EVTP-IVS: Effective Visual Token Pruning For Unifying Instruction Visual Segmentation In Multi-Modal Large Language Models (Supplementary Material)

Wenhui Zhu[13†*] Xiwen Chen[2*] Zhipeng Wang[3*‡] Shao Tang[3] Sayan Ghosh[3] Xuanzhao Dong[1]
Rajat Koner[4] Yalin Wang[1]

[1] Arizona State University, AZ, USA
[2] Clemson University, SC, USA
[3] LinkedIn Corporation, CA, USA
[4] Ludwig Maximilian University of Munich, Munich, Germany

A. Implementation Details

A.1. Baseline Model: *InstructSeg*

We adopt *InstructSeg* [9] as the baseline model throughout our experiments. It is a modular multi-modal architecture designed for instruction-driven segmentation tasks over both images and videos, supporting various scenarios such as referring expression segmentation and reasoning-based segmentation. The model consists of a set of fixed pre-trained components, each responsible for a distinct stage of visual-language processing, including visual encoding, token fusion, temporal reasoning, and mask decoding. No modules are fine-tuned during inference.

Language Model. The core reasoning component is a 2.7B-parameter PHI-2 [6] language model. It receives tokenized inputs including text instructions, visual tokens, reference frame tokens (for videos), and learnable mask tokens. These are concatenated and processed jointly to produce contextualized embeddings that guide the subsequent segmentation modules. The language model is adapted using Low-Rank Adaptation (LoRA) with a rank of 8. We directly use the publicly released LoRA-tuned weights without further training.

Visual Encoders and Token Pruning. *InstructSeg* employs two vision backbones. First, a SIGLIP encoder [11] is used to extract global visual tokens from each input image or frame. These tokens are aligned with text tokens through the language model. To reduce computational overhead and redundancy, we apply several token pruning methods directly to the SigLIP output. This approach retains important semantic information while reducing the number of visual tokens passed into the language model, leading to more efficient inference. In addition, a SWIN-B transformer [8] is used to extract high-resolution spatial features. These features are not pruned and are passed directly to the segmentation decoder to support fine-grained mask prediction.

Segmentation Decoder. For generating pixel-level predictions, we use a pre-trained MASK2FORMER [5] decoder. It combines a pixel decoder with a transformer-based instance decoder to predict N mask proposals. Each proposal is associated with an embedding and a confidence score. The most relevant masks are selected by computing the similarity between predicted mask embeddings and instruction-guided features.

Object-aware Video Perceiver. To support temporal reasoning in video tasks, the model includes an Object-aware Video Perceiver (OVP) module. This component processes a sequence of $T_r = 4$ reference frames alongside the instruction tokens. Cross-attention and $N_1 = 3$ transformer layers are used to extract compressed reference tokens that encode object-specific dynamics over time. These tokens are integrated into the language model to provide temporal context for segmentation.

Vision-guided Text Fusion. To better connect the instruction semantics with the visual scene, a Vision-guided Multi-granularity Text Fusion (VMTF) module is used after the language model. It fuses coarse and fine-grained instruction embeddings with spatial visual features using $N_2 = 3$ cross-attention layers. The resulting embeddings act as dynamic classifiers for the segmentation decoder.

Inference Pipeline. During inference, the model receives an image or key frame $\mathcal{V} \in \mathbb{R}^{H\times W\times 3}$, an optional reference sequence $\mathcal{V}_r$ (for videos), and a natural language instruction $\mathcal{E}$. Visual and reference tokens are extracted and pruned (when applicable), then combined with instruction and mask tokens and passed through the language model. Refined embeddings from VMTF are used to produce final segmentation masks. All components are pre-trained and remain fixed. Architectural hyperparameters such as the number of queries or attention layers follow the default configuration in the original implementation.

InstructSeg provides a strong and flexible baseline for instruction-conditioned segmentation. Its modular design and compatibility with pruning make it a suitable testbed for analyzing token efficiency in visual-language models.

A.2. TFLOPs Estimation Setup

To estimate computational cost, we implement a FLOPs calculator that evaluates the number of floating-point operations (FLOPs) for each major component in InstructSeg. We measure total TFLOPs (in units of 10^{12} FLOPs) for a single forward pass. The analysis includes the language model, visual encoder, token pruning module, mask decoder, temporal reasoning layers, and the fusion module (VMTF).

Dataset Inputs. We evaluate FLOPs on both image-based and video-based segmentation datasets. For each dataset, we define the number of text tokens T, visual tokens per frame V, and the number of frames F. Table 4 summarizes the token configuration for each benchmark:

Table 1. Input configurations for FLOPs estimation.

