Supplementary Materials for *IVTP: Instruction-guided Visual Token Pruning for Large Vision-Language Models*

Kai Huang*, Hao Zou*, Ye Xi, BoChen Wang, Zhen Xie, and Liang Yu

Alibaba Group, China
{zhouwan.hk, zh372956, yx150449, bochen.wbc, xiezhen.xz, liangyu.yl}@alibaba-inc.com

1 Limitation Statement

We identify the following two shortcomings in our method, which will also be the focal points for future improvements: 1) The second stage of our proposed two-stage visual token pruning method for VLMs is somewhat reliant on the aggregation of visually associated semantic information from instructions. The operation facilitated by the CLIP text encoder as a intermediary is susceptible to the influence of the types of ViT and LLM, necessitating that the ViT is CLIP-based and the LLM is LLaMA-based. 2) Although the proposed method can reduce the training computational complexity of VLMs by over 40%, the cost of model remains high due to the vast amount of training data and model parameters. We do not demonstrate the application of the proposed method to a wider range of image-based VLMs or video VLMs, which would further prove the adaptability of the method.

2 Implementation Details

Following LLaVA-1.5, the training of VLMs is divided into two steps. The first step is cross-modal alignment of text and image, which is called pretraining, aimed at learning the mapping of visual embedding to language embedding. During this process, the ViT and LLM will be frozen, and only the parameters of the multimodal projection layer are optimized. The training data are all image-caption pairs, with a learning rate set to 1e-3 and a batch size of 256. The second step is instruction tuning, which primarily focuses on learning the ability to follow instructions. During this process, only the ViT is frozen, and the parameters of the multimodal projection layer and the LLM are optimized. The training data consist of a widely collected set of VQA examples, with a learning rate set to 2e-5 and a batch size of 128. Since the computational complexity of the LLM plays a decisive role in the entire VLMs, compared to those methods that only perform token pruning in the ViT or between the ViT and LLM, our proposed two-stage approach needs to pay particular attention to the

* Equal contribution.

impact brought by the additional visual tokens that need to be pruned in the LLM. On one hand, we carefully control the number of layers in the LLM that participate in token pruning, that is, by using only the first 12 layers instead of all layers, to ensure that the number of visual tokens is reduced to the target count as quickly as possible. On the other hand, when the target number of visual tokens is above 128, we still prune the same number of tokens each time. Whereas when the target number is less than 128 (assume n tokens), we reduce the number of visual tokens to $2n$ in the ViT, and subsequently down to n in the LLM. In this way, we can achieve better model accuracy while ensuring that the computational complexity remains comparable to that of single-stage methods. We further demonstrate the implementation details of the following comparative methods on the LLaVA-1.5 architecture.

Random sampling. In the ViT, the same number of visual tokens is randomly pruned in each layer.

TopK. The visual CLS token is used to compute the cosine similarity with all other patch tokens. The tokens are then sorted, and a certain number of those with the lowest similarity scores are pruned. This process is repeated in each layer.

Spatial pooling. Before the visual tokens are input into the LLM, the sequence of tokens is resized to a three-dimensional matrix by means of two-dimensional spatial interpolation, and then stretched back into a sequence. It is important to note that since spatial interpolation can only accept integer values, it cannot precisely control the final length of the sequence.

EViT [6]. We calculate CLS attentiveness in each layer of the ViT to identify the top-k attentive tokens and fuse the inattentive tokens. Ultimately, the visual tokens consist of n-1 attentive tokens and one fused inattentive token.

ToMe [2]. We also perform bipartite soft matching in each layer of the ViT, merging a fixed number of visual tokens to reach the final number of visual tokens.

Honeybee [3]. It proposes two similar token pruning structures, which are called the Convolutional Abstractor (C-Abstractor) and the Deformable Attention-based Abstractor (D-Abstractor). From their experimental results, it can be observed that the performance of the C-Abstractor is slightly superior to that of the D-Abstractor. Therefore, we only apply the C-Abstractor to LLaVA-1.5. The final number of visual tokens is determined by controlling the number of layers in the ResNet block and adaptive average pooling within this structure.

LLaMA-VID [5]. We downsample the visual tokens to the final number of visual tokens through spatial pooling. However, because the visual information and instruction information are fused into one context token through context attention, the final number of visual tokens is $n + 1$.

QWen-VL [1]. Since its token pruning structure consists of a single-layer cross-attention, which is similar to the qformer in BLIP2 [4], we control the number of learnable queries to match the final number of visual tokens.

3 Pseudo Code of IVTP

The complete process of IVTP is illustrated in Algorithm 1 and Algorithm 2.

