SegMo: Co-Designing Content-Aware Sparsity and Locally-Cohesive Segment Parallelism for Efficient VLM Inference

Supplementary Material

1. Additional Analysis and Technical Clarifications

1.1. Novelty: Upstream Intervention vs. Downstream Pruning

There are three discrete stages of Video VLM inference: (1) Raw video sampling, (2) Vision encoding, and (3) LLM prefill/decoding. SegMo optimizes Stage 1 by selecting high-quality semantic anchors at the source. In contrast, Stage 3 methods (e.g., FastVID) prune tokens only after they pass through the vision encoder. This upstream positioning provides the below key benefits:

Enabling Zero-Communication Parallelism: This early-stage intervention is the structural foundation of our efficiency. By defining semantic boundaries via CAS before encoding, we establish a hard partition granularity for LSP. This allows independent scenes to be processed by GPUs in isolation during the prefill stage. This upstream zero-communication is architecturally infeasible for post-encoding methods (e.g., FastVID, DyCoke, PruneVID), where the cross-GPU synchronization of KV caches is inherently required once tokens enter the transformer.

Accuracy Enhancement vs. Lossy Approximation: Unlike lossy Stage 3 pruning which merely approximates uncompressed results—e.g., FastVID typically retain 90%+ of the original uncompressed baseline accuracy—SegMo (Stage 1) surpasses the original uncompressed baseline (Tab. S1). This validates our approach of elevating the performance ceiling at the source.

Table S1. VideoMME (Med.) accuracy with Qwen2-VL (max 16,384 tokens, 768 frames). Results for FastVID/PruneVID are cited from [26] in the main paper under identical configurations.

Method	Uncompressed	**SegMo**	FastVID*	PruneVID*
VideoMME (Med.)	60.4	**62.2**	58.3	54.0

Orthogonality: SegMo is strictly complementary to downstream optimizations and can be seamlessly integrated with FastVID, DyCoke, PruneVID, among others. By enhancing the uncompressed inference quality at Stage 1, we enable subsequent methods to achieve higher performance ceilings than if they were applied to a standard baseline.

1.2. System Analysis: Scalability and Efficiency

Scalability: Works like FastVID evaluate latency on a single 6,272-token configuration, limiting long-video scalability evidence. In contrast, SegMo is architected for **ultra-long video** where inter-GPU communication bottlenecks latency. By eliminating this via LSP, SegMo achieves significant speedup—mitigating attention's quadratic complexity—yielding substantial reductions.

Efficiency: By selecting frames before vision encoding, SegMo **avoids redundant encoder overhead**. As shown in Tab. S2, under a same 21000 LLM token budget, our pre-encoder filtering achieves significantly lower prefill latency compared to post-encoding methods.

Table S2. For fair comparison, both methods use one H100 without parallelism. SegMo shifts overhead to sampling but slashes Vision Encoder and LLM forward latencies and requires no post-encoding compression.

Method	Prefill Time(s)			
	Sample	Vis. Encoder.	Comp. after VE.	LLM Fwd.
Fastvid	**39.071**	3.518	0.676	0.881
SegMo	39.960	**0.914**	**0**	**0.217**

Micro-pipeline and Overhead Masking. A potential concern for upstream intervention is the "cold start" latency from scene detection and CLIP scoring. We resolve this via a micro-pipeline that parallelizes scene detection and overlaps it with mandatory Video I/O (Tab. S3). This co-design ensures that detection costs are entirely masked by unavoidable I/O wait times, introducing near-zero additional latency to the end-to-end pipeline.

Table S3. Pre-processing latency (s). Standalone detection (19.077s) is **entirely masked** by I/O. CAS incurs a negligible 0.00036s to retrieve results.

Method	Video I/O	Scene Detect **(fully masked)**	**CAS Module**	
			Scene Retrieval	**CLIP**
SegMo	39.071	19.077	**0.00036**	**0.822**

Computational Efficiency of Frame Selection. Identifying the most relevant frame via CLIP ($O(N)$) is prohibitive for long-video. In LongVideoBench, the frame (avg. 1,057, 1 fps) vs. scene (avg. 169.80) disparity makes exhaus-

tive scoring infeasible (Tab. S4). SegMo's $O(N_{scene})$ head-frame strategy sidesteps this via scene semantics—slashing latency while outperforming uniform-sampling accuracy.

