Accelerating Streaming Video Large Language Models via Hierarchical Token Compression

Supplementary Material

In this appendix, we begin by elaborating on the detailed experimental settings in Section A, which covers specific descriptions of the benchmarks, model architectures, and baseline methods employed. Subsequently, Section B presents additional ablation studies, including performance comparisons on various model backbones, the impact of cache update intervals, and hyperparameter sensitivity analyses. Section C and Section D present the peudocode of our STC framework and more visualizations by STC-Cacher.

A. Detailed Experiment Settings

Benchmark Details. We evaluate our STC on various video understanding benchmarks, detailed as follows:

- **OVO-Bench** [34] is a benchmark for evaluating the online video understanding capabilities of Video-LLMs, with a focus on temporal awareness. It includes 644 unique videos, ranging from several minutes to half an hour in length, and features 2,814 human-curated meta-annotations with precise timestamps. The tasks are structured into three categories: Backward Tracing, Real-Time Visual Perception, and Forward Active Responding.
- **StreamingBench** [23] is a comprehensive benchmark designed to evaluate the streaming video understanding capabilities of MLLMs, emphasizing the gap between offline processing and real-time human-like interaction. It consists of 900 videos covering diverse scenarios and 4,500 human-curated QA pairs, where queries are presented at specific timestamps to simulate continuous inputs. The benchmark evaluates 18 distinct tasks organized into three core categories: Real-Time Visual Understanding, Omni-Source Understanding, and Contextual Understanding.
- **EgoSchema** [32] is a large-scale benchmark for long-form, egocentric video question answering, containing over 250 hours of video footage. It is designed to evaluate complex causal reasoning from a first-person perspective by asking "why" a particular action was performed. The benchmark consists of multiple-choice questions that require models to infer the actor's intent.
- **VideoMME** [16] comprises 900 videos and 2,700 multiple-choice questions across six domains, with durations from 11 seconds to 1 hour, categorized into short, medium, and long subsets.
- **MLVU** [64] (Massive Long Video Understanding) is a benchmark for long-form video comprehension, featuring 4,000 videos that range from 30 minutes to over 2 hours, totaling 4,800 hours. It comprises 13 challenging tasks designed to test multimodal and long-context reasoning, including character-centric question answering, plot analysis, and multi-event retrieval.

Model Details. We evaluate our STC on multiple VideoLLMs, including *End-to-End Online VideoLLMs* and *Offline-to-Online Frameworks*, detailed as follows:

End-to-End Online VideoLLMs:

- **LiveCC** [6] explores the use of large-scale, cost-effective automatic speech recognition (ASR) transcripts for training Video LLMs. It proposes a novel streaming training approach that densely interleaves ASR words with video frames according to their timestamps, enabling the model to learn fine-grained, temporally-aligned vision-language modeling. To support this method, the work introduces two new datasets: Live-CC-5M for pre-training and Live-WhisperX-526K for supervised fine-tuning. This approach results in strong general video question answering (QA) performance while also exhibiting a novel capability for real-time video commentary.
- **StreamForest** [60] is a novel architecture specifically designed for efficient online video understanding in streaming scenarios. Unlike conventional Video-LLMs that either process entire videos offline or apply aggressive compression losing spatiotemporal details, StreamForest continuously analyzes video streams at 1 fps through a dual-memory mechanism. Central to the design is the Persistent Event Memory Forest (PEMF), which adaptively organizes video frames into hierarchical event-level tree structures guided by three penalty functions (temporal distance, content similarity, and merge frequency), enabling persistent long-term memory under strict token budgets. Complementing this, a Fine-grained Spatiotemporal Window (FSTW) captures detailed short-term visual cues for enhanced real-time perception. The work also introduces OnlineIT, an instruction-tuning dataset tailored for streaming video tasks, and ODV-Bench, a benchmark for autonomous driving scenarios.
- **Dispider** [36] is a novel framework that enables active real-time interaction with Video Large Language Models by disentangling perception, decision, and reaction into asynchronous modules. Unlike previous streaming methods that alternate between video processing and response generation, Dispider features a lightweight scene-based perception module that continuously monitors video streams and dynamically segments them into non-uniform clips based on visual boundaries. A real-time decision module evaluates whether to trigger responses using special tokens (\<TODO\> and \<ANS\>) and historical context,

while an asynchronous reaction module generates detailed responses without interrupting ongoing video processing. This non-blocking architecture ensures timely, contextually accurate responses for long-duration videos while maintaining computational efficiency. Dispider significantly outperforms existing streaming models on StreamingBench and demonstrates strong performance on conventional long-video benchmarks.

