STViT: Semantic Tokens for Efficient Global and Local Vision Transformers

Anonymous authors
Paper under double-blind review

Abstract

The quadratic computational complexity to the number of tokens limits the practical applications of Vision Transformers (ViTs). Several works propose to prune redundant tokens to achieve efficient ViTs. However, these methods generally suffer from (i) dramatic accuracy drops, (ii) application difficulty in the local vision transformer, and (iii) non-general-purpose networks for downstream tasks. In this work, we propose a novel Semantic Token ViT (STViT), for efficient global and local vision transformers, which can also be revised to serve as backbone for downstream tasks. The semantic tokens represent cluster centers, and they are initialized by pooling image tokens in space and recovered by attention, which can adaptively represent global or local semantic information. Due to the cluster properties, a few semantic tokens can attain the same effect as vast image tokens, for both global and local vision transformers. For instance, only 16 semantic tokens on DeiT-(Tiny,Small,Base) can achieve the same accuracy with more than 100% inference speed improvement and nearly 60% FLOPs reduction; on Swin-(Tiny,Small,Base), we can employ 16 semantic tokens in each window to further speed it up by around 20% with slight accuracy increase. Besides great success in image classification, we also extend our method to video recognition. In addition, we design a STViT-R(ecover) network to restore the detailed spatial information based on the STViT, making it work for downstream tasks, which is powerless for previous token sparsification methods. Experiments demonstrate that our method can achieve competitive results compared to the original networks in object detection and instance segmentation, with over 30% FLOPs reduction for backbone.

1 Introduction

In contrast to standard Convolutional Neural Networks (CNNs) approaches which process images pixel-by-pixel, Vision Transformers (ViTs) (Dosovitskiy et al., 2021; Touvron et al., 2021a;b; Liu et al., 2021; Wu et al., 2021) treat an image as a sequence of patch/image tokens, and have shown promising performance in prevalent visual recognition scenarios. However, these superior performances do not come for free: the quadratic computational complexity to the number of image tokens limits their application in practice. Previous works (Song et al., 2021; Zong et al., 2021) have illustrated the large amount of redundancy in the image tokens and also shown the effect of filtering out unimportant tokens normally according to predefined scoring mechanism. However, these methods face the following challenges. Firstly, the predefined scoring mechanisms for filtering are generally imprecise. In Figure 1, on the left we visualize the class token values in different layers which are commonly used to score the token importance (Xu et al., 2022; Fayyaz et al., 2021; Liang et al., 2022). Different layers have different value distributions, thus using these imprecise scores for filtering would lead to unsatisfactory performance. For example, EViT (Liang et al., 2022) has an accuracy drop of 1.3% when saving 50% FLOPs on DeiT-S (Touvron et al., 2021a). Secondly, the remaining tokens do not distribute evenly in space any more, making them hard to work in local vision transformers[1]. Finally, large-scale token pruning tremendously damages the spatial structure and positional information, and causes difficulties when applied to downstream tasks, which they do not propose a solution to deal with.

[1]In this paper, we define the vision transformer with global self-attention (like DeiT) as global vision transformer and the vision transformer with local self-attention (like Swin) as local vision transformer.

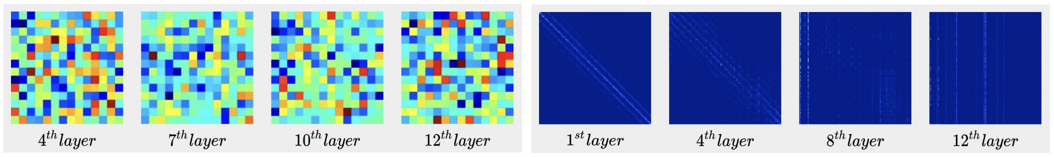


Figure 1: Left: the attention values of class tokens (normalized and reshaped in image shape) in different self-attention layers. Right: the attention maps in different self-attention layers. Zoom-in for better visibility.

To solve these problems, we propose Semantic Token ViT (STViT), for efficient global and local vision transformers, which also can be revised to serve as backbone for downstream tasks. The proposed approach is based on the following observations: (i) unlike local CNNs which learn spatial structure of images, vision transformer discretizes feature map as tokens for global feature exploration, relieving the requirements for maintaining the whole image structure and information; (ii) discrete tokens are more beneficial for optimization (Wang et al., 2022); (iii) in Figure 1, on the right shows the attention maps in different transformer layers, and there are only several vertical lines in the deep layers, which means that only a few tokens with global semantic information matter. Thus, we argue that it is not necessary to maintain massive structured tokens for ViTs, especially in the deep layers. Employing a few discrete tokens with high-level semantic information can potentially achieve both high performance and efficiency.

In STViT, the semantic tokens represent the cluster centers, and the number of them is far less than the original image tokens, significantly reducing the computational cost. Inspired by the fact that multi-head attention can conduct the cluster center recovery (Appendix A.8), we only employ the off-the-shelf self-attention to generate the semantic tokens. Specifically, the first few transformer layers are kept unchanged to obtain the image tokens with low-level features. The image tokens are then fed into our semantic token generation module (STGM) consisting of at least two transformer layers to generate semantic tokens. In each self-attention layer, the semantic tokens are input as queries, and the image tokens are fed as keys and values. The semantic tokens dynamically aggregate image tokens through the attention layers to recover cluster centers. In the first attention layer, the semantic tokens are initialized by an intra and inter-window spatial pooling which takes into account incorporating semantic information in each window and maximizing distance between adjacent windows. Thanks to this spatial initialization, the semantic tokens mainly incorporate local semantic information and achieve discrete and uniform distribution in space. In the following attention layer, besides further clustering, the semantic tokens are equipped with global cluster centers, and the network can adaptively select partial semantic tokens to focus on global semantic information. After the STGM, the original image tokens are discarded, and only semantic tokens are kept for the subsequent transformer layers. Because the generation of semantic tokens is flexible and space-aware, our method can be plugged into both global and local vision transformers. The semantic tokens can be produced in each window for the local vision transformer.

Another property of STViT is its capability to serve as a backbone for downstream tasks, such as object detection and instance segmentation. Discussions have been missing in all previous methods (Xu et al., 2022; Fayyaz et al., 2021; Liang et al., 2022; Ryoo et al., 2021; Zong et al., 2021) about how to use them in downstream task under the massive loss of spatial information during the token sparsification process, which actually seriously impedes the application of their method. Instead, we design a novel STViT-R network based on STViT where a recovery module and dumbbell unit are adopted to periodically restore the full resolution feature map while the intermediate transformer layers continue to use semantic tokens to save computation cost, making our method work in downstream task.

The effectiveness of the proposed method is validated via a comprehensive empirical study on image and video ViT models. Only 16 semantic tokens on DeiT-(Tiny, Small, Base) achieve nearly 50% inference time reduction without any accuracy degradation; on Swin-(Tiny, Small, Base), we also improve the inference throughput by nearly 20% with slight accuracy increase. Moreover, the proposed STViT-R achieves promising results on object detection and instance segmentation. To the best of our knowledge, this is the first work to apply the token sparsification algorithm in local vision transformers, and the first work to use the ViTs as backbones in downstream tasks after large-scale token pruning. Our findings in ViTs uncover that maintaining the full-size feature map is unnecessary, and a few tokens with high-level semantic representations can achieve both high performance and efficiency. Thanks to its simplicity and general-purpose ability, our method can also serve as a new efficient ViT baseline architecture and a starting point for further research from the token sparsification perspective.

2 RELATED WORK

Vanilla transformers have high computational and memory costs because the multi-head self-attention has quadratic computational complexity to the number of image tokens. Recently, various efficient ViTs have been proposed to alleviate this issue. The existing methods mainly focus on reducing the complexity of self-attention or reducing the number of tokens. Swin Transformer (Liu et al., 2021) adopts local self-attention, *i.e.*, attending neighboring tokens within a constant window size, achieving a linear computational complexity in the self-attention with high performance. Many subsequent works (Dong et al., 2021; Yu et al., 2022; Wang et al., 2021; Yang et al., 2021; Huang et al., 2021; Zhou et al., 2021; Chu et al., 2021a) follow the local self-attention design to develop variants.

Token sparsification methods can be mainly categorized into hard pruning (Rao et al., 2021; Pan et al., 2021; Chen et al., 2021; Song et al., 2021; Yin et al., 2022; Meng et al., 2021; Fayyaz et al., 2021; Xu et al., 2022; Kong et al., 2021; Liang et al., 2022; Tang et al., 2021) and soft pruning (Ryoo et al., 2021; Zong et al., 2021). Hard pruning methods filter out some unimportant tokens according to a predefined scoring mechanism. DynamicViT (Rao et al., 2021), SPViT (Kong et al., 2021), and AdaViT (Meng et al., 2021) introduce additional prediction networks to score the tokens. Evo-ViT(Xu et al., 2022), ATS (Fayyaz et al., 2021), and EViT (Liang et al., 2022) utilize the values of class tokens to evaluate the importance of tokens. However, it is difficult to achieve precise scoring as shown in the left of Figure 1. Therefore, they usually suffer from a significant accuracy drop. For instance, EViT (Liang et al., 2022) has an accuracy drop of 1.3% when saving 50% FLOPs on DeiT-S. Soft pruning methods generate new tokens from image tokens by importing additional attention networks. TokenLearner (Ryoo et al., 2021) also argue for a few tokens to replace image tokens. However, its price is a 1.8% accuracy drop when reducing 44% FLOPs, which is far inferior to concurrent works. Besides performance degradation, previous methods also have the following disadvantages. First, whether or how to extend the methods to local vision transformers remains unexplored. Second, it has not been discussed about how to serve the downstream tasks like object detection and instance segmentation after the tokens are pruned.