Algorithm 1 Token pruning with attention rollout

Initialize attention rollout matrix of l_2 layer, $\tilde{\mathbf{A}}^{l} \leftarrow \mathbf{I}$.
for $l = l_1 ... l_2$ **do**
    $\mathbf{A}^{l} \leftarrow$ Mean aggregation the multi-head aggention weights
    $\tilde{\mathbf{A}}^{l} \leftarrow$ Perform attention rollout with Eq. 3
$\tilde{\mathbf{A}}^{l_2}_{\text{cls}} \leftarrow$ Get CLS row from $\tilde{\mathbf{A}}^{l_2}$
$\mathcal{S}^{l_2} \leftarrow$ Obtain the importance scores according to Eq. 4
$\mathbf{h}^{l_2} \leftarrow$ Update the hidden states with pruned tokens
$\mathbf{A}^{l_2}_{\text{mask}} \leftarrow$ Update the attention mask with pruned tokens
return $\mathbf{h}^{l_2}, \mathbf{A}^{l_2}_{\text{mask}}$

Algorithm 2 Two-stage token pruning for LVLMs

$l_g \leftarrow$ number of layers in each group
$l_{\text{ViT}} \leftarrow$ number of layers used for token pruning in ViT
$l_{\text{LLM}} \leftarrow$ number of layers used for token pruning in LLM
for $l = 0 ... l_{\text{ViT}}$ and $l \% l_g == 0$ **do** ▷ Pruning in ViT
    $(\mathbf{h}^{l}, \mathbf{A}^{l}_{\text{mask}}) \leftarrow$ Prune tokens with Algorithm 1
$(T_{\text{LLM}}, \mathbf{X}^{L}) \leftarrow$ Tokenize instructions with LLM tokenizer
$(T_{\text{CLIP}}, \bar{\mathbf{X}}^{L}) \leftarrow$ Tokenize instructions with CLIP text encoder tokenizer
$c \leftarrow$ Calculate visual relevances with Eq. 6
$\mathcal{C} \leftarrow$ Map the relevances to LLM textual tokens with Eq. 7
$\hat{x}^{T}_{\text{cls}} \leftarrow$ Get textual pseudo CLS token with Eq. 8
$\mathbf{X} \leftarrow$ Organize vision-language tokens with Concat$(\mathbf{X}^{V}, \hat{x}^{T}_{\text{cls}}, \mathbf{X}^{L})$
for $l = 0 ... l_{\text{LLM}}$ and $l \% l_p == 0$ **do** ▷ Pruning in LLM
    $(\mathbf{h}^{l}, \mathbf{A}^{l}_{\text{mask}}) \leftarrow$ Prune tokens with Algorithm 1

4 Additional Experimental Results

Additional comparative results in LLaVA-1.5-13B. Table 1 and Figure 1 respectively present the changes in TFLOPs when varying the number of text tokens under the LLaMA-1.5-13B framework, as well as the trade-off between average accuracy and TFLOPs during pure inference. It can be observed that the overall trend is similar to that under the LLAMA-1.5-7B framework. The method we propose can reduce the overall computational complexity by approximately 34% to 72% as the number of text tokens changes. When the number of tokens

Table 1: Comparison results of TFLOPs across different methods as text token count varies from 128 to 1024.

Methods	128	256	512	1024
LLaVA-1.5-13B [7]	19.4	22.4	29.4	39.5
TopK	5.2 (-73.2%)	8.6 (-61.6%)	15.4 (-47.6%)	29.6 (-33.6%)
ToMe [2]	5.2 (-73.2%)	8.6 (-61.6%)	15.4 (-47.6%)	29.6 (-33.6%)
Honeybee [3]	5.4 (-72.2%)	8.7 (-61.2%)	15.4 (-47.6%)	29.6 (-33.6%)
Qwen-VL [1]	5.3 (-72.7%)	8.7 (-61.2%)	15.4 (-47.6%)	29.6 (-33.6%)
IVTP (Ours)	5.5 (-71.6%)	8.8 (-60.7%)	15.6 (-46.9%)	29.6 (-33.6%)

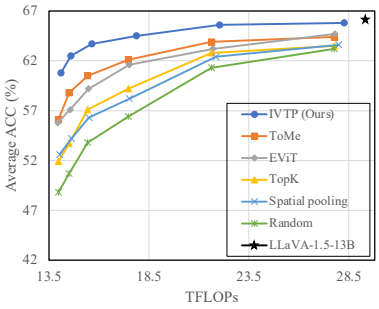

Fig. 1: Comparison of model performance with pure inference setting.

Table 2: Training and inference costs of LLaVA-1.5-7B with different visual token pruning methods.

