMe (Bolya et al., 2023) (keep=0.75)	0.75	0.5783
ResAdapt ($s_{\min}$=0.5)	0.77	0.5857
FlashVid (Fan et al., 2026) (keep=0.75)	0.78	**0.5939**
Vanilla Qwen2.5-VL	1.00	0.6540

E.4.5 Placement and Content-Adaptive Controls

Training-free content-adaptive control. SimThresh reuses the Allocator's frozen encoder without its learned policy: frames whose local dissimilarity $1 - \mathrm{sim}_{\cos}(\boldsymbol{z}_t, \boldsymbol{z}_{t+1})$ exceeds a swept threshold receive $s_t{=}s_{\max}$ and the rest receive $s_t{=}s_{\min}$. ResAdapt attains the highest six-benchmark mean at both retention tiers. Permuting its high-resolution assignments at the same mean retention lowers all six scores.

Table 20: Content-only adaptation and budget permutation on Qwen2.5-VL-7B with 32 frames. SimThresh raises the scale when adjacent-frame dissimilarity crosses a threshold. "Perm." shuffles ResAdapt frame-scale assignments at mean retention $R = 23.8\%$ (five-run mean; per-benchmark drops 1.3–2.4). **Bold**: best in each budget block.

Budget	Method	Retention Ratio R	VideoMME	MMVU	VideoMMMU	LongVideoBench	LVBench	MLVU
∼25%	FixedScale	25.0%	60.0	51.2	46.7	56.8	37.3	59.8
	SimThresh	26.4%	59.6	50.9	47.2	57.0	37.0	59.5
	SimThresh	24.1%	59.1	50.3	47.5	57.3	36.7	59.1
	ResAdapt (Ours)	23.8%	**60.3**	**51.9**	**48.8**	**58.2**	**37.9**	**60.1**
∼10%	FixedScale	12.3%	58.0	47.7	44.3	55.1	35.4	57.5
	SimThresh	11.8%	57.7	47.4	44.9	55.3	35.1	57.2
	SimThresh	10.9%	57.2	46.8	45.2	**55.6**	34.8	56.8
	ResAdapt (Ours)	11.4%	**59.4**	**49.2**	**45.7**	55.4	**35.9**	**58.4**
Perturbation probe (R=23.8%)	ResAdapt (unpermuted)	23.8%	60.3	51.9	48.8	58.2	37.9	60.1
	ResAdapt + budget perm.	23.8%	58.9	49.8	46.4	56.9	36.2	58.7

Composition with model-side compression. ResAdapt removes pixels before backbone encoding, whereas token merging acts on tokens the encoder has already produced, so the two are composable rather than mutually exclusive. At comparable combined budgets, applying ToMe (Bolya et al., 2023) after ResAdapt beats ToMe alone in 7 of 12 benchmark–budget cells and recovers up to 4.9 points on VideoMMMU at the ∼11% tier. Within both reported budget bands, ResAdapt alone outperforms the combined pipeline on all six benchmarks.

Table 21: Compatibility with model-side token merging (Qwen2.5-VL-7B, 32 frames). ToMe follows ResAdapt resized input at comparable combined retention R. **Bold**: best per column within each budget.

Combined R	Configuration	VideoMME	MMVU	VideoMMMU	LongVideoBench	LVBench	MLVU
∼24%	ResAdapt alone	**60.3**	**51.9**	**48.8**	**58.2**	**37.9**	**60.1**
	ToMe alone (s_t=1)	58.7	51.0	41.8	58.0	37.7	58.7
	ResAdapt + ToMe	59.8	51.3	47.6	57.9	37.4	59.6
∼11%	ResAdapt alone	**59.4**	**49.2**	**45.7**	**55.4**	**35.9**	**58.4**
	ToMe alone (s_t=1)	56.4	48.9	39.2	55.2	33.6	58.0
	ResAdapt + ToMe	58.1	48.6	44.1	54.7	34.9	57.3

E.4.6 Generalization Map

Table 16 tests zero-shot transfer across backbone families, Table 18 composition with an agentic pipeline, Tables 8 and 19 static images and pixel-level detail, and Figure 8 clip duration.