In our method, we apply the off-the-shelf transformer layers to reduce token number. (Ma et al., 2021; Jaegle et al., 2021; Bai et al., 2021; Zhang et al., 2019; Li et al., 2019; Chen et al., 2019) adopt similar approaches to achieve efficient non-local relationships. Our method is different from them as below: (i) our method extracts local semantic information instead of non-local relationships; (ii) the semantic tokens are a few cluster centers, which can replace the massive image tokens to achieve image classification; (iii) our method specializes in pruning tokens.

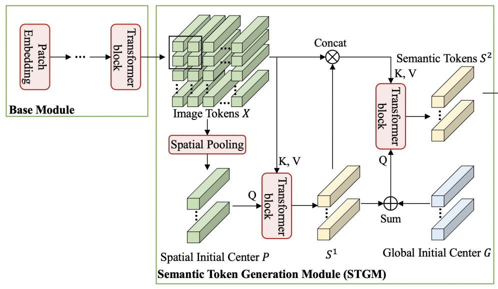
3 METHOD

The proposed STViT is presented in this section, which aims to construct an efficient and high-performance ViT. STViT is first introduced in Section 3.1, followed by how to apply STViT in the local vision transformer in Section 3.2. Based on STViT, STViT-R is developed to restore the spatial resolution for downstream tasks in Section 3.3.

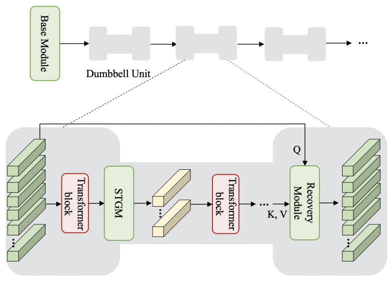
3.1 STVIT

Overall architecture. An overview of STViT architecture is presented in Figure 2a. The patch embedding layer and shallow transformer layers are kept unchanged as a base module in our method. The base module copes with all the image tokens $X \in \mathbb{R}^{N_i \times C}$ to extract low-level features, where N_i is the number of image tokens and C is the number of channels. The image tokens are fed into the semantic token generation module (STGM) to generate N_s semantic tokens $S \in \mathbb{R}^{N_s \times C}$. After the STGM, the image tokens X can be discarded, and only semantic tokens S with high-level semantic information are used in all the subsequent transformers. Due to $N_s \ll N_i$, our method can significantly reduce the computational cost.

Semantic token generation module (STGM). The whole image is represented by a few tokens with high-level semantic information through clustering. Inspired by the fact that self-attention can



(a) STViT



(b) STViT-R

Figure 2: The architectures of our STViT and STViT-R.

conduct cluster center recovery(Appendix A.8), we adopt the off-the-shelf self-attention layers to produce the semantic tokens. The STGM consists of at least two transformer layers.

The initial cluster centers $P \in \mathbb{R}^{N_s \times C}$ are generated by an spatial pooling which pools the image tokens into fixed $w_s \times w_s$ tokens, with $N_s = w_s \times w_s$. w_s is generally set as 4. The spatial pooling can be achieved by a non-parameterized adaptive spatial pooling or a super lightweight network with higher performance, intra and inter-window spatial pooling. The intentions of spatial pooling initialization are three folds. First, the initial cluster centers can distribute uniformly in space, making the generated semantic tokens more discrete and preventing the semantic tokens from collapsing to one point in the following layers. Second, the semantic tokens can be forced to represent more local and distinguished features. Finally, the representation of semantic tokens is associated with the specific spatial locations, which is the basis to allow our method to be applied in local self-attention and downstream tasks. The initial cluster centers P then dynamically integrate the image tokens X according to semantic information by attention mechanism. In the first transformer layer, the processing of the generation of semantic tokens can be written as

$$\hat{S^1} = MHA(P, X, X) + P, \quad S^1 = FFN(\hat{S^1}) + \hat{S^1}, \tag{1}$$

where MHA and FFN are short for multi-head attention layer and feed-forward network, respectively, and the triplet input of MHA are queries, keys, and values in turn. All the norm layers are omitted in all the equations for brevity. The initial cluster centers are produced in a window by an adaptive spatial pooling layer or an intra and inter-window spatial pooling, while the semantic tokens are generated from a global receptive field by a dynamic attention layer to ensure that they can extract high-level semantic representation. In order to further strengthen the clustering effect, we use the second transformer layer to repeat the clustering operation and guide the semantic tokens to extract global information. In this transformer layer, the semantic tokens are updated as:

$$\hat{S^2} = MHA(S^1 + G, Concat(S^1, X), Concat(S^1, X)) + S^1, \quad S^2 = FFN(\hat{S^2}) + \hat{S^2}, \tag{2}$$

where $G \in \mathbb{R}^{N_s \times C}$ is the global cluster centers initialized by Gaussian noise and $Concat(\cdot)$ is a concatenation operation. The global cluster centers G are responsible for global semantic information extraction like the class token. Although S^1 and G are summed together as the queries, they can be decoupled in the attention computation as:

$$\begin{gathered} A_s = (S^1 \cdot W_q) \cdot ((S^1 + X) \cdot W_k), \quad A_g = (G \cdot W_q) \cdot ((S^1 + X) \cdot W_k), \\ A = Softmax(A_s + A_g), \end{gathered} \tag{3}$$

where W_q and W_k are the linear projection weights of queries and keys. We can see that S^1 and G integrate the keys to generate A_s and A_g independently and just share a Softmax operation to produce the final attention map A. Therefore, besides spatial semantic information, the semantic tokens also incorporate global semantic information with a negligible additional overhead. Though very similar, the global cluster centers are actually different from the learned positional encoding. We do not add the global cluster centers to keys. Moreover, it will be shown in the experiments that inserting the actual learned positional encoding will cause an accuracy drop. The number of transformer layers in STGM is flexible. More transformer layers can be associated for clustering. Two transformer layers are employed by default, *i.e.*, S^2 is the output of the STGM. Note that the image tokens are not updated in the STGM.

Intra and inter-window spatial pooling. Give $H \times W$ feature map X, we generate $N_s(w_s \times w_s)$ initial cluster centers. We uniformly split the X into $w_s \times w_s$ windows $[X_w^i]_{i=1}^{N_s}$ with size $\frac{H}{w_s} \times \frac{W}{w_s}$ and each window generates one initial cluster center. To represent abundant semantic information, we take into account intra and inter-window relations, i.e., integrating important tokens in the window and maximizing the distance among initial cluster centers in different windows. Specifically, we formulate an intra-window function f_{intra} to produce a mask, $M_i = f_{intra}(X_w^i)$, which projects the input window from $\mathbb{R}^{\frac{H}{w_s} \times \frac{W}{w_s} \times C} \rightarrow \mathbb{R}^{\frac{H}{w_s} \times \frac{W}{w_s}}$. The idea is to let the intra-window function f_{intra} adaptively select a combination of informative tokens in X_w^i, which is implemented by

$$M_i = Conv(GeLU(LayerNorm(DepthwiseConv(X_w^i)))). \tag{4}$$

Then, we compute each integrated token $\hat{P}_i = Softmax(M_i) \cdot X_w^i$ from each window, and arrange $[\hat{P}_i]_{i=1}^{N_s}$ by spatial structure to form the 2D tensor $\hat{P} \in \mathbb{R}^{w_s \times w_s \times C}$. We adopt an inter-window function f_{inter} to compute the inter-window relations and generate the offset, $O = f_{inter}(\hat{P})$,to revise the mask M. The implementation of f_{inter} is similar to Eq. 4, except the mapping input from $\mathbb{R}^{w_s \times w_s \times C} \rightarrow \mathbb{R}^{w_s \times w_s \times \frac{HW}{w_s^2}}$, where $\frac{HW}{w_s^2}$ is the number of tokens in each window. For each window, the corresponding $O_i \in \mathbb{R}^{\frac{HW}{w_s^2}}$ is sliced from O and reshaped to $\mathbb{R}^{\frac{H}{w_s} \times \frac{W}{w_s}}$. The O_i is used to revise M_i. The final initial cluster center P_i is computed by

$$P_i = Softmax(M_i + O_i) \cdot X_w. \tag{5}$$

Our f_{intra} and f_{intra} are super lightweight and the introducing parameters can be negligible. For example, on DeiT-T, the parameters only increase by 0.05%.

3.2 STViT in local vision transformers.

Local self-attention has been widely used in current ViT models to balance efficiency and accuracy. As the generation of semantic tokens in STViT is flexible in space, it can be naturally applied in local self-attention. Suppose each local self-attention layer contains N_w windows with size $w \times w$, we initialize $w_s \times w_s$ cluster centers in each window by our intra and inter-window spatial pooling. The total number of semantic tokens N_s would be $w_s \times w_s \times N_w$. w and w_s are set as 7 and 3 by default separately. As a result, our method compresses more than 80% image tokens in local self-attention. In the STGM, although initial cluster centers are from $w \times w$ windows, we use larger windows with size $w_k \times w_k$ to obtain keys and values in Eq. 1 and Eq. 2 to mitigate the effect of limited window size. Other operations in STGM are kept the same as Section 3.1.