Table S4. Efficiency of Frame Selection (on LongVideoBench).

Selection Strategy	Latency (s)	CLIP Computations
Most relevant frame	49.650	2,428 frames
Head-frame (SegMo)	**0.913**	**15 frames**

2. Robustness Analysis

Regarding Long-take cases, CAS falls back to uniform sampling, ensuring robustness across all video patterns without performance degradation. Such instances are rare in general long-video: only 6/1800 in VideoMME (Med.&Long), where SegMo correctly resolves 5/6. Even on VideoMME (Short) (Tab. S5) where 69/900 are zero-cut, SegMo's overall accuracy still exceeds the unoptimized baseline.

Table S5. VideoMME-short accuracy (MiniCPM-o 2.6). Despite 69 Long-Takes samples, SegMo outperforms unoptimized baselines.

Method	32 frames	64 frames
Baseline (Uniform)	72.78	75.78
SegMo (Ours)	**77.26**	**78.70**

3. Algorithm for Content-Aware Sparsification

Algorithm 1 provides the detailed implementation of our Content-Aware Frame Budget Allocation strategy, as introduced in Section 4.1. This algorithm details the process of calculating the normalized Query Relevance (RL) and Temporal Redundancy (RD) scores, combining them into the final Information Value $V(C_k, Q)$, and then proportionally allocating the total frame budget M (handling scaling and minimums) to determine the final workload W (which corresponds to the frame budget m_k) for each scene.

4. Hardware-Aware Greedy Partitioning Algorithm

Algorithm Algorithm 2 details the Hardware-Aware Greedy Partitioning strategy (Sec. 4.2), which solves the makespan minimization objective. The algorithm first calculates "ideal cut-off points" based proportionally on heterogeneous GPU capacities ($Cap(g_j)$). However, because our constraint mandates that scenes (C_k) cannot be split, the algorithm must approximate this ideal partition. It iterates through each scene's workload $W(C_k)$ (from CAS) and makes a greedy decision: it assigns the entire scene (as an indivisible unit) to the GPU (current or next) that brings the cumulative cost closest to the pre-calculated ideal cut-off point, thereby minimizing the cluster makespan.

5. Edge Case Allocation Strategy ($K > M$)

Algorithm 3 details the "relevance-prioritized" edge case strategy, referenced in Section 4.1. This algorithm is invoked when the number of detected semantic scenes (K) exceeds the maximum total frame budget (M).In this scenario, the standard proportional allocation (detailed in Algorithm 1) is unsuitable, as there are not enough frames to even allocate one frame per scene. Instead, this policy computes the Query Relevance score for all scenes and allocates exactly one frame to each of the top M most relevant scenes. This ensures the most critical content is preserved while strictly adhering to the budget constraints.

6. Detailed Results for the Full SegMo System (CAS+LSP)

This section provides the complete and detailed experimental results for our full SegMo system (Table S6), which integrates Content-Aware Sparsification (CAS) with Locally-Cohesive Segment Parallelism (LSP). The following tables present the comprehensive accuracy and latency metrics across all benchmarks and model configurations, which are summarized in the main paper (e.g., in Table 1).

7. Full Layer-wise Attention Heatmaps for Local Cohesion

As referenced in Section 3.3 (Insight 1), this section provides the complete layer-wise attention heatmaps that form the empirical basis for our claim that VLM attention is locally-cohesive.The following figure (Figure S1 - Figure S7) displays the attention score matrices for all 28 layers of the MiniCPM-o 2.6 model, generated from experiments on the VideoMME dataset. Across nearly all layers, a consistent pattern is visible: attention scores are highly concentrated within the diagonal blocks (representing semantic scenes), while inter-scene (off-diagonal) attention is negligible.This provides comprehensive evidence that a semantic segment (C_k) is the correct independent unit for VLM processing, driving our co-design for both sparsification and parallelism.

8. Qualitative Frame Selection Analysis

This section provides additional qualitative visualizations demonstrating the effectiveness of our Content-Aware Sparsification (CAS) module across various long-video examples. These figures are designed to illustrate the dynamic allocation of frame m_k based on content value.In the following visualizations, scene status and CAS decisions are highlighted using the following conventions:

1. Selected Frames: Key frames ultimately chosen by the CAS module are marked with a green checkmark (✓) below the frame.
2. Critical Scenes: Scenes containing the Ground Truth information necessary to answer the user query (Q/A scenes) are marked with a red background box. CAS is

observed to allocate a significantly higher frame budget to these critical scenes.

