Fast Vision Transformers with HiLo Attention

Anonymous Author(s)
Affiliation
Address
email

Abstract

Vision Transformers (ViTs) have triggered the most recent and significant breakthroughs in computer vision. Their efficient designs are mostly guided by the indirect metric of computational complexity, *i.e.*, FLOPs, which however has a clear gap with the direct metric such as throughput. Thus, we propose to use the direct speed evaluation on the target platform as the design principle for efficient ViTs. Particularly, we introduce LITv2, a simple and effective ViT which performs favourably against the existing state-of-the-art methods across a spectrum of different model sizes with faster speed. At the core of LITv2 is a novel self-attention mechanism, which we dub **HiLo**. HiLo is inspired by the insight that high frequencies in an image capture local fine details and low frequencies focus on global structures, whereas a multi-head self-attention layer neglects the characteristic of different frequencies. Therefore, we propose to disentangle the high/low frequency patterns in an attention layer by separating the heads into two groups, where one group encodes high frequencies via self-attention within each local window, and another group performs the attention to model the global relationship between the average-pooled low-frequency keys from each window and each query position in the input feature map. Benefit from the efficient design for both groups, we show that HiLo is superior to the existing attention mechanisms by comprehensively benchmarking on FLOPs, speed and memory consumption on GPUs. Powered by HiLo, LITv2 serves as a strong backbone for mainstream vision tasks including image classification, dense detection and segmentation.

1 Introduction

Real-world applications usually require a model to have an optimal speed and accuracy trade-off under limited computational budget, such as UAV and autonomous driving. This motivates substantial works toward efficient vision Transformer (ViT) design, such as PVT [40], Swin [25] and Focal Transformer [47], among others. To measure the computational complexity, a widely adopted metric in recent ViT design is the number of float-point operations, *i.e.*, FLOPs. However, FLOPs is an indirect metric, which can not directly reflect the real speed on the target platform. For example, Focal-Tiny is much slower than Swin-Ti on GPUs although their FLOPs are comparable.

In general, the discrepancy between the indirect metric (FLOPs) and the direct metric (speed) in recent ViTs can be attributed to two main reasons. First, although self-attention is efficient on low-resolution feature maps, the quadratic complexity in both memory and time makes it much slower on high-resolution images due to intensive memory access cost [27], where fetching data from off-chip DRAM can be speed-consuming. Second, some efficient attention mechanisms in ViTs have low theoretical complexity guarantee but are actually slow on GPUs due to particular operations that

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

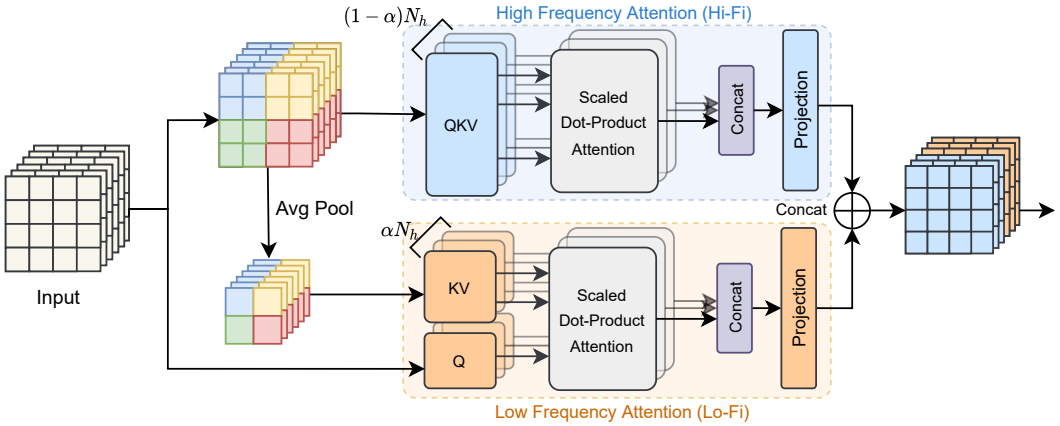

Figure 1: Framework of HiLo attention. N_h refers to the total number of self-attention heads at this layer. α denotes the split ratio for high/low frequency heads. Best viewed in color.

are not hardware-friendly or cannot be parallelized, such as the multi-scale window partition [47], recursion [35] and dilated window [15].

With these observations, in this paper we propose to evaluate ViT by the direct metric, *i.e.*, throughput, not only FLOPs. Based on this principle, we introduce LITv2, a novel efficient and accurate vision Transformer that outperforms most state-of-the-art (SoTA) ViTs on standard benchmarks while being practically faster on GPUs. LITv2 is bulit upon LITv1 [29], a simple ViT baseline which removes all multi-head self-attention layers (MSAs) in the early stages while applying standard MSAs in the later stages. Benefit from this design, LITv1 is faster than many existing works on ImageNet classification due to no computational cost from the early MSAs while the later MSAs only need to process downsampled low-resolution feature maps. However, the standard MSA still suffers from huge computational cost on high-resolution images, especially for dense prediction tasks.