In the local ViT models, each local transformer layer is normally followed by a cross-window connection layer, such as a shift window transformer layer following a local transformer layer on Swin Transformer (Liu et al., 2021). In our method, the attention is computed within $w_s \times w_s$ window in the local self-attention layer, and the cross-window connection can be achieved by computing self-attention in a larger-size (*e.g.*, $4 \times w_s$) sliding window because of the rare number of tokens in each window. For the low-resolution input, our cross-window connection layer is equal to a global self-attention layer.

3.3 STViT for downstream tasks

Our method significantly reduces the computation cost by using a small number of semantic tokens, while its side effect is losing nearly all the detailed position information. High-level vision tasks, such as object detection and instance segmentation, are difficult to be executed on this extremely incomplete feature map. This issue also exists in previous works, which hinders the application of token sparsification methods. To solve this issue, we design a STViT-R network based on STViT to restore the original spatial resolution from the semantic tokens.

Our STViT-R shown in Figure 2b has two modifications compared with STViT. First, we adopt a recovery module to restore the spatial resolution from semantic tokens; second, we regroup the transformer layers and construct dumbbell units composed of our STViT-R.

Recovery module. In the recovery module, only the self-attention layer is employed to restore the spatial resolution without any additional networks. The image tokens X and semantic tokens S are partitioned as N_w^r windows of size $w^r \times w^r$ and $w_s^r \times w_s^r$, respectively. The image tokens in each window aggregate the semantic tokens in the corresponding window, which is represented as:

$$\hat{X} = MHA(X, S, S) + X, \quad X = FFN(\hat{X}) + \hat{X}. \tag{6}$$

This is a reverse operation of the generation of semantic tokens, using high-level semantic information to boost the image tokens.

Dumbbell unit. The transformer layers are regrouped into multiple dumbbell units in our STViT-R. Each dumbbell unit consists of four parts. The transformers in the first part are responsible for coping with image tokens; the second part is the semantic token generation module; the transformer layers in the third part deal with semantic tokens; the last part is the recovery module. Take the application on Swin-S (STViT-R-Swin-S) as an example. One, two, two and one transformer layers are allocated for these four parts, respectively. In total, each dumbbell unit is composed of 6 transformer layers. We concatenate three dumbbell units in Stage 3 of Swin-S. In each dumbbell unit, the intermediate transformer layers process semantic tokens with high-level semantic information to save computational cost, and the complete spatial resolution is recovered at the end. By repeating multiple dumbbell units, the detailed spatial information will be preserved by the network, which can not only enhance the classification but also serve downstream tasks.

4 EXPERIMENTS

STViT will first be applied in two representative ViT models, DeiT (Touvron et al., 2021a) and Swin (Liu et al., 2021) for image classification and video recognition. To validate that our method is effective in downstream tasks, STViT-R is then performed on object detection and instance segmentation tasks.

4.1 IMAGE CLASSIFICATION

Settings. For image classification, all the models are trained on the ImageNet (Deng et al., 2009). By default, the semantic token generation module (STGM) employs the 5^{th} and 6^{th} transformer layers of DeiT (with 12 layers in total), employs the 3^{rd} and 4^{th} transformer layers of Stage 3 of Swin-T (with 12 layers in total), and employs the 11^{th} and 12^{th} transformer layers of Stage 3 of Swin-S and Swin-B (with 24 layers in total). The image resolution in training and inference is 224×224. The batch size is 1,024. All the models are trained from scratch for 300 epochs, and the augmentation and regularization strategies follow the original papers of DeiT and Swin. No knowledge distillation algorithms are used in our experiments. The classification is performed by applying a global average pooling layer on the output tokens of the last transformer layer, followed by a linear classifier. In evaluation, the top-1 accuracy using a single crop is reported. The FLOPs computations of this paper are measured by fvcore[2]. Throughput is measured with the batch size of 128 on a V100 GPU.

On DeiT (Touvron et al., 2021a), we replace the original patch embedding layer containing a 16×16 convolutional layer with four lightweight 3×3 convolutional layers. We set the group of the last convolutional layer as 2 to make the number of parameters not exceeding the original patch embedding layer for fair comparison. The reason for reconstructing the patch embedding layer is to obtain better low-level features to serve the generation of semantic tokens. Otherwise, we need to apply more transformer layers in the base module, causing a huge overhead on FLOPs.

On Swin (Liu et al., 2021), the semantic tokens are generated in Stage 3, and they are not downsampled in Stage 4. The patch merging layer between Stage 3 and Stage 4 is replaced with a simple linear layer to double the number of channels. w_k is set as 10 and 14 for two transformer layers of STGM.

Results. One of the advantages of our method is that it can be applied to both global and local vision transformers to reduce computational complexity. Our main results on DeiT and Swin are summarized in Table 1 and Table 2, respectively. The results of LV-ViT (Jiang et al., 2021) are illustrated in Appendix A.3 due to limited space. We report the top-1 accuracy, FLOPs, and the throughput under different numbers of semantic tokens. On DeiT, the models with 16 semantic tokens achieve the same accuracy as the DeiT models with 196 tokens and save nearly 60% FLOPs on DeiT-S and DeiT-B. With more semantic tokens, the accuracy can consistently outperform the base models. For instance, STViT-DeiT-B with 36 semantic tokens surpasses DeiT-B by 0.4% accuracy with 52% FLOPs reduction.

The local vision transformer like Swin is already an efficient architecture compared to the global vision transformer, so the reduction of FLOPs on Swin models are smaller than on DeiT models.

[2]`https://github.com/facebookresearch/fvcore`

Table 1: Applying STViT to DeiT-T, DeiT-S, and DeiT-B. The top-1 accuracy, complexity in FLOPs, and throughput are reported for different numbers of semantic tokens. *Base* indicates the counterpart DeiT model with our modification of patch embedding layer.

Model	Metrics	Base	No. of semantic tokens			
			16	36	64	100
STViT-DeiT-T	Top-1 Acc(%)	75.0	75.4(+0.4%)	75.6(+0.6)	76.0(+1.1)	76.1(+1.2)
	FLOPs(G)	1.32	0.59(-55%)	0.67(-50%)	0.78(-41%)	0.92(-30%)
	Throughput(img/s)	2752	5511(+101%)	4769(+74%)	4214(+53%)	3551(+29%)
STViT-DeiT-S	Top-1 Acc(%)	80.5	80.6(+0.1)	80.9(+0.4)	81.3(+0.8)	81.4(+0.9)
	FLOPs(G)	4.66	1.97(-58%)	2.256(-52%)	2.67(-43%)	3.21(-31%)
	Throughput(img/s)	1408	2891(+105%)	2542(+80%)	2229(+58%)	1837(+30%)
STViT-DeiT-B	Top-1 Acc(%)	81.8	81.8(-0.0)	82.2(+0.4)	82.6(+0.8)	82.7(+0.9)
	FLOPs(G)	17.61	7.31(-59%)	8.44(-52%)	10.04(-43%)	12.13(-31%)
	Throughput(img/s)	626	1308(+110%)	1150(+85%)	1087(+61%)	826(+33%)

Table 2: Applying STViT to Swin-T, Swin-S, and Swin-B. The top-1 accuracy, complexity in FLOPs, and throughput are reported for different numbers of semantic tokens in each window. *Base* indicates the corresponding original Swin model. *Move STGM* indicates changing the default position of STGM.

Model	Metrics	Base	Move STGM	No. of semantic tokens		
				4	9	16
STViT-Swin-T	Top-1 Acc(%)	81.3	81.0(-0.3%)	80.8(-0.5)	81.5(+0.2)	81.8(+0.5%)
	FLOPs(G)	4.5	3.14(-30%)	2.99(-34%)	3.43(-24%)	4.06(-10%)
	Throughput(img/s)	878	1124(+29%)	1128(+29%)	1061(+22%)	1008(+15%)
STViT-Swin-S	Top-1 Acc(%)	83.0	82.8(-0.2%)	82.4(-0.6%)	83.0(-0.0)	83.1(+0.1%)
	FLOPs(G)	8.7	5.95(-32%)	5.95(-32%)	6.53(-25%)	7.36(-15%)
	Throughput(img/s)	551	739(+35%)	732(+34%)	691(+26%)	657(+20%)
STViT-Swin-B	Top-1 Acc(%)	83.5	83.2(-0.3%)	83.0(-0.5)	83.4(-0.1)	83.7(+0.2%)
	FLOPs(G)	15.4	10.48(-32%)	10.48(-32%)	11.51(-25%)	12.97(-16%)
	Throughput(img/s)	415	558(+35%)	551(+33%)	521(+26%)	489(+19%)

When 9 semantic tokens are used in each window, STViT-Swin models can reduce 25% FLOPs with negligible accuracy loss on all the model sizes. If the number of tokens is reduced to 4 in each window (16 in total), a significant accuracy drop will occur, which indicates that local vision transformers need more semantic tokens than global vision transformers. We can move the STGM towards shallow layers to attain a better complexity/accuracy trade-off. In Table 2, STGM is moved by one transformer layer on STViT-Swin-T and by two layers on STViT-Swin-S and STViT-Swin-B to save over 30% FLOPs with only about 0.3% accuracy drops (column of "Move STGM").

These results demonstrate that our method achieves both effectiveness and efficiency by employing a few semantic tokens to replace original image tokens. Our method reveals that constructing the tokens with high-level semantic representation is more important than maintaining structured tokens in ViTs. As reflected by the throughput, our method does not have overhead of memory or deployment. Compared to the total parameters, the additional parameters introduced by intra and inter-window spatial pooling is negligible(less than 0.05%), so we do not show them.