F Qualitative Case Studies and Visualizations

Task-dependent operating regimes. Figures 16 and 17 juxtapose two VideoMMMU tasks from the same educational domain that induce distinct allocation profiles. In the comprehension case (Figure 16), evidence sits sparsely on a few diagram slides, leading the policy to concentrate fidelity on those slides while coarsening uninformative intervals. Conversely, the adaptation case (Figure 17) requires parsing a dense numeric table across frames, prompting the policy to allocate higher spatial resolution throughout the table sequence.

Evidence localization and its failure mode. The success in Figure 18 illustrates precise allocation: the answer requires reading fleeting text overlays inside repetitive footage, and the policy assigns those instants higher resolution. Figure 19 illustrates a missed cue: the policy allocates more resolution to

Q: Evaluate five statements about Urban Geography City Models (concentric zone, Hoyt sector, multiple nuclei, galactic, Latin American); identify which are correct. *Please ignore the Quiz question in last frame of the video.*

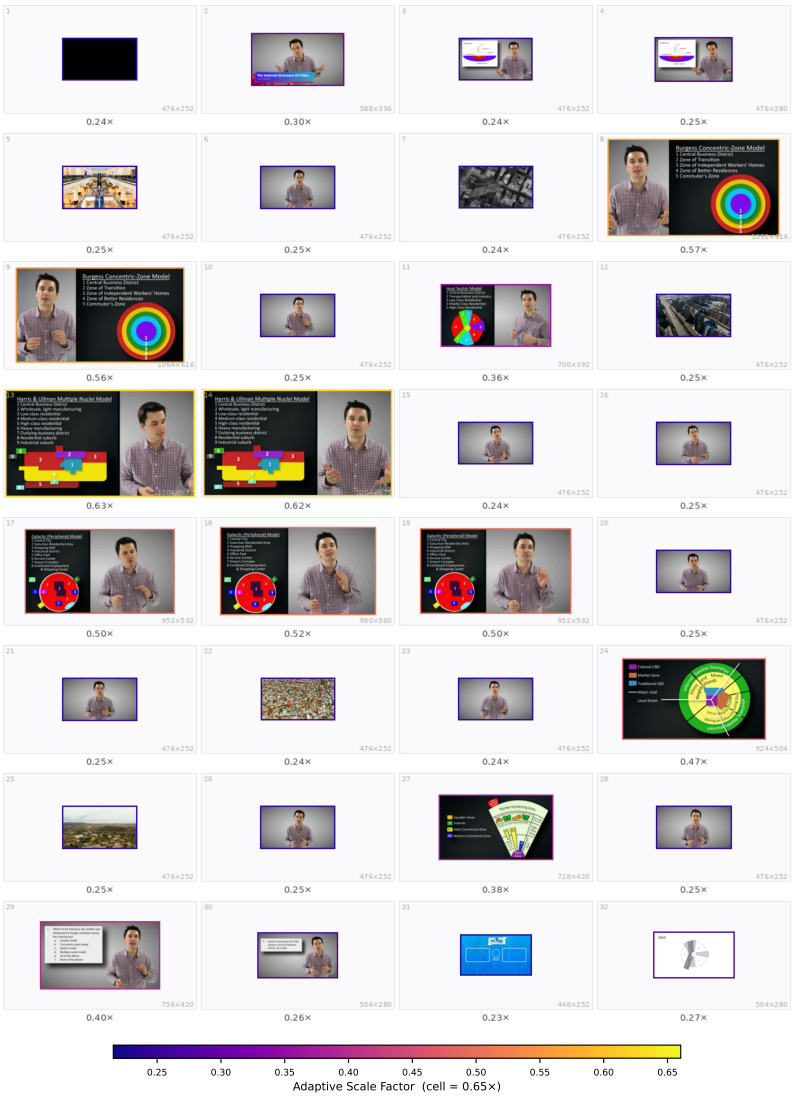


Figure 16: Case 1: VideoMMMU Comprehension (Hu et al., 2025) (Vanilla × → ResAdapt ✓). Each case plots 32 uniformly sampled frames at their assigned scale, with warmer borders for larger s_t. The policy concentrates resolution on diagram-bearing slides, compresses lecturer-only frames, and suppresses the final quiz frame designated as irrelevant by the prompt.

an adjacent frame while compressing the frame that contains the fork. These cases show how fidelity placement can preserve or obscure brief evidence.

Q: Watch and learn the video content. Then apply what you learned to answer: Table 11.47 provides a survey of the youngest online entrepreneurs (ages 17–30) whose net worth $\geq$ \$1M. We want to know whether ages and net worth are independent. χ^2 test statistic = _____



Figure 17: Case 2: VideoMMMU Adaptation (Hu et al., 2025) (Vanilla $\times$ $\rightarrow$ ResAdapt $\checkmark$). Reading a dense numeric table for a χ^2 computation, the policy assigns relatively higher resolution to table-bearing frames within a sub-native scale range.

Q: When is the zodiacal light visible from the video? (A) Mar. 19, (B) Mar. 24, (C) Mar. 25, (D) Mar. 29.