Algorithm 1 Adaptive Frame Budgeting

Input: Scenes $C = \{C_1, ..., C_K\}$, Query Q, MaxTotalFrames M_{max}, Weights w **Output:** Scene cost list $W = \{W_1, ..., W_K\}$

$K \leftarrow$ length(C)
$W \leftarrow [0] \times K$
{1. Calculate raw scores }
RL$(Q, C_k) \leftarrow$ BatchRelevanceScore(C, Q)
RD$(C_k) \leftarrow$ BatchRedundancyScore(C)
{2. Normalize weights}
$s\_weights \leftarrow$ Normalize(RL(Q, C_k))
$d\_weights \leftarrow$ Normalize(RD(C_k))
{3. Calculate final information value}
$information\_values \leftarrow [0] \times K$
for $k \leftarrow 1$ **to** K **do**
    $information\_values[k] \leftarrow w \cdot s\_weights[k] + (1 - w) \cdot d\_weights[k]$
end for
{4. Proportional allocation based on initial budget (n*K)}
$B_{initial} \leftarrow n \times K$
$budgets \leftarrow [0] \times K$
$B_{allocated} \leftarrow 0$
for $k \leftarrow 1$ **to** K **do**
    $alloc \leftarrow$ round$(information_values[k] \times B_{initial})$
    $budgets[k] \leftarrow \max(1, alloc)$ {Ensure at least 1 frame}
    $B_{allocated} \leftarrow B_{allocated} + budgets[k]$
end for
{5. Scale down if allocated budget exceeds max limit}
if $B_{allocated} > M_{max}$ **then**
    $scale\_factor \leftarrow M_{max} / B_{allocated}$
    for $k \leftarrow 1$ **to** K **do**
        $budgets[k] \leftarrow \max(1,$ round$(budgets[k] \times scale_factor))$
    end for
end if
$W \leftarrow budgets$
return W

Algorithm 2 Hardware-Aware Greedy Partitioning

Input: Scene costs $W = \{W_1, ..., W_K\}$, GPU capacities $Cap = \{Cap_1, ..., Cap_N\}$ **Output:** Partition $\pi = \{P_1, ..., P_N\}$, (list of lists of scene indices)

$total\_cost \leftarrow$ sum(W)
$total\_capacity \leftarrow$ sum(Cap)
$\pi \leftarrow$ [[] for $i \leftarrow 1$ to N] {Initialize empty assignments}
$ideal\_cutoffs \leftarrow []$
{1. Calculate ideal cost cut-off points for N-1 boundaries}
for $j \leftarrow 1$ **to** $N - 1$ **do**
    $proportion \leftarrow$ sum$(Cap[1..j])/total_capacity$
    $ideal\_cutoffs$.append$(total_cost \times proportion)$
end for
$ideal\_cutoffs$.append(infinity) {Add sentinel for last GPU}
{2. Assign scenes greedily based on cut-off points}
$current\_cost \leftarrow 0$
$gpu\_idx \leftarrow 0$
for $k \leftarrow 1$ **to** K **do**
    if $W[k] > 0$ **then**
        {Only process scenes with budget > 0}
        $cutoff \leftarrow ideal\_cutoffs[gpu\_idx]$
        $cost\_if\_assign\_current \leftarrow$ abs$(current_cost + W[k] - cutoff)$
        $cost\_if\_assign\_next \leftarrow$ abs$(current_cost - cutoff)$
        if $cost\_if\_assign\_next < cost\_if\_assign\_current$ **then**
            $gpu\_idx \leftarrow gpu\_idx + 1$ {Move to next GPU}
        end if
        $\pi[gpu\_idx]$.append(k) {Assign scene k to GPU j}
        $current\_cost \leftarrow current\_cost + W[k]$
    end if
end for
return π