Dataset	Text Tokens (T)	Visual Tokens (V)	Frames (F)
RefCOCO	15	729	1
RefCOCO+	15	729	1
RefCOCOg	23	729	1
MM-Conv	50	729	1
ReasonSeg	80	729	1
RefYouTube	20	729	4
RefDAVIS	18	729	4
ReVOS	25	729	4

Token Pruning and Fixed Tokens. To evaluate the impact of token pruning, we test four pruning settings with $V' \in \{729, 146, 73, 36\}$ visual tokens retained per frame, corresponding to pruning ratios from 0% to 95%. For video data, the total number of visual tokens is $V' \times F$. Each input also includes a fixed set of $T_{\text{fixed}} = 100$ auxiliary tokens (e.g., prompts, [MASK], [CLS]), resulting in a total sequence length:

$$S = T + T_{\text{fixed}} + V'.$$

FLOPs Computation. We compute FLOPs for each component using closed-form expressions based on transformer operations. Nonlinear activations, biases, and normalization layers are omitted for simplicity. The estimates are as follows:

- **Language model:** For L layers, hidden size d, and intermediate FFN dimension d_{int}:

$$\begin{aligned}\text{FLOPs}_{\text{attn}} &= 3Sd^2 + 2S^2d + Sd^2,\\ \text{FLOPs}_{\text{ffn}} &= 2Sd \cdot d_{\text{int}},\\ \text{FLOPs}_{\text{LM}} &= L(\text{attn} + \text{ffn}) + Sd|\mathcal{V}|.\end{aligned}$$

- **Vision encoder:** Given N patches, hidden size d_v, and L_v layers:

$$\text{FLOPs}_{\text{vision}} = \sum_{\ell=1}^{L_v} \left(6Nd_v^2 + 2N^2d_v\right) + Nd_vd.$$

- **Token pruning:** For input V and pruned V' tokens:

$$\text{FLOPs}_{\text{prune}} = 2Vd + \frac{VV'd}{10} + V'd.$$

- **Mask decoder:** Using Q object queries, hidden dim d_m, and L_d layers:

$$\begin{aligned}\text{FLOPs}_{\text{mask}} = L_d \cdot (&12Qd_m^2 + 2Q^2d_m + 2QV'd_m)\\ &+ Qd_mV'.\end{aligned}$$

- **Temporal reasoning:** With Q_t temporal queries, F frames, and L_t layers:

$$\text{FLOPs}_{\text{temporal}} = L_t(Q_tFd^2 + 4Q_td^2).$$

- **VMTF fusion:** For fusion dim d_f and L_f layers:

$$\text{FLOPs}_{\text{vmtf}} = L_f(T_{\text{eff}}dd_f + V'd_vd_f + 2T_{\text{eff}}V'd_f).$$

Each value corresponds to a single forward pass. For every configuration, we compute FLOPs under both unpruned ($V' = V$) and pruned settings, and report the reduction percentage.

Model Hyperparameters. All FLOPs estimates are based on the default InstructSeg configuration:

- **Language model:** $d = 2560$, $d_{\text{int}} = 10240$, $L = 32$ layers, vocabulary size $|\mathcal{V}| = 51200$.
- **Vision encoder:** $d_v = 1152$, $N = 729$ patches from 384×384 input, $L_v = 27$ layers.
- **Mask decoder:** $Q = 100$ queries, $d_m = 256$, $L_d = 9$ layers.
- **Temporal module:** $Q_t = 128$ queries, $L_t = 3$ layers, $F = 4$ reference frames.
- **VMTF fusion:** $d_f = 1024$, $L_f = 3$ layers.
- **Token pruning:** Hidden dim $d = 2560$.
- **Fixed tokens:** $T_{\text{fixed}} = 100$.

These settings are used consistently across all datasets without per-task tuning.

Table 2. Task-specific language instructions used for each instructed segmentation task.

Task	Visual Type	Dataset	Instruction Template	Example Text Prompt
Referring Expression Segmentation	Image	RefCOCO / + / g	*You need to perform Referring Expression Segmentation on the image according to the Text Prompt.*	*"A baseball catcher with an open mitt"*
Reasoning Segmentation	Image	ReasonSeg	*You need to perform Reasoning Segmentation on the image according to the Text Prompt.*	*"The person who appears to have already won in the battle"*
Referring Video Object Segmentation	Video	Ref-YouTube-VOS, etc.	*You need to perform Referring Video Object Segmentation on the video according to the Text Prompt.*	*"A duck is held by a person with her both hands"*
Reasoning Video Object Segmentation	Video	ReVOS	*You need to perform Reasoning Video Object Segmentation on the video according to the Text Prompt.*	*"Which person is in the leading position?"*

A.3. Instruction Templates and Dataset Construction

Following *InstructSeg* [9], we use task-specific instruction templates to construct language prompts. Each segmentation task is paired with a distinct instruction format based on its modality and reasoning type. These templates remain consistent across training and evaluation. Table 5 summarizes the instruction settings and representative examples.