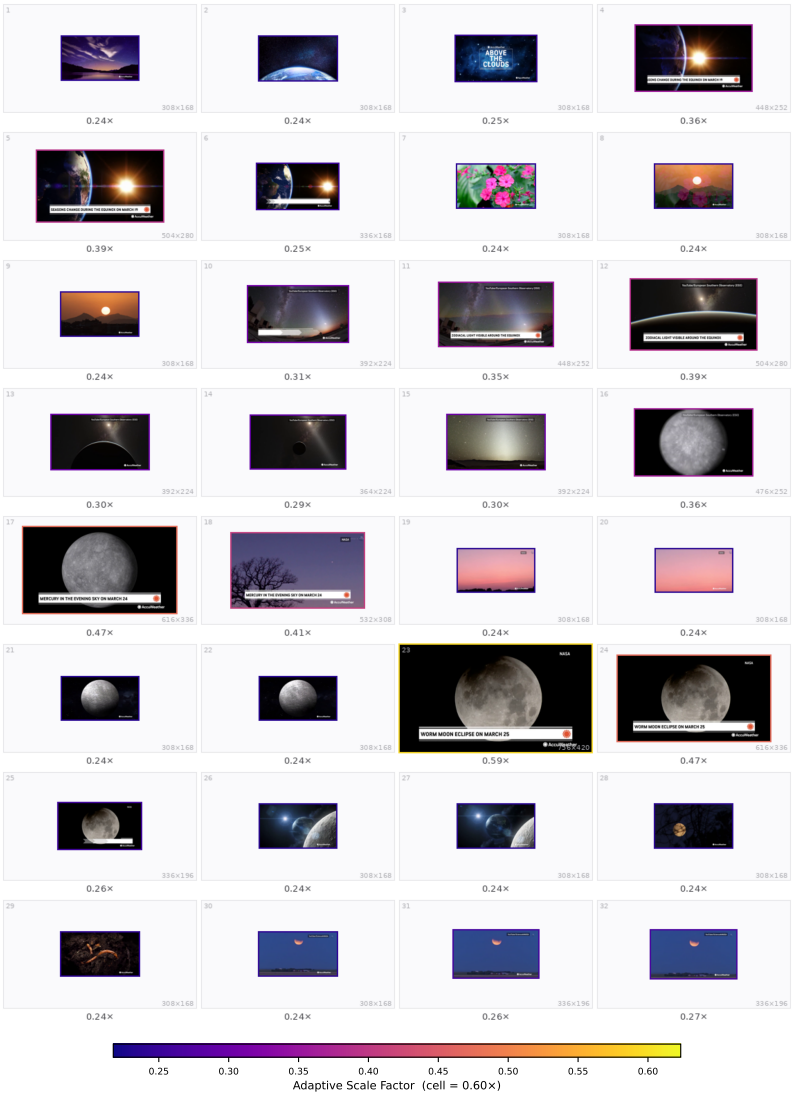


Figure 18: Case 3: VideoMME (Fu et al., 2025) (Vanilla × → ResAdapt ✓). The policy enlarges frames with the decisive date overlays and compresses the surrounding sky footage.

Q: Which item does the man throw into the trash at the beginning of the video? (A) A