Comparisons with existing token sparsification methods. In Table 3, we compare STViT with the state-of-the-art token sparsification methods on DeiT. Due to different base models used by different methods, we adopt accuracy difference between each model and its base model Δ to evaluate them for fair. Results indicate that our method achieves the lowest accuracy drop Δ with the highest FLOPs reduction, outperforming all the state-of-the-art methods in both accuracy and efficiency significantly.

4.2 Video Recognition

Setting. For video recognition, we apply our STViT to Video Swin (Liu et al., 2022). All the models are pretrained on ImageNet-1K and trained on Kinetics-400 (Carreira & Zisserman, 2017). We generate semantic tokens from each frame as illustrated in Section 3.2. The initialization from pretrained model and other implementation details are as same as Video Swin (Liu et al., 2022).

Results. The results are presented in Table 5. On Swin-T and Swin-S, STViT-Swin can save about 27% FLOPs with 0.3% accuracy drop, which shows that our method works on video recognition.

Table 3: Comparisons with the state-of-the-art token sparsification methods on DeiT-S and DeiT-B. $\triangle$ shows the accuracy difference between each model and its base model.

(a) DeiT-S

Model	Top-1 Acc	FLOPs(G)	$\triangle$
DynamicViT (Rao et al., 2021)	79.3	2.9(-37%)	-0.5
IA-RED^2 (Pan et al., 2021)	79.1	3.2(-30%)	-0.7
TokenLearner (Ryoo et al., 2021)	76.1	1.9(-44%)	-1.8
DGE (Song et al., 2021)	79.7	3.1 (-49%)	-0.6
Evo-ViT (Xu et al., 2022)	79.4	3.0(-35%)	-0.4
EViT (Liang et al., 2022)	78.5	2.3(-50%)	-1.3
STViT(Ours)	80.6	1.97(**-58%**)	**+0.1**

(b) DeiT-B

Model	Top-1 Acc	FLOPs(G)	$\triangle$
IA-RED^2 (Pan et al., 2021)	80.3	11.8(-33%)	-1.5
DynamicViT (Rao et al., 2021)	81.3	11.2(-36%)	-0.5
PS-ViT (Tang et al., 2021)	81.5	9.8(-44%)	-0.3
TokenLearner (Ryoo et al., 2021)	83.7	28.7(-48%)	-1.1
Evo-ViT (Xu et al., 2022)	81.3	10.2(-33%)	-0.5
EViT (Liang et al., 2022)	80.0	8.7(-51%)	-1.8
STViT(Ours)	81.8	7.31(**-59%**)	**-0.0**

Table 4: Results on COCO object detection and instance segmentation under Cascade Mask R-CNN with 3× schedule. The FLOPs are measured for backbones.

	AP^b	AP$^b_{50}$	AP$^b_{75}$	AP^b_s	AP^m	AP$^m_{50}$	AP$^m_{75}$	AP^m_s	FLOPs(G)
Swin-S	51.8	70.4	56.3	35.2	44.7	67.9	48.5	28.8	194
STViT-R-Swin-S	51.8	70.6	56.1	36.7	44.7	67.8	48.6	29.0	134(-31%)
Swin-B	51.9	70.9	56.5	35.4	45.0	68.4	48.7	28.9	343
STViT-R-Swin-B	52.2	70.8	56.8	36.5	45.2	68.3	49.1	29.5	233(-32%)

4.3 Applications in object detection and instance segmentation

Settings. Experiments of object detection and instance segmentation are conducted on COCO 2017 (Lin et al., 2014). We evaluate STViT-R with Swin in Cascade Mask R-CNN (Cai & Vasconcelos, 2018; He et al., 2017) detection frameworks. The w_s is set to 3. All the other settings follow Swin (Liu et al., 2021).

Comparison to Swin Transformer. The performance of STViT-R-Swin using the Cascade Mask R-CNN framework with 3× schedule is shown in Table 4. Our method achieves better performance with more than 30% FLOPs reduction for backbone. This validates that the recovery module and dumbbell unit can restore detailed spatial information, and the global context information integrated from the semantic tokens significantly benefits object detection. Ignoring spatial structure in the intermediate layers does not affect the object detection task, which is a meaningful fact to help design efficient object detection frameworks. Another interesting finding is that our method has a remarkable improvement for small object detection which is a challenging problem in the detection community. For instance, our STViT-R-Swin-S outperforms Swin-S by 1.5% on AP_s^b.

4.4 Ablation study

All the following ablation experiments of STViT and STViT-R are conducted on the DeiT-S and Swin-S, respectively.

Initialization analysis. The semantic tokens are the cluster centers recovered by attention layers. The initialization of cluster centers induces the representation of semantic tokens. Spatial and global initialization are adopted in Section 3.1 to guide the semantic tokens to integrate local and global semantic information separately. We compare different initialization components in Table 6. When performing single global initialization(3^{rd} row), we replace the spatial initial cluster centers with global initial cluster centers in the first transformer layer. The accuracy of using a single initialization method is far lower than using both, which shows the effectiveness of our initialization strategy.

Table 5: Applying STViT to Video Swin (Swin-T and Swin-S) on Kinetics-400. All the models are pre-trained on ImageNet-1K. The views are 4×3. The top-1 accuracy and complexity in FLOPs are reported. Speed is evaluated by FPS.

Model	Top-1 Acc(%)	FLOPs(G)	Speed
Swin-T	78.8	88	779
STViT-Swin-T	78.5(-0.3)	64.4(-27%)	975(+25%)
Swin-S	80.6	166	456
STViT-Swin-S	80.3(-0.3)	120.5(-27%)	572(+25%)

Table 6: Accuracy with different initialization of STViT. *Spatial*, *Global*, and *Learned* indicate spatial initialization, global initialization, and learned PE, respectively.

Spatial	Global	Learned	Top-1 Acc(%)
✓			80.1
	✓		79.5
✓	✓		80.6
✓		✓	80.5

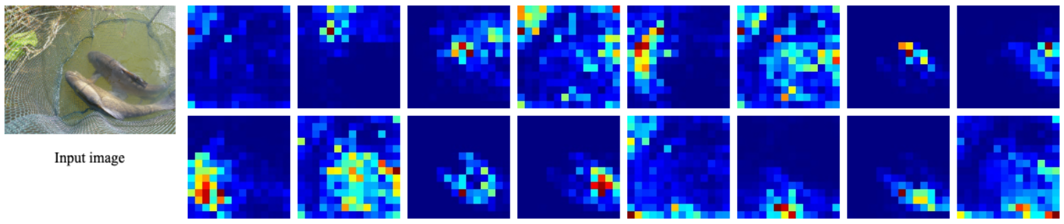


Figure 3: Visualization example of attention maps in the first attention layer of STGM.

Table 7: Performance evaluation on different numbers of transformer layers in STGM. Keeping the base module unchanged.

No. of transformers	2	3	4
Top-1 Acc(%)	80.6	80.3	80.4
FLOPs(G)	1.97	2.03	2.09

Table 8: Ablation study on STViT-R w/o dumbbell units (*w/o DU*) and reusing semantic tokens (*Reusing ST*).

	STViT-R	w/o DU	Reusing ST
AP^b	51.8	51.4	51.6
AP^m	44.7	44.4	44.5

Our global initial cluster centers look similar to learned positional encoding since both use random initialization. To verify their distinction, we experiment with a real learned positional encoding by additionally adding the global initial cluster centers to keys, which causes 0.1% accuracy drop (the last row of Table 6). Therefore, our global initialization is different from learned positional encoding.

We visualize the attention maps of the first attention layer in the STGM in Figure 3. Because the queries of this layer are spatial initial cluster centers, these attention maps visualize the local semantic information integration by attention. The attention layer groups the semantic information according to the position of initial cluster centers, which ensures to extract fine-grained semantic information and keep the difference among semantic tokens. The attention map in the second attention layer is visualized in Appendix A.6, which reveals that the network fixedly selects particle semantic tokens to represent global semantic information. We also show the semantic representation of image tokens in the same transformer layer on DeiT in Appendix A.6. Compared to original image tokens, our semantic tokens in Figure 3 show more high-level semantic information.

Number of transformer layers in STGM. Two transformer layers are adopted in the STGM by default. The effects of employing different numbers of transformer layers are shown in Table 7. The additional layers are from the ones behind STGM to keep the total number of layers unchanged. More transformer layers do not bring improvement.

The effectiveness of the dumbbell unit. To verify the effectiveness of our dumbbell unit, we experiment STViT-R without dumbbell units, *i.e.*, STViT equipped with only the recovery module. We employ 6^{th} and 7^{th} transformer layers to construct STGM and the last transformer layers to construct the recovery module in Stage 3. The FLOPs is as same as the full-model STViT-R for fair comparison. The results on the COCO dataset are reported in Table 8. Inferior results demonstrate the effectiveness of the dumbbell unit.

Reusing semantic tokens in dumbbell units. Semantic tokens are generated in each dumbbell unit. If they are produced only once in the first dumbbell unit and reused as initial cluster centers in the subsequent dumbbell units, the result is shown in Table 8 with a slight performance drop.

5 CONCLUSION

In this paper, we propose a simple and effective token sparsification method, semantic token vision transformer (STViT). Our method utilizes the clustering property of self-attention to generate a few semantic tokens with high-level information representation to replace the redundant image/video tokens, which can be applied in both global and local vision transformers. By simply configuring the recovery module, our method can be successfully applied to downstream tasks. Extensive experiments demonstrate that our method achieves better accuracy along with less inference time in most cases. The success in downstream tasks significantly boosts the development of token sparsification methods. We hope that this work can inspire more future research to pay much attention to high-level semantic representation in ViTs.