Baseline Pruning Methods

We compare with several state-of-the-art token pruning methods designed for general-purpose vision-language models. Specifically, we include ToM [2], VisionZip [10], VisPruner [12], and DivPrune [1], all of which are compatible with frozen architectures and do not require retraining.

To ensure fair comparison, we unify the application point for all methods: visual token pruning is applied at the output of the SigLIP encoder. This allows consistent input representation into the language model. For attention-based methods (e.g., VisionZip), we extract attention weights directly from SigLIP. This differs slightly from the original setup, which typically relies on CLIP, but preserves each algorithm core mechanism. Since these pruning method were originally designed for Visual understanding tasks such as VQA and used primarily in LLaVA or LLama architecture MLLM, minor adaptation is necessary to evaluate them in the IVS based on MLLMs setting. We exclude other pruning baseline methods for several reasons:

- **Retraining-dependent methods**, such as TokenPacker [7] and M3 [3], require end-to-end optimization and introduce structural changes, which are incompatible with frozen models.
- **Temporal-incompatible designs**, such as FastV [4], as-

sume architectures tailored to video encoders, which differ significantly from our setup.

In summary, we focus on general, transferable, one-shot pruning methods that can be fairly applied to the unified visual token stream without requiring architecture-specific or fine-tuning.

B. Visualization and Experiment Result Analysis

B.1. Image-Based IVS Pruning Analysis

To better understand how token pruning affects segmentation quality in practice, we examine qualitative results using only 20% of the visual tokens. As shown in Figure 1, we compare outputs from our pruned model with those generated using the full token set across a range of visual inputs and prompts. In the top examples with blue prompt bubbles, the model performs well despite heavy pruning. For instance, in the basketball image, the player is clearly segmented with accurate boundaries. Similarly, in both dog-related prompts, the model correctly identifies the dog with limited information. These results suggest that when objects are visually prominent and well separated from the background, a small number of well-distributed tokens is often enough to support good segmentation. In the middle group, marked with orange bubbles, the segmentation is generally correct but misses some details. For example, in the subway handle image, several handles are not segmented. In the wrestling example, parts of the bodies are either incomplete or blurred. These cases indicate that while the model still captures the main object, it struggles with fine structures when fewer tokens are available. The last row, highlighted in red, shows a failure case. The prompt asks for a "dog with its mouth open," but the model only segments the larger black dog, completely missing the smaller beige dog that more closely matches the prompt. This likely results from insufficient token coverage in that region, suggesting that static selection may not adapt well to scenes with multiple similar objects or complex layouts.

Future pruning method could benefit from combining spatial coverage with semantic information, such as attention or uncertainty signals, to better identify which regions matter. Adapting the token budget based on input complexity or using multi-scale sampling might also help retain more useful information. Overall, while static pruning works well in many settings, more adaptive methods may be needed for robust performance across diverse visual inputs.

B.2. Video-Based IVS Pruning Analysis

From previous experiment(Refer to table 2 and 3 in main manuscript), we find that pruning visual tokens to 20% introduces relatively minor performance degradation in video-based IVS comparing to image-based IVS. This observation prompts a closer examination of the underlying factors contributing to this robustness. To this end, we analyze qualitative results in Figure 2, comparing segmentation outputs from models using full and pruned token sets across representative video frames. Our findings suggest that the pruned model preserves segmentation quality despite the significant reduction in token count. In certain instances, pruning even improves boundary sharpness by suppressing irrelevant or redundant visual information and also shown better segmentation performance. This effect is particularly noticeable in scenes characterized by static backgrounds or repetitive textures, where token reduction helps eliminate noise that may otherwise interfere with accurate segmentation. A key factor contributing to this resilience is the high degree of temporal redundancy inherent in video data. Consecutive frames often share similar visual content, enabling the model to leverage temporal continuity to maintain spatial coherence and semantic consistency across frames. As a result, the model can tolerate missing or reduced spatial cues in individual frames by drawing upon context from neighboring frames. This temporal smoothing effect mitigates the negative impact of token pruning and makes video a particularly suitable modality for token pruning-efficient modeling. Nevertheless, the limitations of the current approach become evident in more challenging scenarios. As shown in Figure 3, the method struggles with rapid motion and fine-grained object discrimination. In the first example, fast-moving objects such as motorcycles exhibit motion blur and large frame-to-frame variation, which disrupt the model's ability to consistently retain relevant tokens. In the second example, dense scenes involving multiple, visually similar objects (e.g., fish) pose difficulty for token selection strategies that prioritize spatial coverage over semantic distinctiveness.

These observations suggest that future pruning strategies should move beyond static heuristics and incorporate temporal dynamics, motion sensitivity, and object-level saliency into the selection process. Learning-based token prioritization mechanisms, possibly informed by task objectives or attention distributions, may offer a more adaptive and effective approach for maintaining segmentation performance under constrained token budgets.