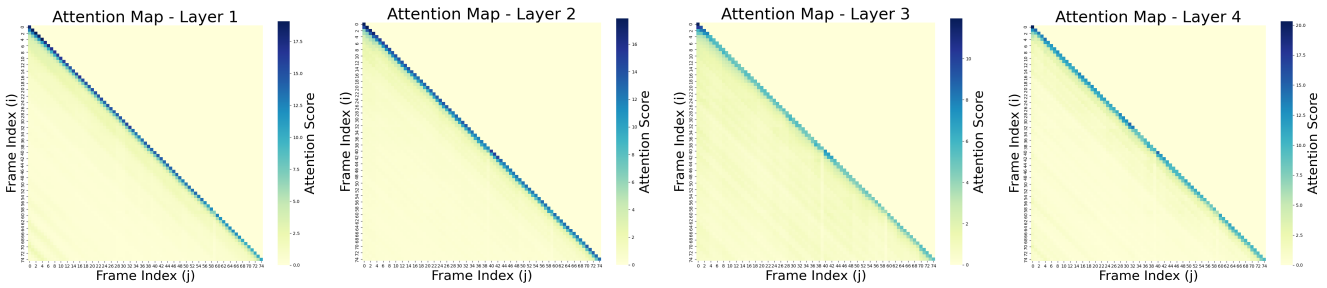
Algorithm 3 Relevance-Prioritized Allocation (Edge Case: $K > M_{max}$)

Input: Scenes $C = \{C_1, ..., C_K\}$, Query Q, MaxTotalFrames M_{max}, FullFrameList F_{all} **Output:** Scene cost list $W = \{W_1, ..., W_K\}$

RL$(Q, C_k) \leftarrow []$
$K \leftarrow$ length(C)
$W \leftarrow [0] \times K$ {Initialize all costs to zero}
{1. Calculate Relevance Scores}
for $k \leftarrow 1$ **to** K **do**
    $rep\_frame \leftarrow$ GetRepresentativeFrame(C_k, F_{all})
    $s\_raw \leftarrow$ CLIP_Score(rep_frame, Q)
    RL(Q, C_k).append(s_raw)
end for
{2. Scene count exceeds budget.}
$top\_indices \leftarrow$ GetTopKIndices(RL$(Q, C_k), M_{max})$
$N_{top} \leftarrow$ length$(top_indices)$
for $j \leftarrow 1$ **to** N_{top} **do**
    $i \leftarrow top\_indices[j]$ {Get the actual scene index}
    $W[i] \leftarrow 1$ {Allocate 1 frame to top relevant scenes}
end for
return W

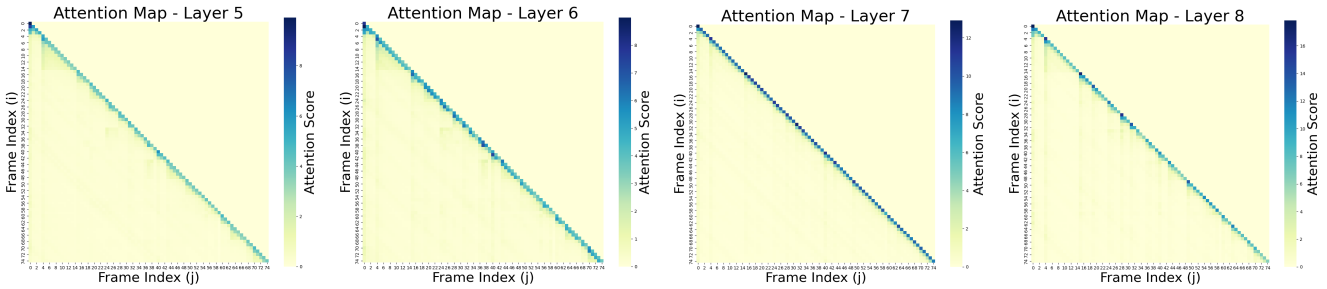
Table S6. SegMo (CAS + LSP) consistently improves accuracy over the Naive Data Parallel baseline, even when both systems are executed in parallel on 2 GPUs.