REFERENCES

Song Bai, Philip Torr, et al. Visual parser: Representing part-whole hierarchies with transformers. *arXiv preprint arXiv:2107.05790*, 2021.

Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6154–6162, 2018.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6299–6308, 2017.

Tianlong Chen, Yu Cheng, Zhe Gan, Lu Yuan, Lei Zhang, and Zhangyang Wang. Chasing sparsity in vision transformers: An end-to-end exploration. *Advances in Neural Information Processing Systems*, 34, 2021.

Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 433–442, 2019.

Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021a.

Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021b.

MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Ats: Adaptive token sampling for efficient vision transformers. *arXiv preprint arXiv:2111.15667*, 2021.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning. *arXiv preprint arXiv:2112.13890*, 2021.

Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9167–9176, 2019.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=BjyvwnXXVn_`.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

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*, pp. 10012–10022, 2021.

Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, and Han Hu. Video swin transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3202–3211, 2022.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453, 2021.

Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. Adavit: Adaptive vision transformers for efficient image recognition. *arXiv preprint arXiv:2111.15668*, 2021.

Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red˜2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34, 2021.

Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Lin Song, Songyang Zhang, Songtao Liu, Zeming Li, Xuming He, Hongbin Sun, Jian Sun, and Nanning Zheng. Dynamic grained encoder for vision transformers. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Yehui Tang, Kai Han, Yunhe Wang, Chang Xu, Jianyuan Guo, Chao Xu, and Dacheng Tao. Patch slimming for efficient vision transformers. *arXiv preprint arXiv:2106.02852*, 2021.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021a.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021b.

Pichao Wang, Xue Wang, Hao Luo, Jingkai Zhou, Zhipeng Zhou, Fan Wang, Hao Li, and Rong Jin. Scaled relu matters for training vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. *arXiv preprint arXiv:2108.00154*, 2021.

Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 22–31, 2021.

Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 418–434, 2018.

Yifan Xu, Zhijie Zhang, Mengdan Zhang, Kekai Sheng, Ke Li, Weiming Dong, Liqing Zhang, Changsheng Xu, and Xing Sun. Evo-vit: Slow-fast token evolution for dynamic vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers, 2021.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10809–10818, 2022.

Tan Yu, Gangming Zhao, Ping Li, and Yizhou Yu. Boat: Bilateral local attention vision transformer. *arXiv preprint arXiv:2201.13027*, 2022.

Songyang Zhang, Xuming He, and Shipeng Yan. Latentgnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pp. 7374–7383. PMLR, 2019.

Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019.

Jingkai Zhou, Pichao Wang, Fan Wang, Qiong Liu, Hao Li, and Rong Jin. Elsa: Enhanced local self-attention for vision transformer. *arXiv preprint arXiv:2112.12786*, 2021.

Zhuofan Zong, Kunchang Li, Guanglu Song, Yali Wang, Yu Qiao, Biao Leng, and Yu Liu. Self-slimmed vision transformer. *arXiv preprint arXiv:2111.12624*, 2021.

A Appendix

A.1 Detailed Architectures

Detailed architecture specification on STViT-DeiT and STViT-Swin. The detailed architectures of our STViT-DeiT and STViT-Swin are shown in Table 9 and Table 10, where an input image size 224×224 is assumed for all the networks and the default numbers of semantic tokens are 16 and 36 separately. "Tr(ST)" denotes the transformers processing semantic tokens.

The details of patch embedding layer on DeiT. On DeiT, we use a multiple convolutional layers with small kernel size to substitute original patch embedding layer. The architecture detail of our new patch embedding layer is shown in Table 11. The parameter numbers of our patch embedding layer and original patch embedding layer are $52488 + 432C$ and $768C$, respectively. Because the value of C is 192 at least, the parameter number of ours is smaller than the original. Thus, our patch embedding layer is a lightweight module and does not introduce additional parameter overhead. Our patch embedding layer has stronger representation ability, which allows that less transformer layers are applied in the base module.

Table 9: Detailed architecture of STViT-DeiT.

	Output size	STViT-DeiT-T	STViT-DeiT-S	STViT-DeiT-B
Base	14×14	Patch Embedding $\begin{bmatrix} dim\ 192, head\ 3 \end{bmatrix} \times 4$	Patch Embedding $\begin{bmatrix} dim\ 384, head\ 6 \end{bmatrix} \times 4$	Patch Embedding $\begin{bmatrix} dim\ 768, head\ 12 \end{bmatrix} \times 4$
STGM	4×4	$\begin{bmatrix} dim\ 192, head\ 3 \end{bmatrix} \times 2$	$\begin{bmatrix} dim\ 384, head\ 6 \end{bmatrix} \times 2$	$\begin{bmatrix} dim\ 768, head\ 12 \end{bmatrix} \times 2$
Tr(ST)	4×4	$\begin{bmatrix} dim\ 192, head\ 3 \end{bmatrix} \times 6$	$\begin{bmatrix} dim\ 384, head\ 6 \end{bmatrix} \times 6$	$\begin{bmatrix} dim\ 768, head\ 12 \end{bmatrix} \times 6$

Table 10: Detailed architecture of STViT-Swin.

		Output size	STViT-Swin-T	STViT-Swin-S	STViT-Swin-B
Base	Stage 1	56×56	Patch Embedding $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 96, head\ 3 \end{bmatrix} \times 2$	Patch Embedding $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 96, head\ 3 \end{bmatrix} \times 2$	Patch Embedding $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 192, head\ 6 \end{bmatrix} \times 2$
	Stage 2	28×28	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 192, head\ 6 \end{bmatrix} \times 2$	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 192, head\ 6 \end{bmatrix} \times 2$	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 256, head\ 8 \end{bmatrix} \times 2$
	Stage 3	14×14	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 384, head\ 12 \end{bmatrix} \times 2$	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 384, head\ 12 \end{bmatrix} \times 10$	Patch Merging $\begin{bmatrix} win.\ sz.\ 7\times 7, \\ dim\ 512, head\ 16 \end{bmatrix} \times 10$
STGM	Stage 3	6×6	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 384, head\ 12 \end{bmatrix} \times 2$	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 384, head\ 12 \end{bmatrix} \times 2$	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 512, head\ 16 \end{bmatrix} \times 2$
Tr(ST)	Stage 3	6×6	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 384, head\ 12 \end{bmatrix} \times 2$	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 384, head\ 12 \end{bmatrix} \times 6$	$\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 512, head\ 16 \end{bmatrix} \times 6$
	Stage 4	6×6	Linear Layer $\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 768, head\ 24 \end{bmatrix} \times 2$	Linear Layer $\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 768, head\ 24 \end{bmatrix} \times 2$	Linear Layer $\begin{bmatrix} win.\ sz.\ 3\times 3, \\ dim\ 1025, head\ 32 \end{bmatrix} \times 2$

Table 11: The architecture detail of our patch embedding layer on DeiT. C indicates the channel number of the subsequent transformer layers.

	Kernel size	Input channel	Output channel	Stride	Group
Conv1	3	3	24	2	1
Conv2	3	24	48	2	1
Conv3	3	48	96	2	1
Conv4	3	96	C	2	2

A.2 COMPUTATIONAL COMPLEXITY ANALYSIS

Global vision transformer (DeiT). We define that the number of image tokens is N, the number of semantic tokens is M, and their dimension is C. The patch embedding layer is neglected. The computational complexity of a global transformer processing image tokens (IT) is:

$$\begin{aligned} \Omega(MHA(IT)) &= 4NC^2 + 2N^2C, \\ \Omega(FFN(IT)) &= 8NC^2. \end{aligned} \tag{7}$$

The computational complexity of a global transformer processing semantic tokens (ST) is:

$$\begin{aligned} \Omega(MHA(ST)) &= 4MC^2 + 2M^2C, \\ \Omega(FFN(ST)) &= 8MC^2. \end{aligned} \tag{8}$$

The relationships between computational complexity and token number in attention and FFN are quadratic and linear, respectively. Due to the $N \ll M$, our method significantly reduces the cost of transformers, especially the attention. The computational complexity of STGM is:

$$\Omega(STGM) = 2MC^2 + 2NC^2 + 2MNC. \tag{9}$$

In global vision transformers, our STGM is also an efficient module. The computational complexity of whole DeiT and our STViT-DeiT are:

$$\begin{aligned}\Omega(DeiT) &= 144NC^2 + 24N^2C,\\ \Omega(STViT) &= 52NC^2 + 12M^2C + 76MC^2 + 8N^2C + 4MNC.\end{aligned} \tag{10}$$

Local vision transformer (Swin). We define that the number of image tokens (IT) is N, the number of image tokens in each window is W, the number of semantic tokens (ST) in each window is M, and their dimension is C. We only compute the computational complexity in each transformer. The computational complexity of a local transformer processing image tokens (IT) is:

$$\begin{aligned}\Omega(MHA(IT)) &= 4NC^2 + 2W^2NC,\\ \Omega(FFN(IT)) &= 8NC^2.\end{aligned} \tag{11}$$

The computational complexity of a local transformer processing image tokens (ST) is:

$$\begin{aligned}\Omega(MHA(ST)) &= 4(N/W)MC^2 + 2M^2(N/W)C,\\ \Omega(FFN(ST)) &= 8MC^2.\end{aligned} \tag{12}$$

Swin makes the computational complexity linear to token number, while our method further reduces the computational complexity. The computational complexity of STGM is:

$$\Omega(STGM) = 2(N/W)MC^2 + 2NC^2 + 2(N/W)M^2C. \tag{13}$$

A.3 The results on LV-ViT

Setting. In LV-ViT (Jiang et al., 2021), by default, the STGM employs the 6^{th} and 7^{th} transformer layers of LV-ViT-S (with 16 layers in total). We downsample the token labels to match the size of our semantic tokens.