B.3. Dataset Comparison: DAVIS vs. YouTube-VOS

Our main experiments (main manuscript) reveal that pruning leads to a larger performance drop on Refer-DAVIS17 compared to Refer-YouTube-VOS, even under the same token retention ratio. To better understand this discrepancy, we summarize key dataset statistics in Table 6, focusing on factors that may influence pruning robustness. Refer-DAVIS17 features shorter sequences with dense frame-level annotations and highly focused object categories. Each ob-

Table 3. Comparison of Refer-DAVIS17 and Refer-YouTube-VOS datasets.

Statistic	Refer-DAVIS17	Refer-YouTube-VOS
Number of Videos	90	3975
Number of Objects	205	7451
Number of Expressions	1544	27899
Avg. Objects per Video	2.3	1.87
Avg. Expressions per Object	7.53	3.74
Avg. Frames per Sequence	67.4	126.0
Annotation Density	Dense (every frame)	Sparse (5 fps)
Category Diversity	Low (e.g., people, dogs)	High (animals, vehicles, tools)

ject is described by multiple referring expressions and segmented in every frame, making fine-grained localization critical. Consequently, pruning even a small number of important tokens can noticeably affect segmentation boundaries or object completeness. In contrast, Refer-YouTube-VOS consists of longer, more diverse videos with sparser annotations and greater temporal redundancy. This structure provides the model with more contextual support across frames, allowing it to compensate for missing details caused by pruning. These properties make YouTube-VOS more resilient to token reduction.

Overall, the differences in temporal structure, annotation density, and content diversity help explain the dataset-specific behavior observed in our experiments. These insights also motivate future work on dataset-aware pruning strategies that can adapt to the demands of different video understanding tasks.

C. Implementation Details

C.1. Baseline Model: *InstructSeg*

We adopt *InstructSeg* [9] as the baseline model throughout our experiments. It is a modular multi-modal architecture designed for instruction-driven segmentation tasks over both images and videos, supporting various scenarios such as referring expression segmentation and reasoning-based segmentation. The model consists of a set of fixed pre-trained components, each responsible for a distinct stage of visual-language processing, including visual encoding, token fusion, temporal reasoning, and mask decoding. No modules are fine-tuned during inference.

Language Model. The core reasoning component is a 2.7B-parameter PHI-2 [6] language model. It receives tokenized inputs including text instructions, visual tokens, reference frame tokens (for videos), and learnable mask tokens. These are concatenated and processed jointly to produce contextualized embeddings that guide the subsequent segmentation modules. The language model is adapted using Low-Rank Adaptation (LoRA) with a rank of 8. We directly use the publicly released LoRA-tuned weights without further training.

Visual Encoders and Token Pruning. *InstructSeg* employs two vision backbones. First, a SIGLIP encoder [11] is used to extract global visual tokens from each input image or frame. These tokens are aligned with text tokens through the language model. To reduce computational overhead and redundancy, we apply several token pruning methods directly to the SigLIP output. This approach retains important semantic information while reducing the number of visual tokens passed into the language model, leading to more efficient inference. In addition, a SWIN-B transformer [8] is used to extract high-resolution spatial features. These features are not pruned and are passed directly to the segmentation decoder to support fine-grained mask prediction.

Segmentation Decoder. For generating pixel-level predictions, we use a pre-trained MASK2FORMER [5] decoder. It combines a pixel decoder with a transformer-based instance decoder to predict N mask proposals. Each proposal is associated with an embedding and a confidence score. The most relevant masks are selected by computing the similarity between predicted mask embeddings and instruction-guided features.

Object-aware Video Perceiver. To support temporal reasoning in video tasks, the model includes an Object-aware Video Perceiver (OVP) module. This component processes a sequence of $T_r = 4$ reference frames alongside the instruction tokens. Cross-attention and $N_1 = 3$ transformer layers are used to extract compressed reference tokens that encode object-specific dynamics over time. These tokens are integrated into the language model to provide temporal context for segmentation.

Vision-guided Text Fusion. To better connect the instruction semantics with the visual scene, a Vision-guided Multi-granularity Text Fusion (VMTF) module is used after the language model. It fuses coarse and fine-grained instruction embeddings with spatial visual features using $N_2 = 3$ cross-attention layers. The resulting embeddings act as dynamic classifiers for the segmentation decoder.

Inference Pipeline. During inference, the model receives an image or key frame $\mathcal{V} \in \mathbb{R}^{H \times W \times 3}$, an optional reference sequence $\mathcal{V}_r$ (for videos), and a natural language instruction $\mathcal{E}$. Visual and reference tokens are extracted and pruned (when applicable), then combined with instruction and mask tokens and passed through the language model. Refined embeddings from VMTF are used to produce final segmentation masks. All components are pre-trained and remain fixed. Architectural hyperparameters such as the number of queries or attention layers follow the default configuration in the original implementation.