Model	Frames	Benchmark	Baseline (Data Parallel, 2 GPUs) Acc	SegMo (Ours) (LSP, 2 GPUs) Acc
Qwen2-VL-7B-Inst.	32	LVBench	32.44	**40.75** +8.31
	64	LVBench	32.98	**39.81** +6.83
	128	LVBench	37.73	**40.68** +2.95
	256	LVBench	36.33	**39.81** +3.48
	32	Video-MME (4-15m)	62.10	**64.21** +2.11
	64	Video-MME (4-15m)	62.40	**64.21** +1.81
	128	Video-MME (4-15m)	63.00	**66.22** +3.22
	32	Video-MME (30-60m)	63.60	**64.22** +0.62
	64	Video-MME (30-60m)	64.00	**64.22** +0.22
	128	Video-MME (30-60m)	63.00	**64.44** +1.44
	32	LongVideoBench	45.68	**50.23** +4.55
	64	LongVideoBench	45.63	**50.80** +5.17
	128	LongVideoBench	48.62	**50.74** +2.12
MiniCPM-o 2.6	32	LVBench	34.92	**42.02** +7.10
	64	LVBench	36.86	**43.23** +6.37
	128	LVBench	38.47	**43.03** +4.56
	256	LVBench	41.62	**43.30** +1.68
	32	Video-MME (4-15m)	60.56	**64.32** +3.76
	64	Video-MME (4-15m)	62.33	**65.10** +2.77
	128	Video-MME (4-15m)	64.30	**67.79** +3.49
	32	Video-MME (30-60m)	57.67	**59.89** +2.22
	64	Video-MME (30-60m)	59.00	**59.00**
	128	Video-MME (30-60m)	58.00	**58.33** +0.33
	32	LongVideoBench	46.78	**53.25** +6.47
	64	LongVideoBench	46.23	**53.54** +7.31
	128	LongVideoBench	48.78	**54.00** +5.22



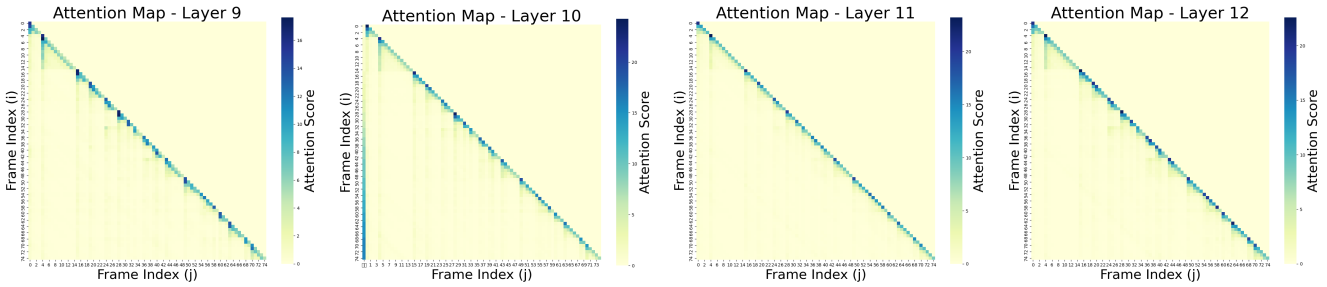
(a) Layer 1 (b) Layer 2 (c) Layer 3 (d) Layer 4

Figure S1. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). These figures show the attention matrices for all 28 layers of the MiniCPM-o 2.6 model on a VideoMME sample. The strong diagonal blocks confirm high intra-scene cohesion. (Part 1 of 7, Layers 1-4)



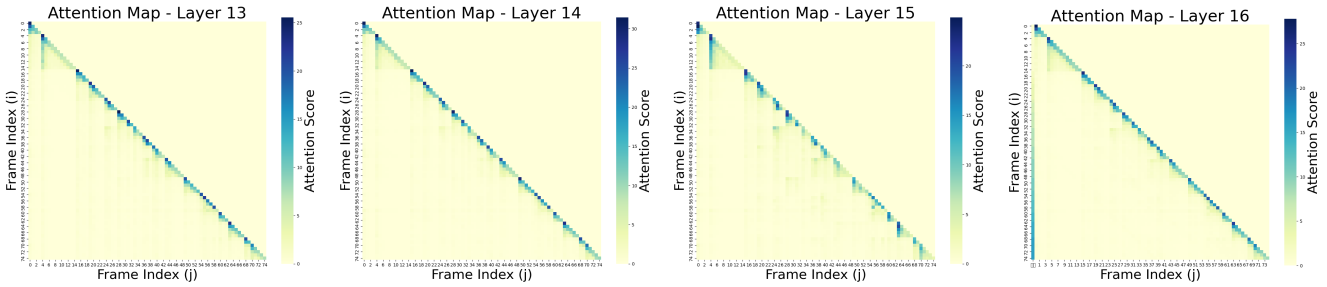
(a) Layer 5 (b) Layer 6 (c) Layer 7 (d) Layer 8