Results. The main results are shown in Table 12. Token labeling in LV-ViT is not friendly for our method. Token labeling emphasizes the important of all the output tokens and advocates that each output token should be associated with an individual location-speific label (Jiang et al., 2021), while our semantic tokens generated by clustering emphasize high-level semantic information. We still achieve good performance. In Table 13, we compare our STViT with the state-of-the-art token sparsification method EViT (Liang et al., 2022) on LV-ViT-S. Results indicate that our method outperform it.

A.4 STViT-R for image classification

STViT-R equipped with recovery modules is designed for downstream tasks, while it also can perform image classification. The corresponding results are reported in Table 14. On both Swin-S and Swin-B, STViT-R can save 33% FLOPs with 0.3% accuracy drop. The hyper-parameters we used in STGM are as same as STViT-Swin.

A.5 Applications in semantic segmentation

Settings. ADE20K (Zhou et al., 2019) is a widely-used semantic segmentation dataset, including a broad range of 150 semantic classes. It has 25K images in total, with 20K for training, 2K for validation, and 3K for testing. UperNet (Xiao et al., 2018) in mmseg (Contributors, 2020) is utilized as our base framework. The w_s is set to 3. Models are trained for 240K iterations. All the other settings follow the Swin Transformer (Liu et al., 2021).

Comparison to Swin Transformers. Table 15 presents the results of STViT-R-Swin on semantic segmentation. With similar FLOPs reduction, the drop on mIoU is larger compared with those in object detection tasks, which shows that our method still has a gap on dense prediction compared to the full-token network, and this is the limitation of this work.

We analyze the relatively poor performance from two views. First, the STGM strictly prunes more than 80% tokens by attention, which remains the high-level semantic information but loses nearly all the detailed information. Semantic segmentation is a dense pixel-level classification task, and the semantic tokens are difficult to enhance the pixel-level representation. Second, our spatial pooling layer with large kernel size in STGM and self-attention layers can be regarded as low-frequency filters. STGM filters most high-frequency information, which is necessary for semantic segmentation.

A.6 Additional visualization

We visualize the attention map of the second attention layer in STGM in Figure 4a. The shape of attention map is $N_s \times (N_s + N_i)$, where $N_s = 16$, and $N_i = 196$. The results of the attention computation between semantic tokens S^1 (queries) and semantic tokens S^1 (keys) are shown in the most left 16 columns, and the rest columns show the computation between semantic tokens S^1 and image tokens X. The figure shows that the second semantic token highlights the region of semantic tokens, while other semantic tokens highlight the image tokens. Figure 4c visualizes the attention maps in the self-attention layers after STGM. The second semantic token is incorporated by the majority of semantic tokens. These phenomenons illustrate that the second semantic token focuses on more global semantic information, which further verifies our global cluster center initialization can guide the semantic tokens to extract global semantic information. The phenomenons in Figure 4a and Figure 4c nearly emerge in all the images.

Neglecting the most left 16 columns of Figure 4a, we reshape it into 16 14×14 attention maps like Figure 3 and show them in Figure 4b. Thanks to the clustering of second attention and global initialization G, we can see that the semantic information is more accurate and meaningful.

We visualize the attention maps of semantic tokens with single global initialization in Figure 4d. Without spatial initialization, the response regions are more global and similar. In contrast, our semantics of each semantic token are associated with the specific spatial location, which is the basis to allow our method to be applied in local self-attention and downstream tasks. Additionally, our attention maps contain more recognized and diverse semantic information, reflecting the effectiveness of our spatial initialization.

To compare the semantic representation between our semantic tokens and original image tokens on DeiT, we uniformly sample 49 image tokens (7×7) from 196 image tokens (14×14) in the 5^{th} transformer layer of DeiT-S and show their attention response in Figure 5. In the same depth, our semantic tokens in Figure 3 capture high-level semantic information with completed foreground and background, while the semantic information of image tokens in Figure 5 is redundant and unclear. High-level semantic information captured by STGM helps our models achieve similar performance with a few tokens.

A.7 Additional ablation study

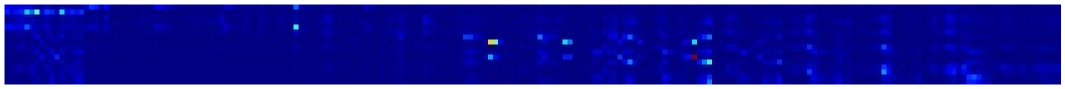
All the following experiments of STViT and STViT-R are conducted on DeiT-S and Swin-S unless otherwise specified, respectively.

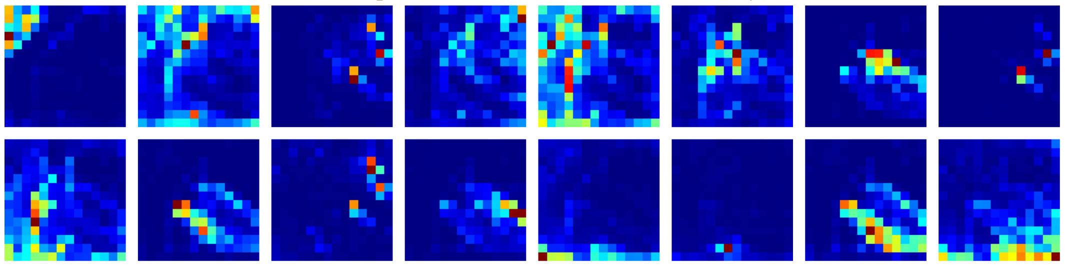
Table 12: Results of STViT on LV-ViT-S.

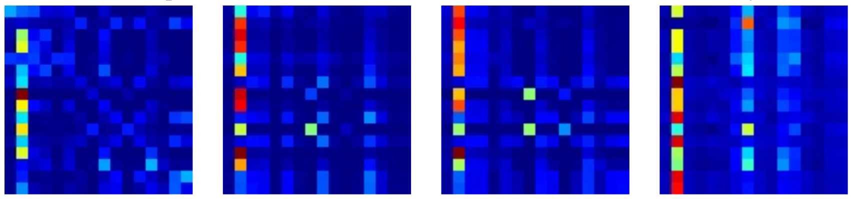
Model	Metrics	Base	No. of semantic tokens 36	49	100
STViT-LV-ViT-S	Top-1 Acc(%)	83.3	82.7(-0.6)	82.8(-0.5)	83.1(-0.2%)
	FLOPs(G)	6.6	3.69(-44%)	3.91(-41%)	4.62(-30%)
	Throughput(img/s)	1159	2073(+78%)	1933(+72%)	1592(+37%)

Table 13: Comparisons with the state-of-the-art token sparsification method EViT on LV-ViT-S.

Method	Top-1 Acc	FLOPs(G)
EViT (Liang et al., 2022)	82.5(-0.8)	3.9(-41%)
EViT (Liang et al., 2022)	83.0(-0.3)	4.7(-29%)
STViT(Ours)	82.7(-0.6)	3.7(-44%)
STViT(Ours)	83.1(-0.2)	4.6(-30%)


(a) The attention map (16 × 216) of the second attention layer in STGM.


(b) The attention maps (14 × 14) of 16 semantic tokens in the second attention layer of STGM.


(c) Some examples of attention maps (16 × 16) of the self-attention layers after STGM.

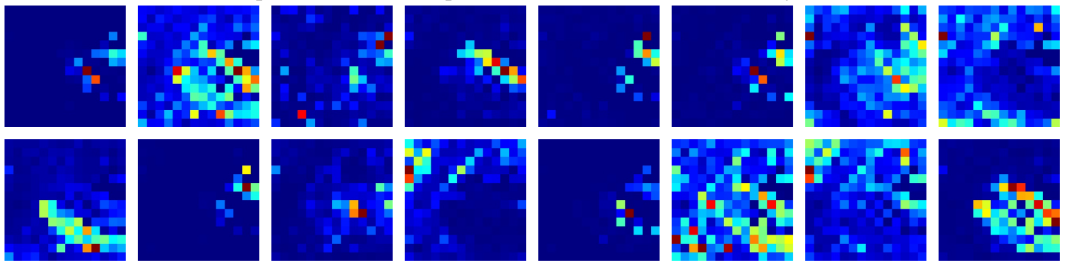

(d) The attention maps (14 × 14) of 16 semantic tokens generated by single global initialization.

Figure 4: Additional visualization of attention maps.

Table 14: STViT-R is evaluated on Swin-S and Swin-B on ImageNet. The top-1 accuracy, complexity in FLOPs, and throughput are reported.

STViT-R-Swin-S			STViT-R-Swin-B		
Top-1 Acc(%)	FLOPs(G)	Throughput(img/s)	Top-1 Acc(%)	FLOPs(G)	Throughput(img/s)
82.7(-0.3)	5.83(-33%)	717(+30%)	83.2(-0.3)	10.26(-33%)	539(+30%)

Table 15: Results of semantic segmentation on the ADE20K val set. A multi-scale inference with resolution $[0.5, 0.75, 1.0, 1.25, 1.5, 1.75]\times$ is applied. FLOPs and latency are measured in backbones with resolution 512×512.