InstructSeg provides a strong and flexible baseline for instruction-conditioned segmentation. Its modular design and compatibility with pruning make it a suitable testbed for analyzing token efficiency in visual-language models.

C.2. TFLOPs Estimation Setup

To estimate computational cost, we implement a FLOPs calculator that evaluates the number of floating-point operations (FLOPs) for each major component in InstructSeg. We measure total TFLOPs (in units of 10^{12} FLOPs) for a single forward pass. The analysis includes the language model, visual encoder, token pruning module, mask decoder, temporal reasoning layers, and the fusion module (VMTF).

Dataset Inputs. We evaluate FLOPs on both image-based and video-based segmentation datasets. For each dataset, we define the number of text tokens T, visual tokens per frame V, and the number of frames F. Table 4 summarizes the token configuration for each benchmark:

Table 4. Input configurations for FLOPs estimation.

Dataset	Text Tokens (T)	Visual Tokens (V)	Frames (F)
RefCOCO	15	729	1
RefCOCO+	15	729	1
RefCOCOg	23	729	1
MM-Conv	50	729	1
ReasonSeg	80	729	1
RefYouTube	20	729	4
RefDAVIS	18	729	4
ReVOS	25	729	4

Token Pruning and Fixed Tokens. To evaluate the impact of token pruning, we test four pruning settings with $V' \in \{729, 146, 73, 36\}$ visual tokens retained per frame, corresponding to pruning ratios from 0% to 95%. For video data, the total number of visual tokens is $V' \times F$. Each input also includes a fixed set of $T_{\text{fixed}} = 100$ auxiliary tokens (e.g., prompts, [MASK], [CLS]), resulting in a total sequence length:

$$S = T + T_{\text{fixed}} + V'.$$

FLOPs Computation. We compute FLOPs for each component using closed-form expressions based on transformer operations. Nonlinear activations, biases, and normalization layers are omitted for simplicity. The estimates are as follows:

- **Language model:** For L layers, hidden size d, and intermediate FFN dimension d_{int}:

$$\begin{aligned}\text{FLOPs}_{\text{attn}} &= 3Sd^2 + 2S^2d + Sd^2,\\ \text{FLOPs}_{\text{ffn}} &= 2Sd \cdot d_{\text{int}},\\ \text{FLOPs}_{\text{LM}} &= L(\text{attn} + \text{ffn}) + Sd|\mathcal{V}|.\end{aligned}$$

- **Vision encoder:** Given N patches, hidden size d_v, and L_v layers:

$$\text{FLOPs}_{\text{vision}} = \sum_{\ell=1}^{L_v} \left(6Nd_v^2 + 2N^2d_v\right) + Nd_vd.$$

- **Token pruning:** For input V and pruned V' tokens:

$$\text{FLOPs}_{\text{prune}} = 2Vd + \frac{VV'd}{10} + V'd.$$

- **Mask decoder:** Using Q object queries, hidden dim d_m, and L_d layers:

$$\begin{aligned}\text{FLOPs}_{\text{mask}} = L_d \cdot (12Qd_m^2 + 2Q^2d_m + 2QV'd_m)\\ + Qd_mV'.\end{aligned}$$

- **Temporal reasoning:** With Q_t temporal queries, F frames, and L_t layers:

$$\text{FLOPs}_{\text{temporal}} = L_t(Q_tFd^2 + 4Q_td^2).$$

- **VMTF fusion:** For fusion dim d_f and L_f layers:

$$\text{FLOPs}_{\text{vmtf}} = L_f(T_{\text{eff}}dd_f + V'd_vd_f + 2T_{\text{eff}}V'd_f).$$

Each value corresponds to a single forward pass. For every configuration, we compute FLOPs under both unpruned ($V' = V$) and pruned settings, and report the reduction percentage.

Model Hyperparameters. All FLOPs estimates are based on the default InstructSeg configuration:

- **Language model:** $d = 2560$, $d_{\text{int}} = 10240$, $L = 32$ layers, vocabulary size $|\mathcal{V}| = 51200$.
- **Vision encoder:** $d_v = 1152$, $N = 729$ patches from 384×384 input, $L_v = 27$ layers.
- **Mask decoder:** $Q = 100$ queries, $d_m = 256$, $L_d = 9$ layers.
- **Temporal module:** $Q_t = 128$ queries, $L_t = 3$ layers, $F = 4$ reference frames.
- **VMTF fusion:** $d_f = 1024$, $L_f = 3$ layers.
- **Token pruning:** Hidden dim $d = 2560$.
- **Fixed tokens:** $T_{\text{fixed}} = 100$.