fork, (B) A pair of chopsticks, (C) A box of noodles, (D) A spoon.

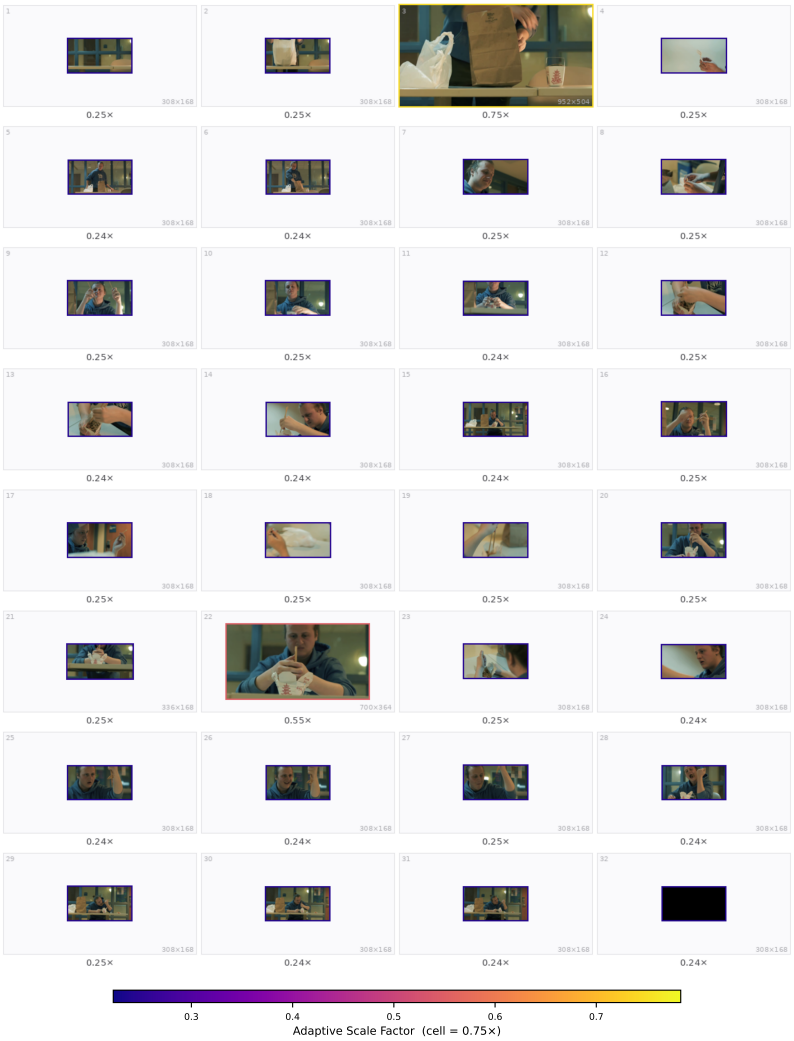


Figure 19: Case 4: VideoMME (Fu et al., 2025) (Vanilla ✓ → ResAdapt ×; failure). The policy enlarges an adjacent frame but compresses the frame containing the fork, losing the decisive fine detail.