Method	Backbone	mIoU	FLOPs(G)	Latency(ms)
UperNet	Swin-S	49.3	49	28
UperNet	STViT-R-Swin-S	48.3	34(-31%)	21(-27%)
UperNet	Swin-B	49.7	87	44
UperNet	STViT-R-Swin-B	48.9	60(-31%)	32(-28%)

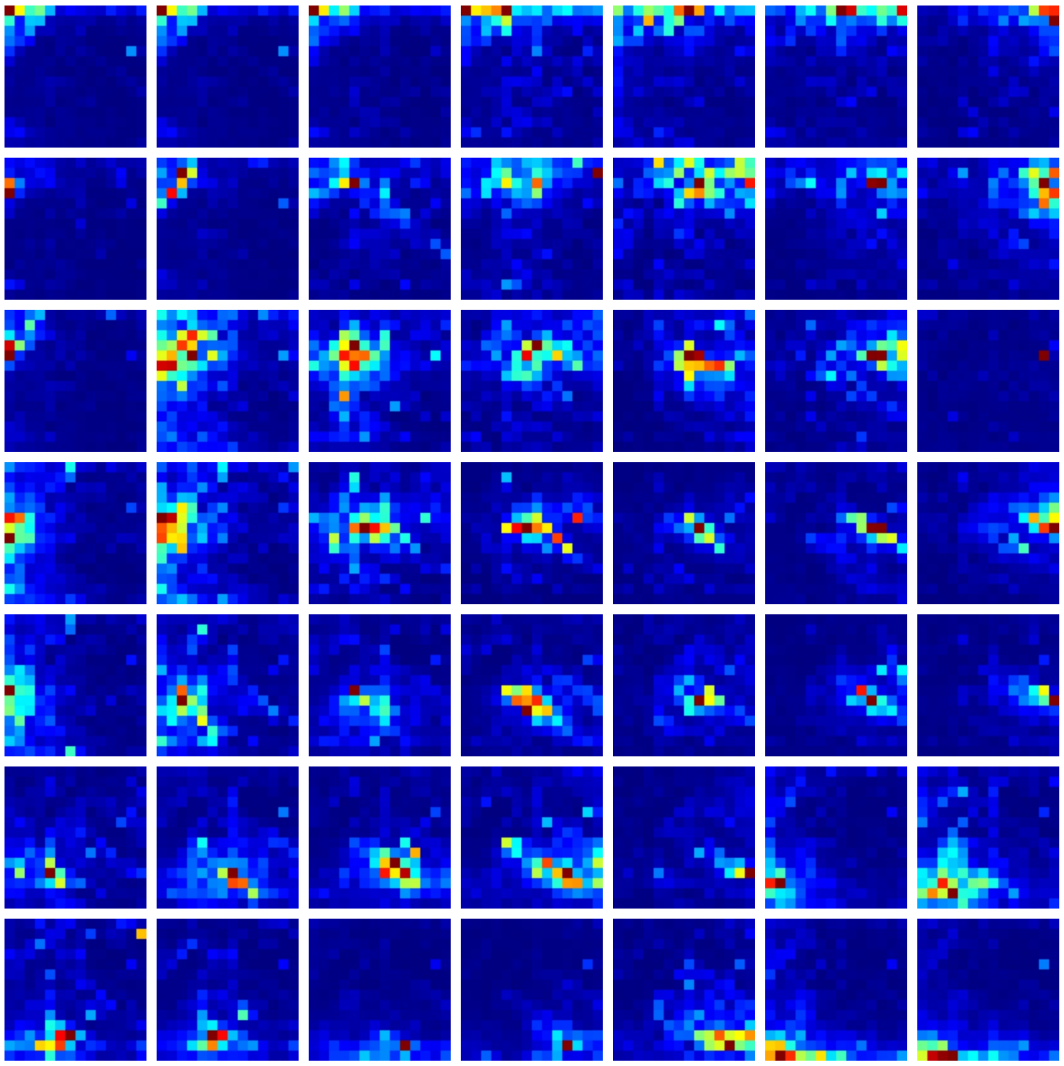


Figure 5: Attention response of 49 image tokens (7×7) uniformly sampled from 196 image tokens (14×14) in the 5^{th} transformer layer of DeiT-S.

Table 16: Performance evaluation on the different positions of our STGM.

Pos.	3-5	4-6	5-7	7-9	8-10	10-12
Top-1 Acc(%)	80.1	80.6	80.6	81.1	81.1	80.6
FLOPs(G)	1.62	1.97	2.31	3.01	3.36	4.06

Table 17: Performance evaluation on different positional encoding methods. *Learned*, *Conditional*, and *Relative* indicate learned positional encoding, conditional positional encoding, and relative positional encoding, respectively.

	STViT-DeiT-S Acc	STViT-Swin-T Acc
Learned	80.4	81.5
Conditional	80.4	81.4
Relative	80.6	81.3
No pos.	80.6	81.5

The position of STGM. The effects of different position of STGM are shown in Table 16. Two transformer layers are employed in STGM in all the experiments. We can see that the performance achieves improvement with appropriately moving the STGM towards deep layers due to better features of image tokens.

Positional encoding. We try to apply positional encoding to our semantic tokens. Table 17 shows comparisons of different positional encoding methods, including learned positional encoding, conditional positional encoding (Chu et al., 2021b), and relative positional encoding (Liu et al., 2021). All the positional encoding methods do not work on DeiT-S and Swin-T, even though relative positional encoding improves Swin-T by 1.2%. These experiments demonstrate that the interaction between our semantic tokens depends on high-level semantic information and nearly does not use position relationships.

Alternative schemes of spatial pooling. We use an intra and inter-window spatial pooling in STGM to generate initial cluster centers, which adaptively save meaningful semantic information and avoids overlap between adjacent windows as much as possible. Furthermore, we explore more spatial pooling schemes, including: (i) spatial pooling with large-size kernel and overlap, (ii) multi-scale spatial pooling, and (iii) adaptive spatial pooling. We adopt 25 semantic tokens in these experiments. In (i), the kernel size and overlap are set to 6 and 4, respectively. In (ii), we use two adaptive pooling layers which produce 9 and 16 tokens separately. The results are presented in Table 18 on DeiT-T. We can see that overlap and multiple scales cannot boost the performance, which also demonstrates that discrete semantic tokens with high-level semantic information benefit our method.

Table 18: Alternative schemes of spatial pooling.

	Scheme i	Scheme ii	Adaptive spatial pooling	Intra and inter-window spatial pooling
Top-1 Acc(%)	74.9	75.0	75.2	75.5

A.8 Cluster center recovery by self attention

We present an analysis showing how cluster centers are recovered through the attention mechanism. Let K be the number of clusters. Let $\mathcal{N}(\mu_i, \sigma^2 I/d), i = 1, \dots, K$ be the K Gaussian distributions, with center $\mu_i \in \mathbb{R}^d$ and covariance matrix $\sigma^2 I/d$. Let $x_{i,j} \in \mathbb{R}^d, j = 1, \dots, n,$ be the n data points independently sampled from $\mathcal{N}(\mu_i, \sigma^2 I/d)$. Given data points $\mathcal{D} = \{x_{i,j}, i \in [K], j \in [n]\}$, of course without knowing the association of each data point to its underlying Gaussian distribution, our goal is to recover the underlying cluster centers $\mu_i, i \in [K]$. We assume that all the center vectors of Gaussian distributions are well separated, i.e. $\langle \mu_j, \mu_k \rangle \leq \gamma$ if $j \neq k$. For the convenience of study, we assume $|\mu_i| = 1, i \in [K]$.

Let $\widehat{\mu}_i \in \mathbb{R}^d, i \in [K]$ the initialized cluster centers, with all the cluster centers being well normalized. Define Δ as the gap for any initialized $\widehat{\mu}_i$ to the target cluster centers μ_i than to other clusters μ_j, i.e.

$$\Delta = \min_{i \in [K]} \min_{j \neq i} \langle \widehat{\mu}_i, \mu_i - \mu_j \rangle$$

The new cluster centers are estimated through the self-attention mechanism, i.e.

$$\widehat{\mu}'_k = \frac{1}{Z_k} \sum_{i=1}^{K} \sum_{j=1}^{n} \exp\left(\lambda \langle \widehat{\mu}_k, x_{i,j} \rangle\right) x_{i,j}$$

where $\lambda > 0$ is a scaling factor and Z_i is defined as

$$Z_k = \sum_{i=1}^{K} \sum_{j=1}^{n} \exp\left(\lambda \langle \widehat{\mu}_k, x_{i,j} \rangle\right)$$

Theorem 1. *With sufficiently large d and $n \gg d$, with a probability $1 - O(K/n^2)$, we have*

$$\frac{\langle \mu_k, \widehat{\mu}'_k \rangle}{|\widehat{\mu}'_k|} \geq 1 - O\left(\frac{\log K + \log d}{d\Delta}\right)$$

Proof. Define $u_{i,j} = x_{i,j} - \mu_i$. We have

$$\widehat{\mu}'_k = \frac{1}{Z_k}\sum_{i=1}^K \exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle\right)\left\{\left(\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\right)\mu_i + \sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)u_{i,j}\right\}$$

We first bound $\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)$. Since $u_{i,j} \sim \mathcal{N}(0, \sigma^2 I/d)$ and $|\widehat{\mu}_k| = 1$, we know that $\langle\widehat{\mu}_k, u_{i,j}\rangle \sim \mathcal{N}(0, \sigma^2/d)$. Hence, with a probability $1 - 2\delta$, we have

$$\left|\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right) - n\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(\lambda x)\right]\right| \leq 3\exp\left(\lambda\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}}\right) + 2\sqrt{n\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(2\lambda x)\right]\log\frac{2}{\delta}}$$

Since

$$\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(\lambda x)\right] = \sqrt{\frac{d}{2\pi\sigma}}\int_{-\infty}^{+\infty}\exp\left(\lambda x - \frac{x^2 d}{2\sigma^2}\right)dx = \exp\left(\frac{\lambda^2\sigma^2}{2d}\right)$$

we have, with a probability $1 - 2\delta$,

$$\left|\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right) - n\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)\right| \leq 3\exp\left(\lambda\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}}\right) + 2\sqrt{n\exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\log\frac{2}{\delta}}$$

With large enough n, we have

$$3\exp\left(\lambda\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}}\right) + 2\sqrt{n\exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\log\frac{2}{\delta}} \leq C\sqrt{n}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)$$

and therefore

$$(1-\tau)n\exp\left(\frac{\lambda^2\sigma^2}{2d}\right) \leq \sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right) \leq (1+\tau)n\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)$$

where

$$\tau \leq \frac{C}{\sqrt{n}}$$

Here $C > 0$ is a universal constant.