These settings are used consistently across all datasets without per-task tuning.

Table 5. Task-specific language instructions used for each instructed segmentation task.

Task	Visual Type	Dataset	Instruction Template	Example Text Prompt
Referring Expression Segmentation	Image	RefCOCO / + / g	*You need to perform Referring Expression Segmentation on the image according to the Text Prompt.*	*"A baseball catcher with an open mitt"*
Reasoning Segmentation	Image	ReasonSeg	*You need to perform Reasoning Segmentation on the image according to the Text Prompt.*	*"The person who appears to have already won in the battle"*
Referring Video Object Segmentation	Video	Ref-YouTube-VOS, etc.	*You need to perform Referring Video Object Segmentation on the video according to the Text Prompt.*	*"A duck is held by a person with her both hands"*
Reasoning Video Object Segmentation	Video	ReVOS	*You need to perform Reasoning Video Object Segmentation on the video according to the Text Prompt.*	*"Which person is in the leading position?"*

C.3. Instruction Templates and Dataset Construction

Following *InstructSeg* [9], we use task-specific instruction templates to construct language prompts. Each segmentation task is paired with a distinct instruction format based on its modality and reasoning type. These templates remain consistent across training and evaluation. Table 5 summarizes the instruction settings and representative examples.

Baseline Pruning Methods

We compare with several state-of-the-art token pruning methods designed for general-purpose vision-language models. Specifically, we include ToM [2], VisionZip [10], VisPruner [12], and DivPrune [1], all of which are compatible with frozen architectures and do not require retraining.

To ensure fair comparison, we unify the application point for all methods: visual token pruning is applied at the output of the SigLIP encoder. This allows consistent input representation into the language model. For attention-based methods (e.g., VisionZip), we extract attention weights directly from SigLIP. This differs slightly from the original setup, which typically relies on CLIP, but preserves each algorithm core mechanism. Since these pruning method were originally designed for Visual understanding tasks such as VQA and used primarily in LLaVA or LLama architecture MLLM, minor adaptation is necessary to evaluate them in the IVS based on MLLMs setting. We exclude other pruning baseline methods for several reasons:

- **Retraining-dependent methods**, such as TokenPacker [7] and M3 [3], require end-to-end optimization and introduce structural changes, which are incompatible with frozen models.
- **Temporal-incompatible designs**, such as FastV [4], as-

Figure 1. Qualitative results under 20% visual token pruning and comparing with using original 100% visual token. Blue prompt boxes indicate successful cases where segmentation closely matches the full-token model. Orange-yellow boxes denote partial successes where overall object segmentation is preserved but fine-grained details are lost or imprecise. Red boxes represent failure cases, often involving multiple objects or complex spatial layouts, where the model fails to accurately localize or identify the target object.

sume architectures tailored to video encoders, which differ significantly from our setup.

In summary, we focus on general, transferable, one-shot pruning methods that can be fairly applied to the unified visual token stream without requiring architecture-specific or fine-tuning.

D. Visualization and Experiment Result Analysis

D.1. Image-Based IVS Pruning Analysis

To better understand how token pruning affects segmentation quality in practice, we examine qualitative results using only 20% of the visual tokens. As shown in Figure 1, we

Figure 2. Video segmentation results using 20% visual token pruning. Each example compares our pruned model (top row in each pair) with the original model using 100% tokens (bottom row). In the first sequence, showing a person and a bicycle, pruning reduces clutter and leads to better segmentation in some frames. In the second sequence, which includes multiple people in a crowd, the pruned model produces better spatial separation between foreground and background.

compare outputs from our pruned model with those generated using the full token set across a range of visual inputs and prompts. In the top examples with blue prompt bubbles, the model performs well despite heavy pruning. For instance, in the basketball image, the player is clearly segmented with accurate boundaries. Similarly, in both dog-related prompts, the model correctly identifies the dog with limited information. These results suggest that when objects are visually prominent and well separated from the background, a small number of well-distributed tokens is often enough to support good segmentation. In the middle group, marked with orange bubbles, the segmentation is generally correct but misses some details. For example, in the subway handle image, several handles are not segmented. In the wrestling example, parts of the bodies are either incomplete or blurred. These cases indicate that while the model still captures the main object, it struggles with fine structures when fewer tokens are available. The last row, highlighted in red, shows a failure case. The prompt asks for a "dog with its mouth open," but the model only segments the larger black dog, completely missing the smaller beige dog that more closely matches the prompt. This likely results from insufficient token coverage in that region, suggesting that static selection may not adapt well to scenes with multiple similar objects or complex layouts.