G Operating Boundaries, Limitations, and Future Work

Where front-end allocation amortizes. The computational overhead of the front-end Allocator exhibits distinct scaling behaviors: scale prediction remains lightweight and nearly independent of context length (85.6–97.4 ms from 16 to 128 frames), whereas coarse preview feature extraction scales linearly with frame count (734.5 ms at 128 frames). Consequently, the net wall-clock benefit is governed by whether downstream attention savings compensate for upstream preview encoding. For longer contexts (64 and 128 frames), every evaluated operating point delivers net end-to-end acceleration, achieving up to a 59.5% latency reduction ($2.47\times$ speedup) at 128 frames. Conversely, for short contexts (16 frames) or conservative compression regimes ($R > 70\%$), the fixed front-end encoding floor can exceed modest downstream savings (Table 10). Pre-encoding allocation thus delivers its strongest practical returns on long-duration sequences and deep compression regimes where quadratic language attention and ViT FLOPs dominate total runtime.

Perceptual resolution limits of coarse previews. Because the Allocator predicts per-frame fidelity exclusively from coarse SmolVLM preview features, it faces an inherent perceptual bottleneck when query relevance hinges on subtle visual evidence that is already illegible at preview resolution. Such failure modes typically involve microscopic text, low-contrast small objects, or brief visual cues visible for only a single frame (as exemplified by the fork localization failure in Figure 19). Raising $s_{\min}$ improves recognition accuracy on text-dense benchmarks such as TextVQA (improving from 67.8 to 83.1; Table 9) at the cost of higher token consumption. Promising avenues to overcome this preview bottleneck include incorporating resolution-sufficiency estimation (Kimhi et al., 2026) to detect ambiguous previews, or adopting a hierarchical two-stage preview policy that selectively re-inspects uncertain frames before final backbone commitment.

Locality of temporal regularization vs. global video semantics. The temporal-similarity regularizer $\mathcal{L}_{\text{sim}}$ operates locally on consecutive frame pairs (f_t, f_{t+1}). While this adjacent pairwise penalty effectively discourages duplicate high-resolution spending across neighboring video frames, it does not explicitly capture long-range visual repetitions—such as periodic human actions, recurring camera sweeps over identical scenery, or presentation slides revisited after digressions. In such scenarios, cross-temporal redundancy is regulated primarily by the Allocator's full-video attention and global cost pressure rather than pairwise regularizers. A promising direction for future research is extending pairwise temporal regularization into global spatiotemporal graph formulations, global keyframe memory buffers, or memory-augmented retrieval mechanisms (Wu et al., 2025) that can identify and downscale semantically identical frames separated by arbitrary temporal horizons.

Serving dynamics and systems-level scaling. Our empirical evaluations profile batch-size-1 sequential inference on dedicated NVIDIA H100 hardware. In real-world enterprise serving environments governed by continuous batching, heterogeneous requests, and variable-length inputs (such as vLLM or SGLang engines), additional deployment opportunities and systems challenges arise. In multi-query serving scenarios on identical video streams, caching coarse preview representations can partially amortize preview extraction overhead across requests, while cross-request pipelining could further mitigate front-end latency. While Tables 21 and 18 establish the modular composability of ResAdapt with in-encoder token merging and multi-turn agentic search harnesses, fully characterizing system throughput, memory bandwidth trade-offs, and multi-tenant tail latency under continuous batching workloads represents an important avenue for future systems research.

Credit assignment and reward granularity. Finally, ResAdapt is trained via contextual bandit policy gradients with sequence-level scalar rewards, avoiding frame-level resolution annotations. In our group-relative formulation, contrasting multiple candidate allocations ($M = 16$) per prompt provides the comparative signal to distinguish effective frame allocations from redundant ones. Incorporating learned value baselines, self-supervised spatial saliency pre-training, or visual grounding verifiers that supply intermediate dense reward signals could further improve sample efficiency and accelerate policy convergence.