Offline-to-Online Frameworks:

- **ReKV** [12] is a novel, training-free framework designed to enable existing Video Large Language Models (Video-LLMs) with efficient Streaming Video Question-Answering (StreamingVQA) capabilities. Unlike traditional VideoQA systems that must process an entire video before responding to a query, ReKV continuously analyzes video streams in a streaming manner, allowing for prompt responses. During the encoding phase, the framework employs a sliding-window attention mechanism to reduce computational overhead, while simultaneously storing processed video KV Cache in memory to prevent information loss. When a query is posed, ReKV utilizes a retrieval module to load only the query-relevant KV-Caches to serve as context, enabling efficient answer generation. This design decouples the video encoding and question-answering processes, significantly enhancing efficiency and responsiveness, particularly when handling long videos.

Baseline Details. We compare our STC with below dominant token compression methods:

- **ToMe** [3] is a token reduction method that improves Vision Transformer (ViT) throughput by gradually merging similar tokens across transformer layers, rather than pruning them. Unlike pruning-based approaches, ToMe can be applied off-the-shelf without retraining while maintaining compatibility with batched inference.
- **VidCom**^{2} [27] proposes a plug-and-play inference acceleration framework centered on frame uniqueness. Unlike methods using uniform compression, it employs a two-stage strategy: first dynamically adjusting compression intensity based on the distinctiveness of each video frame, and then performing adaptive token compression. Notably, it addresses the implementation constraints of previous methods by maintaining full compatibility with efficient operators like Flash Attention [10].
- **VisionZip** [55] introduces a text-agnostic token reduction framework applied before the LLM input. It identifies informative "dominant" tokens based on the self-attention weights within the vision encoder and aggregates the remaining redundant tokens through similarity-based merging to preserve contextual details.

B. Additional Ablation Studies

In this section, we present further quantitative analysis of our proposed method. **Results on other models.** We first present the performance comparison on additional model architectures in Table 6 and Table 7.

Methods	OVO-Bench			
	EPM	STU	REC	Avg
Attn ($R_{\text{Cacher}} = 85\%$)	30.6	33.9	11.2	25.2
MLP ($R_{\text{Cacher}} = 85\%$)	56.9	43.3	27.5	42.6
Attn and MLP ($R_{\text{Cacher}} = 75\%$)	**57.9**	**46.1**	**29.9**	**44.6**

Table 6. **Effects of different reusing features in STC-Cacher with StreamForest.** "Attn" and "MLP" are reusing features from attention and MLP.

Methods	OVO-Bench			
	EPM	STU	REC	Avg
Feature	56.9	45.5	28.9	43.8
Value	**58.3**	44.9	29.8	44.3
Key	57.9	**46.1**	**29.9**	**44.6**

Table 7. **Compares various features for dynamic token evaluation to identify the optimal dynamic token set for feature caching and reuse.** We compare the performance of reusing Feature, Value, and Key.

Cache Interval	OVO-Bench			
	EPM	STU	REC	Avg
$\mathcal{N} = 1$	54.6	51.1	25.5	43.7
$\mathcal{N} = 4$	52.2	42.7	24.9	39.9
$\mathcal{N} = 7$	51.9	44.4	23.8	40.0
$\mathcal{N} = 10$	50.8	46.1	22.9	39.9
$\mathcal{N} = \infty$	44.1	38.2	18.3	33.5

Table 8. **Ablation study on the cache update interval $\mathcal{N}$.** We compare different update frequencies ranging from frame-by-frame updates ($\mathcal{N} = 1$) to no updates ($\mathcal{N} = \infty$). Results indicate that more frequent updates consistently yield better performance.