Methods	MiniGPT-v2 (7B)				CogVLM (17B)				
	OKVQA	VizWiz	GQA	IconVQA	OKVQA	VizWiz	GQA	VQAv2	TextVQA
Original	57.8	60.1	53.6	51.5	64.7	76.4	65.2	84.7	69.7
TopK	50.2	52.6	43.8	47.5	57.0	68.8	56.6	78.9	63.5
ToMe	52.7	53.5	47.3	47.9	59.4	71.9	58.7	79.5	64.8
Ours	54.4	56.0	49.5	49.2	62.7	73.2	61.8	81.1	66.2

Table 3: Training and inference costs of LLaVA-1.5-7B with different visual token pruning methods.

Model	Pretraining		SFT		Inference Time
	GPU Memory	Time	GPU Memory	Time	
LLaVA-1.5-7B [7]	32.8G	3.47h	42.6G	11.21h	2.54h
Random sampling	17.7G	0.61h	37.4G	7.63h	1.74h
TopK	17.7G	0.66h	37.5G	7.59h	1.71h
Spatial pooling	17.7G	0.66h	37.5G	7.63h	1.75h
EViT [6]	17.7G	0.65h	37.5G	7.66h	1.77h
ToMe [2]	17.8G	0.66h	37.4G	7.64h	1.75h
Honeybee [3]	18.2G	0.68h	37.9G	7.68h	1.78h
LLaMA-VID [5]	19.5G	0.68h	38.8G	7.70h	1.82h
Qwen-VL [1]	17.8G	0.67h	37.3G	7.64h	1.74h
IVTP (Ours)	18.3G	0.71h	37.9G	7.96h	1.85h

drops from 576 to 256 in prue inference setting, there is only a 0.4% decrease in the average accuracy, demonstrating superior performance.

Extending to other LVLMs. Table 2 shows the comparative effects of the proposed method under pure inference mode when applied to other similar LVLMs. The vast majority of existing LVLMs follow a similar architecture, typically comprising a ViT-based visual encoder for visual tokenization, followed by feeding it along with language tokens into an LLM for multimodal understanding and gen-

Table 4: Comparative results with varying numbers of layers involved in visual token pruning within the LLM.

Layers	Avg. Acc (PI)	Avg. Acc	TFLOPs
3	57.0	60.3	8.0
6	59.2	61.2	8.1
12	60.4	62.0	8.2
18	60.7	62.2	8.4
24	60.6	62.2	8.7
30	60.7	62.3	8.9

Table 5: Comparative results with different numbers of visual tokens inputted into the LLM.

Layers	Avg. Acc (PI)	Avg. Acc	TFLOPs
320	60.6	62.4	9.2
256	60.7	62.4	9.0
192	60.6	62.1	8.7
160	60.3	61.9	8.4
128	60.4	62.0	8.2
96	59.3	61.2	8.2
80	58.6	60.4	8.1

eration. From the table, it can be seen that the proposed method still achieves similar advantages, further demonstrating its generalization capability.

Comparison of training and inference efficiency. Table 3 displays the memory consumption and time expenditure of different methods during the pretraining, SFT, and inference phase. It can be observed that visual token pruning is most effective during the pretraining phase of the VLM. Taking our proposed method as an example, memory consumption can be reduced by 44% and the actual time expenditure decreases by 80%. Due to the greater proportion of text tokens, and the need to fine-tune the LLM with its substantial share of training parameters, the reduction effect is less pronounced in the SFT phase, resulting in an approximate 29% decrease in actual time expenditure. During the inference process, since it involves single-instance batches, the proportion of the model's forward computation within the entire inference is affected. Despite this, there is still an approximate time saving of about 27% achieved compared with the original inference.

Number of token pruning layers in LLM. Considering the computational complexity of the model, we only perform visual token pruning on the first several layers of the LLM, with the default selection being the first 12 layers. Tables 4 explore the impact of using different numbers of layers for token pruning in the LLM. It can be seen that when fewer layers are used, the computational efficiency of the model also increases. Although this allows the visual tokens to quickly reach to the target number, the lack of hierarchical selection and information exchange between each tokens lead to a more noticeable gap in model accuracy. When more layers are used for token pruning, the improvement in model accuracy is quite limited, but the increase in computational complexity is more significant.

Intensity of token pruning at different stages. We further investigate the impact of different intensities of two-stage token pruning on the final model performance. Since the number of tokens reduced in each pruning operation within the ViT and LLM is fixed, we adjust the intensity of token pruning performed in the ViT and LLM by controlling the number of visual tokens input to the LLM. Taking the reduction of visual tokens from 576 to 64 as an example, when adopting a global average token pruning strategy, the number

Table 6: Experimental results with different relevance threshold values.