We second bound $\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)u_{i,j}$. We write each $u_{i,j} = u_{i,j}^{\perp} + u_{i,j}^{\parallel}$, where $u_{i,j}^{\parallel} = \langle u_{i,j}, \widehat{\mu}_k\rangle\widehat{\mu}_k$ and $u_{i,j}^{\perp}$ is a $d-1$ dimensional Gaussian vector. We have

$$\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)u_{i,j} = \left(\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\widehat{\mu}_k + \sum_{j=1}^n u_{i,j}^{\perp}$$

Since $u_{i,j}^{\perp} \sim \mathcal{N}\left(0, \sigma^2 I_{d-1}/d\right)$, we have $\sum_{j=1}^n u_{i,j}^{\perp} \sim \mathcal{N}\left(0, n\sigma^2 I_{d-1}/d\right)$. Using the concentration of χ_{d-1}^2 distribution, we have, with a probability $1 - \delta$

$$\left|\sum_{j=1}^n u_{i,j}^{\perp}\right|^2 \leq \frac{n\sigma^2}{d}\left(d - 1 + 2\sqrt{(d-1)\log\frac{1}{\delta}} + 2\log\frac{1}{\delta}\right) \leq n\sigma^2\left(1 + 3\sqrt{\frac{\log(1/\delta)}{d}}\right)$$

To bound $\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\langle\widehat{\mu}_k, u_{i,j}\rangle$, following the same procedure, we have, with a probability $1 - 2\delta$

$$\begin{aligned}&\left|\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\langle\widehat{\mu}_k, u_{i,j}\rangle - n\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(\lambda x)x\right]\right| \\ \leq\ & 3\exp\left(\lambda\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}}\right)\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}} + \sqrt{n\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(2\lambda x)x^2\right]\frac{2}{\delta}}\end{aligned}$$

Since

$$\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(\lambda x)x\right] = \sqrt{\frac{d}{2\pi\sigma^2}}\int \exp\left(\lambda x - \frac{x^2 d}{2\sigma^2}\right)x dx = \frac{\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)$$

and

$$\begin{aligned}
&\mathrm{E}_{x\sim\mathcal{N}(0,\sigma^2/d)}\left[\exp(2\lambda x)x^2\right]\\
= \;& \sqrt{\frac{d}{2\pi\sigma^2}}\int \exp\left(2\lambda x - \frac{x^2 d}{2\sigma^2}\right)x^2 dx\\
= \;& \sqrt{\frac{d}{2\pi\sigma^2}}\exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\int \exp\left(\frac{d}{2\sigma^2}\left[x - \frac{\lambda\sigma^2}{d}\right]^2\right)\left(\left[x-\frac{\lambda\sigma^2}{d}\right]^2 + 2\frac{\lambda\sigma^2}{d}\left[x - \frac{\lambda\sigma^2}{d}\right] + \frac{\lambda^2\sigma^4}{d^2}\right)dx\\
= \;& \exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\left(\frac{\lambda^2\sigma^4}{d^2} + \left(\frac{2\sigma^2}{d}\right)^{3/2}\mathrm{E}_{x\sim\mathcal{N}(0,1)}\left[x^2\right]\right)\\
= \;& \exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\left(\frac{\lambda^2\sigma^4}{d^2} + 2\left(\frac{2\sigma^2}{d}\right)^{3/2}\right) \le \frac{2\lambda^2\sigma^4}{d^2}\exp\left(\frac{2\lambda^2\sigma^2}{d}\right)
\end{aligned}$$

We thus have, with a probability $1-2\delta$,

$$\begin{aligned}
&\left|\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\langle\widehat{\mu}_k, u_{i,j}\rangle - \frac{n\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)\right|\\
&\quad\le \; 3\exp\left(\lambda\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}}\right)\sigma\sqrt{\frac{2}{d}\log\frac{n}{\delta}} + \sqrt{\frac{4n\lambda^2\sigma^4}{d^2}\exp\left(\frac{2\lambda^2\sigma^2}{d}\right)\log\frac{2}{\delta}}
\end{aligned}$$

When n is sufficiently large, we have, with a probability $1-2\delta$

$$(1-\tau)\frac{n\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right) \le \sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)\langle\widehat{\mu}_k, u_{i,j}\rangle \le (1+\tau)\frac{n\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)$$

where $\tau \le C/\sqrt{n}$. By putting them together, with a probability $1-4\delta$, we have

$$\sum_{j=1}^n \exp\left(\lambda\langle\widehat{\mu}_k, u_{i,j}\rangle\right)u_{i,j} = n(1+\beta)\frac{n\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)\widehat{\mu}_k + n\nu_i$$

with $\beta\in[1-\tau, 1+\tau]$ and

$$|\nu_i| \le \frac{2\sigma}{\sqrt{n}}$$

Finally, we have, with a probability $1-4K\delta$

$$\mu_k' = \frac{n}{Z_k}\sum_{i=1}^K \exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle\right)\left((1+\alpha_i)\exp\left(\frac{\lambda^2\sigma^2}{2d^2}\right)\mu_i + (1+\beta_i)\frac{\lambda\sigma^2}{d}\exp\left(\frac{\lambda^2\sigma^2}{2d}\right)\widehat{\mu}_k + \nu\right)$$

Using the same analysis, we have, with a probability $1-4K\delta$,

$$\frac{Z_k}{n} = \exp\left(\frac{\lambda^2\sigma^2}{2d^2}\right)\sum_{i=1}^K \exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle\right)(1+\alpha_i)$$

Now, we can bound $|\widehat{\mu}_k' - \mu_k|$. We have

$$\begin{aligned}
&|\mu_k - \widehat{\mu}_k'|\\
\le \;& \left|\frac{n}{Z_k}\exp\left(\lambda\langle\mu_k, \widehat{\mu}_k\rangle + \frac{\lambda^2\sigma^2}{2d^2}\right)(1+\alpha_k) - 1\right| + \frac{n}{Z_k}\sum_{i\neq k}\exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle + \frac{\lambda^2\sigma^2}{2d^2}\right)(1+\alpha_i)\\
&+\frac{n\lambda\sigma^2|\mu_k - \widehat{\mu}_k|}{Z_k d}\sum_{i=1}^K \exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle + \frac{\lambda^2\sigma^2}{2d^2}\right)(1+\beta_i) + \frac{n}{Z_k}\sum_{i=1}^K \exp\left(\lambda\langle\mu_i, \widehat{\mu}_k\rangle\right)\nu_i
\end{aligned}$$

To further develop the bound for $|\mu_k - \widehat{\mu}'_k|$, we have

$$Z_k \geq n \exp\left(\frac{\lambda^2\sigma^2}{2d^2} + \lambda\langle \mu_k, \widehat{\mu}_k\rangle\right)\left(1 - \frac{C}{\sqrt{n}}\right)$$

and

$$Z_k \leq n \exp\left(\frac{\lambda^2\sigma^2}{2d^2} + \lambda\langle \mu_k, \widehat{\mu}_k\rangle\right)\left(1 + \frac{C}{\sqrt{n}}\right)(1 + (K-1)\exp(-\lambda\Delta))$$

We thus have

$$\begin{aligned} & |\mu_k - \widehat{\mu}'_k| \\ \leq & \left|\frac{\exp(\lambda\Delta)}{\exp(\lambda\Delta) + K - 1}\left(1 - \frac{2C}{\sqrt{n}}\right)^2 - 1\right| + \left(1 + \frac{2C}{\sqrt{n}}\right)^2 \frac{K-1}{\exp(\lambda\Delta)}\left(1 + \frac{\lambda\sigma^2}{d}|\mu_k - \widehat{\mu}_k|\right) \\ & + \frac{\lambda\sigma^2}{d}|\mu_k - \widehat{\mu}_k| + \left(1 + \frac{2C}{\sqrt{n}}\right)\frac{\sigma}{\sqrt{n}} \end{aligned}$$

By choosing $\lambda = (\log d + \log K)/\Delta$, and by assuming n is significantly larger than d, we have

$$|\mu_k - \widehat{\mu}'_k| \leq O\left(\frac{\log d + \log K}{d\Delta}\right)$$

implying that

$$\frac{\langle \mu_k, \widehat{\mu}'_k\rangle}{|\widehat{\mu}'_k|} \geq 1 - O\left(\frac{\log d + \log K}{d\Delta}\right)$$

□