B.1. Impact of Cache Update Interval

Table 8 investigates sensitivity to cache update interval $\mathcal{N}$. We vary $\mathcal{N}$ from frame-by-frame to static. Results show that **more frequent updates yield better performance**. Performance drops sharply as the interval becomes static, revealing feature drift. This confirms periodic updates are essential for tracking temporal dynamics.

B.2. Hyperparameter Sensitivity Analysis

B.2.1. Token Retention Ratios

Table 9 analyzes the impact of token retention ratios in the Cacher and Pruner modules.

- **Impact of Cacher Ratio (R_{Cacher}):** As shown in Table 9 (a), a higher update ratio generally yields superior performance, as refreshing more tokens preserves dynamic information against rapid changes.
- **Impact of Joint Ratios:** Table 9 (b) shows the interplay between Cacher and Pruner ratios. Performance remains

Ablation Settings	OVO-Bench			
	EPM	STU	REC	Avg
(a) Impact of Cacher Ratio				
$R_{\text{Cacher}} = 85\%$	54.9	45.5	24.5	41.6
$R_{\text{Cacher}} = 75\%$	52.2	42.7	24.9	39.9
$R_{\text{Cacher}} = 50\%$	50.8	46.6	25.4	40.9
(b) Impact of Joint Ratios				
$R_{\text{Cacher}} = 50\%$, $R_{\text{Pruner}} = 75\%$	51.5	46.1	25.6	41.1
$R_{\text{Cacher}} = 75\%$, $R_{\text{Pruner}} = 50\%$	50.8	46.2	26.9	41.3
$R_{\text{Cacher}} = 75\%$, $R_{\text{Pruner}} = 75\%$	50.8	46.1	24.6	40.5

Table 9. **Ablation studies on Cacher and Pruner hyperparameters.** We analyze (a) the impact of the Cacher update ratio R_{Cacher}, and (b) the joint effect of R_{Cacher} and the Prun ratio R_{Pruner}.

Metrics	OVO-Bench			
	EPM	STU	REC	Avg
$a_{\text{temporal}} + a_{\text{spatial}}$	51.2	48.9	25.9	42.0
$a_{\text{temporal}} + 2a_{\text{spatial}}$	51.1	48.8	26.0	42.0
$2a_{\text{temporal}} + a_{\text{spatial}}$	50.1	47.7	25.9	41.6

Table 10. **Effects of balancing hyper-parameter α between a_{temporal} and a_{spatial}.**

robust across combinations, demonstrating that our framework does not rely on narrow hyperparameter tuning.

B.2.2. Spatial-Temporal Balance

Finally, we investigate the hyperparameters used to combine spatial (a_{spatial}) and temporal (a_{temporal}) scores in Table 10. Results show that performance is optimal when scores are balanced or slightly favor the spatial term. Heavily up-weighting the temporal score leads to degradation, implying that while temporal dynamics are useful, spatial semantics remain the fundamental basis for reasoning.

C. Algorithm Pseudocode

In this section, we provide the detailed pseudocode for the two stages of our Streaming Token Compression (STC) framework. Algorithm 1 details the cache-aware selective computation in the ViT (STC-Cacher), and Algorithm 2 describes the dual-anchor pruning mechanism for the LLM input (STC-Pruner).

D. More Visualizations by STC-Cacher

Figure 8 presents more visualization results by STC-Cacher, which intuitively demonstrates that STC-Cacher can concentrate computational overhead on temporally significant regions while compressing computation for visual tokens that remain static across temporal dimensions.

Algorithm 1 STC-Cacher (ViT Acceleration)

Require: Video frame sequence $\mathbf{V} = \{\mathbf{v}_t\}_{t=1}^{T}$, Cache Interval $\mathcal{N}$, Cache Reuse Ratio $R_{\text{Cacher}} \in [0, 1)$
Ensure: Sequence of visual tokens $\mathbb{Z} = \{\mathbf{Z}_t\}_{t=1}^{T}$