Threshold	Pretraining Acc	Avg. Acc (PI)	Avg. Acc
0.1	72.6	58.9	60.6
0.2	95.5	60.4	62.0
0.3	98.1	59.2	61.2
0.4	99.3	58.2	60.0

Table 7: Ablation experiments on textual pseudo CLS token.

Methods	Avg. Acc (PI)	Avg. Acc
IVTP	60.4	62.0
LLM agg. → CLIP agg.	57.3	59.5
hybrid → visual & textul	59.6	61.3
w/ pseudo CLS	60.3	62.1

of visual tokens input to the LLM is approximately 250, whereas with a $2n$ strategy it would be 128. Table 5 shows the comparison of model performance when the number of visual tokens input to the LLM varies from 80 to 320. It can be observed that although the first strategy offers a slight increase in average accuracy of about 0.4% compared to the second strategy, there is also an increase of 0.8 TFLOPs in computational complexity. This reflects that the second strategy has a better cost-performance ratio.

The impact of relevance threshold. We empirically set the relevance threshold τ used for filtering irrelevant text tokens at 0.2, which is referenced from the noise data filtering of image-text pairs based on CLIP similarity. Typically, a low τ could result in the generated textual pseudo CLS token carrying more noise, while increasing τ may lead to the omission of visually relevant text tokens. Table 6 lists the impact of using different relevance thresholds on model performance. Since the datasets used in the pretraining phase are caption datasets, the majority of the corresponding instructions do not have specific referents. Therefore, we label the samples filtered out by the threshold as 1 and those not filtered as 0, and we calculate the filtering accuracy during pretraining. The comparative experimental results shown in the table are essentially consistent with the above analysis. A lower threshold introduces more noise, leading to a decline in model accuracy. Conversely, a higher threshold results in the omission of textual semantic information, which also leads to suboptimal effects.

Additional studies on textual pseudo CLS. Table 7 presents additional investigations into the generation and utilization of textual pseudo CLS tokens. Given that the text tokens corresponding to instructions obtained from the CLIP text encoder inherently possess semantic coherence with visual features, we attempt to use pseudo CLS tokens aggregated from CLIP's text tokens instead of those from LLM. The results in the first row of the table indicate that such a substitution has a significant negative impact on the model. In contrast, aggregating pseudo CLS tokens through LLM allows for end-to-end optimization of the model with its text token hidden states. The pseudo CLS token obtained from CLIP remains fixed throughout and are difficult to adapt to the complex and variable visual language tasks. The second row of the table shows that we fuse the visual CLS token with the textual pseudo CLS token in a weighted manner to guide the token pruning in LLM using the hybrid CLS token. It is observed that there is a slight decline in model performance. This suggests that when the instructions involve more explicit visual objects, directly using

the generated textual pseudo CLS can better maintain the corresponding visual information. We also attempt to involve the aggregated pseudo CLS token as an actual token in the model training and inference. The results presented in the last row of the table indicate that this single token by itself did not have a significant impact on the model's performance.

5 Additional Visualizations

Figure 2 offers additional visualization examples, demonstrating the process of identifying attentive tokens. The images used for this illustration were randomly chosen from the benchmark datasets. These examples confirm that our IVTP can effectively handles a diverse range of instructions.

Fig. 2: Visualization of the visual token pruning results. Each sample, viewed from left to right, consists of the raw image, the token pruning by the TopK, and the token pruning with different questions by the proposed method, respectively.

References

1. Bai, J., Bai, S., Yang, S., Wang, S., Tan, S., Wang, P., Lin, J., Zhou, C., Zhou, J.: Qwen-vl: A frontier large vision-language model with versatile abilities. arXiv preprint arXiv:2308.12966 (2023) 2, 4
2. Bolya, D., Fu, C.Y., Dai, X., Zhang, P., Feichtenhofer, C., Hoffman, J.: Token merging: Your vit but faster. In: The Eleventh International Conference on Learning Representations (2022) 2, 4
3. Cha, J., Kang, W., Mun, J., Roh, B.: Honeybee: Locality-enhanced projector for multimodal llm. arXiv preprint arXiv:2312.06742 (2023) 2, 4
4. Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. arXiv preprint arXiv:2301.12597 (2023) 2
5. Li, Y., Wang, C., Jia, J.: Llama-vid: An image is worth 2 tokens in large language models. arXiv preprint arXiv:2311.17043 (2023) 2, 4
6. Liang, Y., Chongjian, G., Tong, Z., Song, Y., Wang, J., Xie, P.: Evit: Expediting vision transformers via token reorganizations. In: International Conference on Learning Representations (2021) 2, 4
7. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. arXiv preprint arXiv:2310.03744 (2023) 4