Figure S2. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 2 of 7, Layers 5-8)



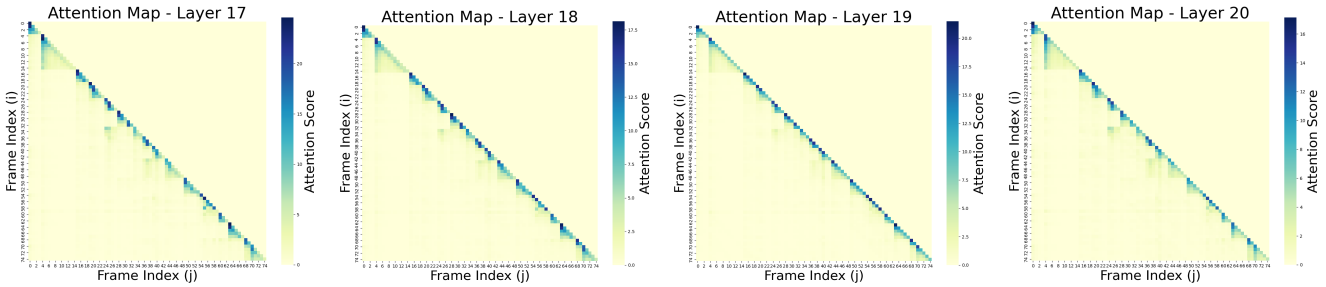
(a) Layer 9 (b) Layer 10 (c) Layer 11 (d) Layer 12

Figure S3. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 3 of 7, Layers 9-12)



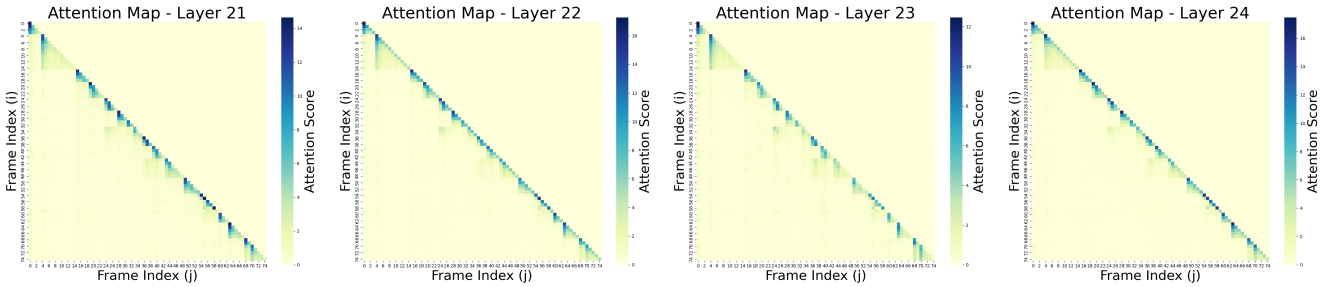
(a) Layer 13 (b) Layer 14 (c) Layer 15 (d) Layer 16

Figure S4. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 4 of 7, Layers 13-16)



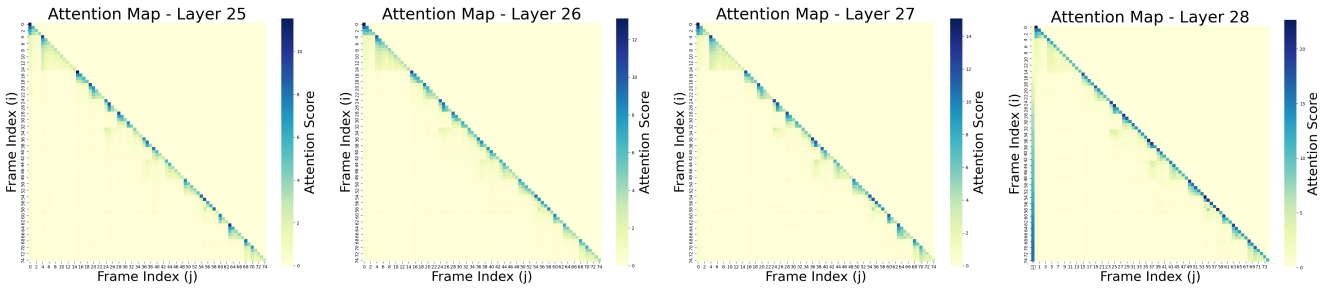
(a) Layer 17 (b) Layer 18 (c) Layer 19 (d) Layer 20