Initialize output list $\mathbb{Z} \leftarrow \emptyset$
for $t = 1 \rightarrow T$ **do**
  if $t \pmod N == 1$ **or** $t == 1$ **then**
    // (I) Reference Frame: Full Computation
    Perform full forward pass on $\mathbf{v}_t$
    Cache representations: $\mathcal{C}^l_{\text{ref}} \leftarrow \{\mathbf{K}^l_{\text{ref}}, \mathbf{V}^l_{\text{ref}}, \mathbf{A}^l_{\text{ref}}, \mathbf{M}^l_{\text{ref}}\}$ for all layers l
    $\mathbf{Z}_t \leftarrow$ Output from final layer
  else
    // (II) Non-reference Frame: Selective Computation
    for each layer l **do**
      // 1. Identify Dynamic Tokens
      $S_{\mathbf{f}} \leftarrow \text{CosSim}(\mathbf{K}^l_{\text{curr}}, \mathbf{K}^l_{\text{ref}})$
      $\mathcal{I}_{\mathbf{f}} \leftarrow \underset{i}{\text{arg top-k}}(1 - S_{\mathbf{f}}[i])$ where $k = \lfloor N_{tok} \cdot (1 - R_{\text{Cacher}}) \rfloor$
      // 2. Selective Attention with Scatter Update
      Compute Query/Value for $\mathcal{I}_{\mathbf{f}}$: $Q^l_{\text{sel}}, V^l_{\text{sel}}$
      $\bar{V}^l_{\mathbf{f}} \leftarrow \mathbf{V}^l_{\text{ref}}; \quad \bar{V}^l_{\mathbf{f}}[\mathcal{I}_{\mathbf{f}}] \leftarrow V^l_{\text{sel}}$
      $A^l_{\text{sel}} \leftarrow \text{Attention}(Q^l_{\text{sel}}, \bar{V}^l_{\mathbf{f}})$
      $\bar{A}^l_{\mathbf{f}} \leftarrow \mathbf{A}^l_{\text{ref}}; \quad \bar{A}^l_{\mathbf{f}}[\mathcal{I}_{\mathbf{f}}] \leftarrow A^l_{\text{sel}}$
      // 3. Selective MLP with Scatter Update
      $M^l_{\text{sel}} \leftarrow \text{MLP}(\text{LN}(\bar{A}^l_{\mathbf{f}}[\mathcal{I}_{\mathbf{f}}]))$
      $\bar{M}^l_{\mathbf{f}} \leftarrow \mathbf{M}^l_{\text{ref}}; \quad \bar{M}^l_{\mathbf{f}}[\mathcal{I}_{\mathbf{f}}] \leftarrow M^l_{\text{sel}}$
    end for
    $\mathbf{Z}_t \leftarrow$ Output from final layer $\bar{M}^L_{\mathbf{f}}$
  end if
  Append $\mathbf{Z}_t$ to $\mathbb{Z}$
end for
return $\mathbb{Z}$