Future pruning method could benefit from combining spatial coverage with semantic information, such as attention or uncertainty signals, to better identify which regions matter. Adapting the token budget based on input complexity or using multi-scale sampling might also help retain more useful information. Overall, while static pruning works well in many settings, more adaptive methods may be

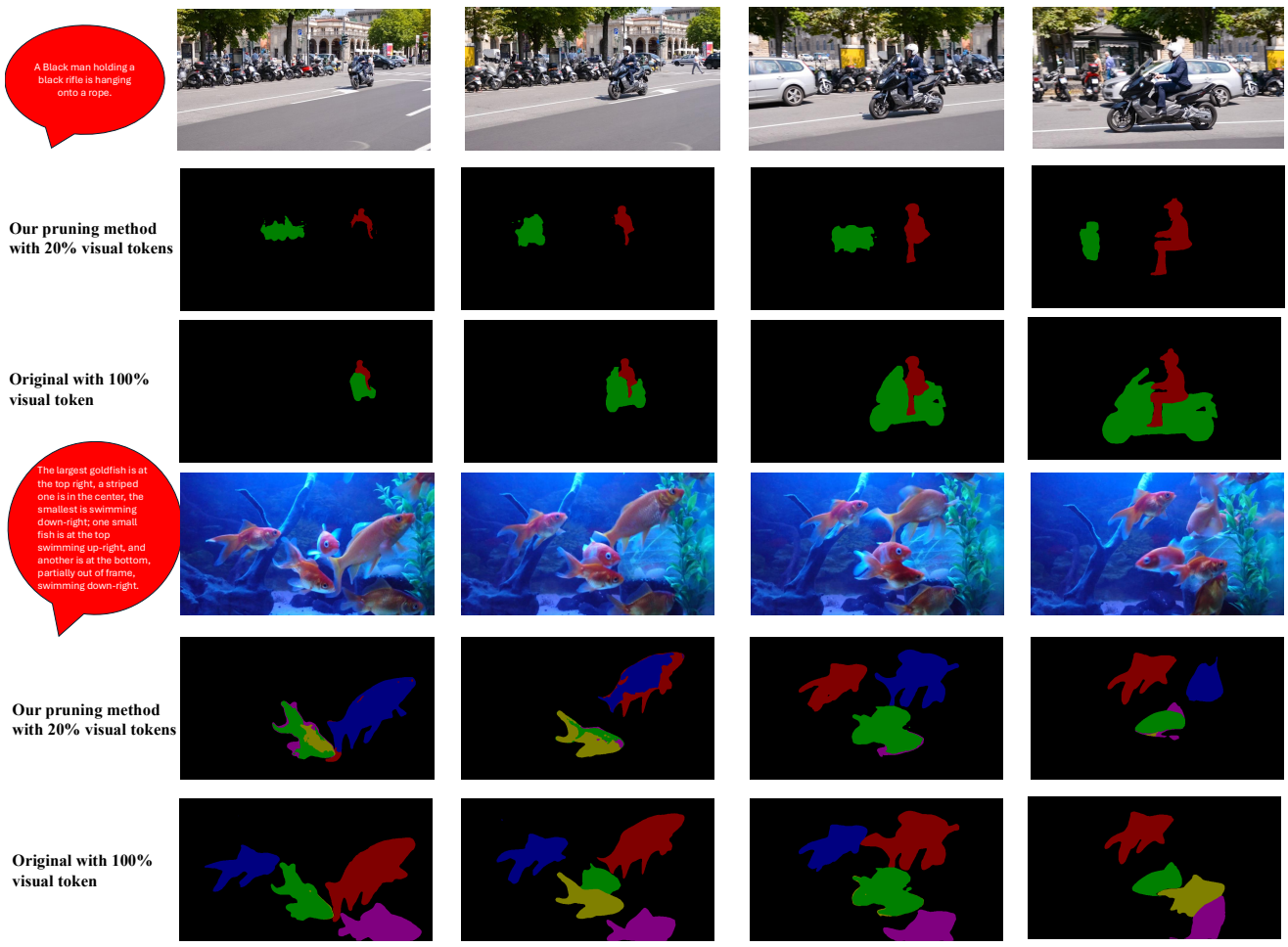

Figure 3. Failure cases of our visual token pruning method under 20% token budget. In the top sequence, involving fast-moving motorcycles, the model fails to consistently segment both the rider and the vehicle, likely due to motion blur and temporal misalignment. In the bottom sequence, the scene contains multiple fish with highly similar appearances. The pruned model struggles to preserve instance separation and fails to segment all targets accurately. These examples illustrate that pruning method can underperform in cases of rapid motion or when many visually similar objects appear in the same frame.

needed for robust performance across diverse visual inputs.