Figure S5. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 5 of 7, Layers 17-20)



(a) Layer 21 (b) Layer 22 (c) Layer 23 (d) Layer 24

Figure S6. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 6 of 7, Layers 21-24)



(a) Layer 25 (b) Layer 26 (c) Layer 27 (d) Layer 28

Figure S7. Full layer-wise attention heatmaps for Insight 1 (Local Cohesion). (Part 7 of 7, Layers 25-28)

Question: Which of the following is visible in the background of the video when the miniature bottle is shown empty?

A. Water. **B. Cork.** **C. Shells.** **D. Star.**

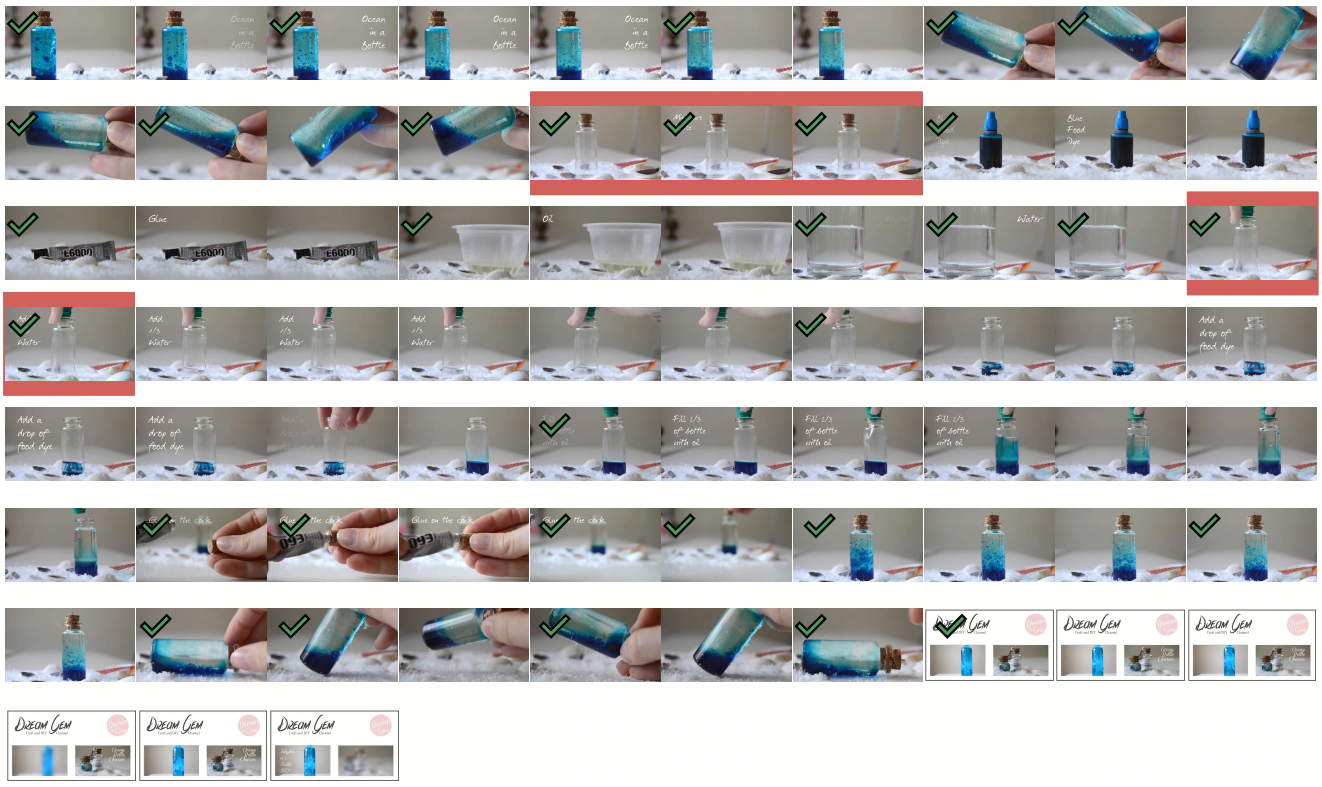


Figure S8. Supplemental Visualization Result of CAS.