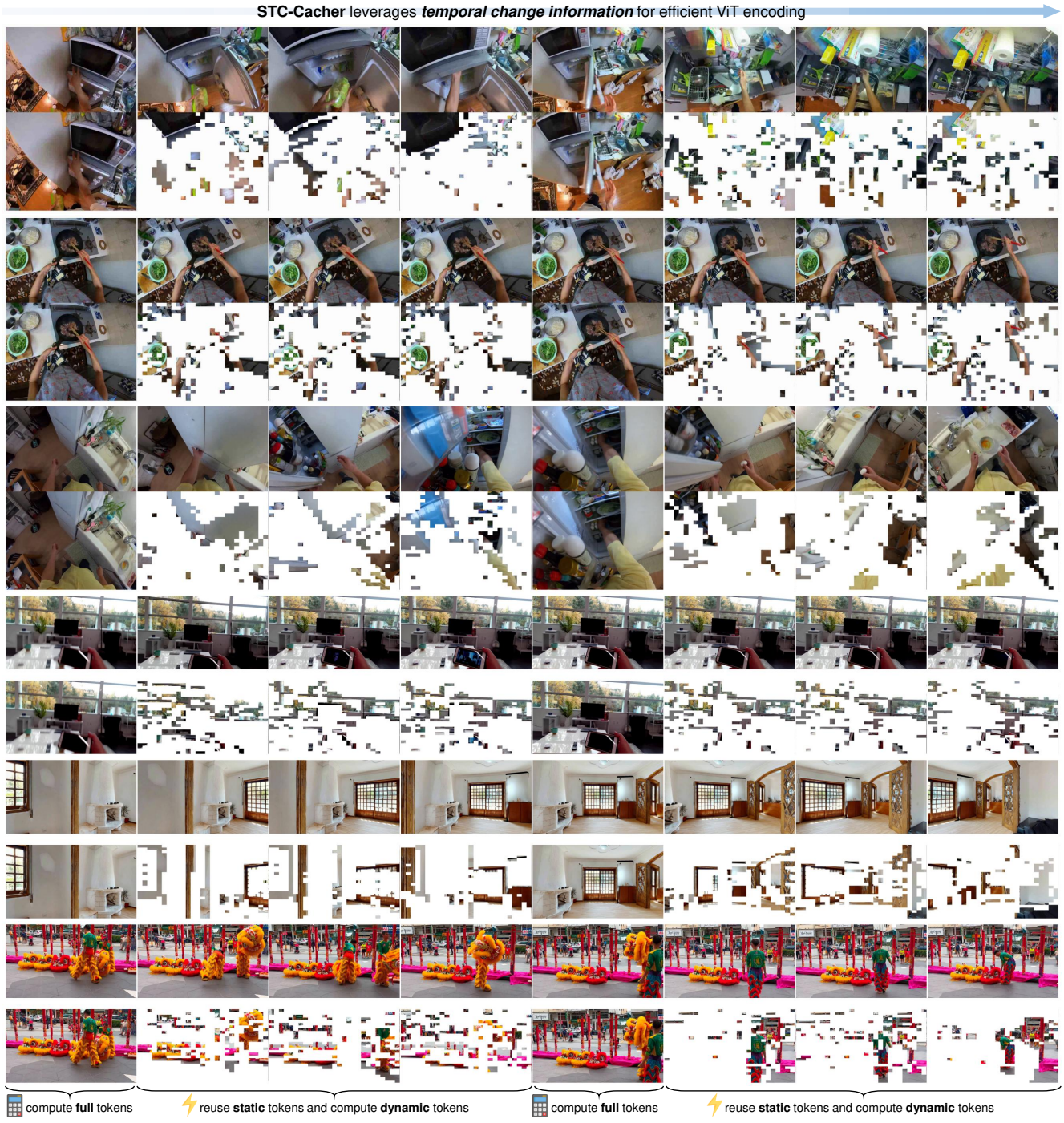

Figure 8. **More visualization of cache-aware selective computation by STC-Cacher.**

Algorithm 2 STC-Pruner (LLM Input Compression)

Require: Visual tokens $\mathbf{Z}_t$ (from STC-Cacher), History Buffer $\mathcal{H}$, Pruning Ratio $R_{\text{Pruner}} \in [0, 1)$, Balance Factor α

Ensure: Pruned tokens $\mathbf{Z}'_t$, Updated Buffer $\mathcal{H}$

// (I) Anchor Establishment

$a_{\text{temporal}} \leftarrow \frac{1}{|\mathcal{H}|} \sum_{h \in \mathcal{H}} h$ *(Historical Context)*

$a_{\text{spatial}} \leftarrow \frac{1}{|\mathbf{Z}_t|} \sum_{z \in \mathbf{Z}_t} z$ *(Current Frame Context)*

// (II) Dynamics Scoring

for each token $z_j \in \mathbf{Z}_t$ **do**

  Calculate joint dissimilarity:

  $S(z_j) \leftarrow \alpha \cdot d_{\cos}(z_j, a_{\text{temporal}}) + (1 - \alpha) \cdot d_{\cos}(z_j, a_{\text{spatial}})$

end for

// (III) Token Pruning

Calculate retention count: $k' \leftarrow \lfloor |\mathbf{Z}_t| \cdot (1 - R_{\text{Pruner}}) \rfloor$

$\mathbf{Z}'_t \leftarrow \text{TopK}(\mathbf{Z}_t, k', \text{key} = S)$

// (IV) Context Update

$\mathcal{H} \leftarrow \text{Enqueue}(\mathcal{H}, a_{\text{spatial}})$ *(Update history for next step)*

return $\mathbf{Z}'_t$