D.2. Video-Based IVS Pruning Analysis

From previous experiment(Refer to table 2 and 3 in main manuscript), we find that pruning visual tokens to 20% introduces relatively minor performance degradation in video-based IVS comparing to image-based IVS. This observation prompts a closer examination of the underlying factors contributing to this robustness. To this end, we analyze qualitative results in Figure 2, comparing segmentation outputs from models using full and pruned token sets across representative video frames. Our findings suggest that the pruned model preserves segmentation quality despite the significant reduction in token count. In certain instances, pruning even improves boundary sharpness by suppressing irrelevant or redundant visual information and also shown better segmentation performance. This effect is particularly noticeable in scenes characterized by static backgrounds or repetitive textures, where token reduction helps eliminate noise that may otherwise interfere with accurate segmentation. A key factor contributing to this resilience is the high degree of temporal redundancy inherent in video data. Consecutive frames often share similar visual content, enabling the model to leverage temporal continuity to maintain spatial coherence and semantic consistency across frames. As a result, the model can tolerate missing or reduced spatial cues in individual frames by drawing upon context from neighboring frames. This temporal smoothing effect mitigates the negative impact of token pruning and makes video a particularly suitable modality for token pruning-efficient modeling. Nevertheless, the limitations of the current approach become evident in more challenging scenarios. As

Table 6. Comparison of Refer-DAVIS17 and Refer-YouTube-VOS datasets.

Statistic	Refer-DAVIS17	Refer-YouTube-VOS
Number of Videos	90	3975
Number of Objects	205	7451
Number of Expressions	1544	27899
Avg. Objects per Video	2.3	1.87
Avg. Expressions per Object	7.53	3.74
Avg. Frames per Sequence	67.4	126.0
Annotation Density	Dense (every frame)	Sparse (5 fps)
Category Diversity	Low (e.g., people, dogs)	High (animals, vehicles, tools)

shown in Figure 3, the method struggles with rapid motion and fine-grained object discrimination. In the first example, fast-moving objects such as motorcycles exhibit motion blur and large frame-to-frame variation, which disrupt the model's ability to consistently retain relevant tokens. In the second example, dense scenes involving multiple, visually similar objects (e.g., fish) pose difficulty for token selection strategies that prioritize spatial coverage over semantic distinctiveness.

These observations suggest that future pruning strategies should move beyond static heuristics and incorporate temporal dynamics, motion sensitivity, and object-level saliency into the selection process. Learning-based token prioritization mechanisms, possibly informed by task objectives or attention distributions, may offer a more adaptive and effective approach for maintaining segmentation performance under constrained token budgets.

D.3. Dataset Comparison: DAVIS vs. YouTube-VOS

Our main experiments (main manuscript) reveal that pruning leads to a larger performance drop on Refer-DAVIS17 compared to Refer-YouTube-VOS, even under the same token retention ratio. To better understand this discrepancy, we summarize key dataset statistics in Table 6, focusing on factors that may influence pruning robustness. Refer-DAVIS17 features shorter sequences with dense frame-level annotations and highly focused object categories. Each object is described by multiple referring expressions and segmented in every frame, making fine-grained localization critical. Consequently, pruning even a small number of important tokens can noticeably affect segmentation boundaries or object completeness. In contrast, Refer-YouTube-VOS consists of longer, more diverse videos with sparser annotations and greater temporal redundancy. This structure provides the model with more contextual support across frames, allowing it to compensate for missing details caused by pruning. These properties make YouTube-VOS more resilient to token reduction.

Overall, the differences in temporal structure, annotation density, and content diversity help explain the dataset-specific behavior observed in our experiments. These insights also motivate future work on dataset-aware pruning strategies that can adapt to the demands of different video understanding tasks.

References

[1] Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. Divprune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 9392–9401, 2025. 3, 7

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 3, 7

[3] Mu Cai, Jianwei Yang, Jianfeng Gao, and Yong Jae Lee. Matryoshka multimodal models. In *Workshop on Video-Language Models@ NeurIPS 2024*, 2024. 3, 7

[4] 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*, pages 19–35. Springer, 2024. 3, 7

[5] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1290–1299, 2022. 1, 5

[6] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1:3, 2023. 1, 5

[7] Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. *International Journal of Computer Vision*, pages 1–19, 2025. 3, 7

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1, 5

[9] Cong Wei, Yujie Zhong, Haoxian Tan, Yingsen Zeng, Yong Liu, Zheng Zhao, and Yujiu Yang. Instructseg: Unifying instructed visual segmentation with multi-modal large language models. *ICCV 2025*, 2025. 1, 3, 5, 7

[10] 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 Computer Vision and Pattern Recognition Conference*, pages 19792–19802, 2025. 3, 7

[11] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023. 1, 5

[12] Qizhe Zhang, Aosong Cheng, Ming Lu, Renrui Zhang, Zhiyong Zhuo, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. Beyond text-visual attention: Exploiting vi-

sual cues for effective token pruning in vlms. *arXiv preprint arXiv:2412.01818*, 2024. 3, 7