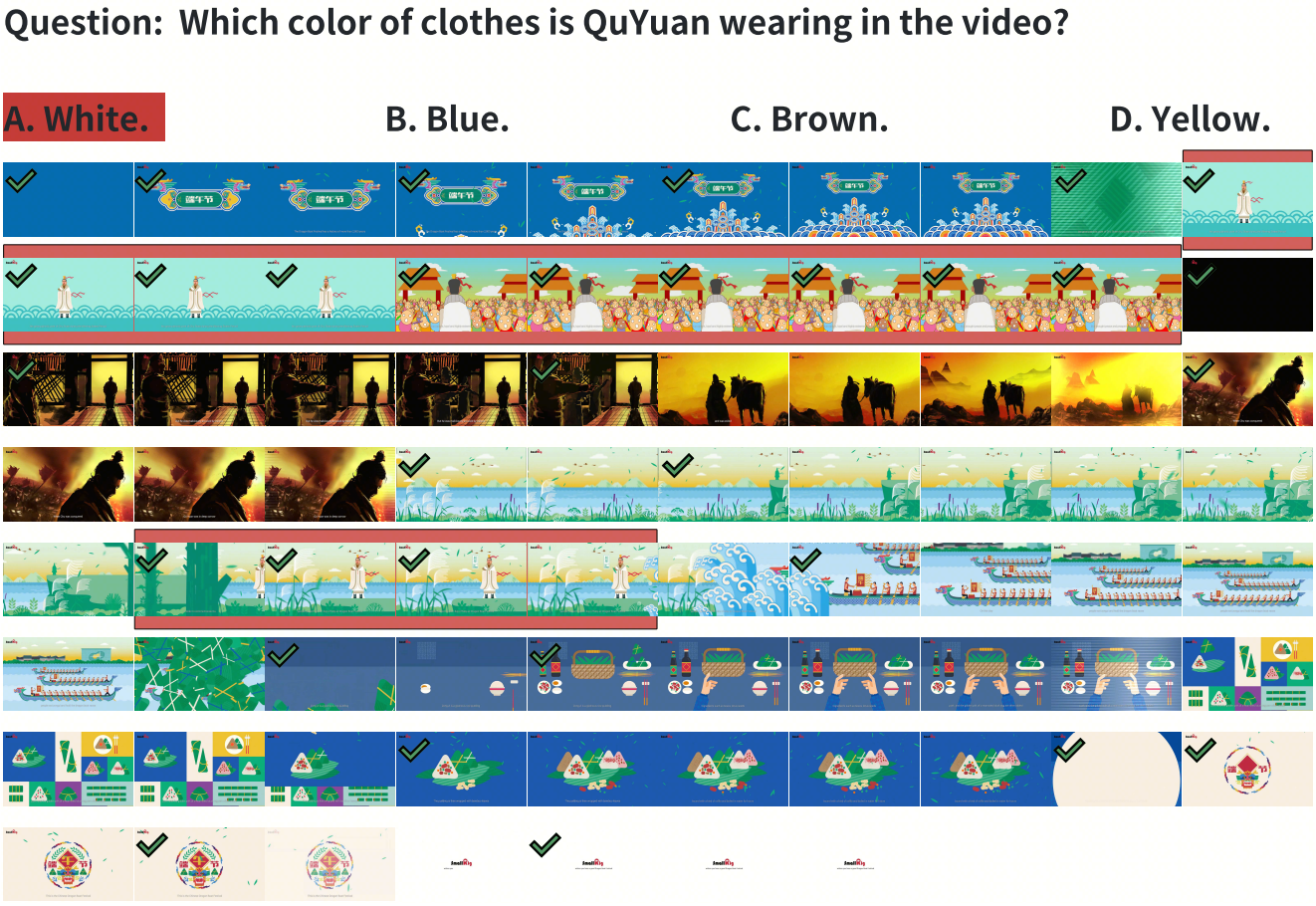


Figure S9. Supplemental Visualization Result of CAS.

Question: How is the smoke generated by the man depicted in the video?

A. By burning a piece of cloth. **B. By lighting a torch.**

C. By smoking. **D. By lighting a bonfire.**

Figure S10. Supplemental Visualization Result of CAS.