To address this problem, we propose a novel efficient attention mechanism, termed **HiLo**. HiLo is motivated by the fact that natural images contain rich frequencies where high/low frequencies play different roles in encoding image patterns, *i.e.*, local fine details and global structures, respectively. A typical MSA layer enforces the same global attention across all image patches without considering the characteristics of different underlying frequencies. This motivates us to propose to separate an MSA layer into two paths where one path encodes high-frequency interactions via local self-attention with relatively high-resolution feature maps while the other path encodes low-frequency interactions via global attention with down-sampled feature maps, which leads to a great efficiency improvement.

Specifically, HiLo employs two efficient attentions to disentangle **Hi**gh/**Lo**w frequencies in feature maps. As shown in Figure 1, in the upper path, we allocate a few heads to the high frequency attention (Hi-Fi) to capture fine-grained high frequencies by local window self-attention (*e.g.*, 2×2 windows), which is much more efficient than standard MSAs. The lower path, implementing the low-frequency attention (Lo-Fi), first applies average pooling to each window to obtain low-frequency signals. Then, we allocate the remaining heads for Lo-Fi to model the relationship between each query position in the input feature map and the average-pooled low-frequency keys from each window. Benefit from the reduced length of keys and values, Lo-Fi also achieves significant complexity reduction. Finally, we concatenate the refined high/low-frequency features and forward the resulting output into subsequent layers. Since both Hi-Fi and Lo-Fi are not equipped with time-consuming operations such as dilated windows and recursion, the overall framework of HiLo is fast on GPUs. We show by comprehensive benchmarks that HiLo achieves advantage over the existing attention mechanisms in terms of performance, FLOPs, throughput and memory consumption.

Besides, we find the fixed relative positional encoding in LITv1 dramatically slows down its speed on dense prediction tasks due to the interpolation for different image resolutions. For better efficiency, we propose to adopt one 3×3 depthwise convolutional layer with zero-padding in each FFN to incorporate the implicitly learned position information from zero-padding [20]. Moreover, the 3×3 convolutional filters simultaneously help to enlarge the receptive field of the early multi-layer perceptron (MLP) blocks in LITv1. Finally, we conduct extensive experiments on ImageNet, COCO and ADE20K to evaluate the performance of LITv2. Comprehensive comparisons with SoTA models show that our architecture achieves competitive performance with faster throughput, making ViTs more feasible to run low-latency applications for real-world scenarios.

2 Related Work

Vision Transformers. Vision Transformers are neural networks that adopt self-attention mechanisms into computer vision tasks. In [13], Dosovitskiy *et al*. propose a ViT for image classification, which inherits the similar architecture from a standard Transformer [37] in natural language processing (NLP) tasks. Since then, subsequent works have been proposed to improve ViT by incorporating more convolutional layers [43, 48], introducing pyramid feature maps [40, 25], enhancing the locality [49], as well as automatically searching a well-performed architecture [4, 2] with neural architecture search (NAS). Some others also seek for token pruning to accelerate the inference speed of ViTs [30]. Compared to existing works, this paper focuses on a general ViT-based backbone for computer vision (CV) tasks and aims to achieve better efficiency on GPUs while maintaining competitive performance.

Efficient attention mechanisms. Efficient attention mechanisms aim to reduce the quadratic complexity of standard MSAs. Existing efforts in NLP can be roughly categories into low-rank decomposition [39], kernelization [21, 32], memory [33] and sparsity mechanism [7]. However, simply adopting these method usually performs suboptimally in CV tasks [25, 50]. In CV, representative efficient self-attention mechanisms includes spatial reduction attention (SRA) [40], local window attention [25] and Twins attention [9]. However, they only focus on either local or global attention at the same layer, which neglects the another. Some works consider both simultaneously, such as Focal attention [47] and QuadTree [35]. However, due to the inefficient operations which are not hardware-friendly and cannot be reflected in FLOPs (*e.g*., multi-scale window partition, recursion), they are slow on GPUs even compared to standard MSA. To this end, the proposed HiLo attention simultaneously captures rich local-global information at the same MSA layer and is faster and more memory-efficient compared to the existing works.

Frequency domain analysis in vision. The frequency domain analysis in CV has been well studied in the literature. According to [10, 12], the low frequencies in an image usually capture global structures and color information while the high frequencies contain fine details of objects (*e.g*., sharp edges). Based on this insight, a plethora of solutions have been proposed for image super-resolution [52, 14], generalization [19], image re-scaling [44] and neural network compression [46, 5]. In the same spirit, HiLo separately deals with low and high frequencies in an MSA layer to achieve better efficiency.

3 Background

Multi-head self-attention. Transformers are built upon multi-head self-attention, which enables to capture long-range relationships for tokens at different positions. Specifically, let $\mathbf{X} \in \mathbb{R}^{N \times D}$ be the input sequence into a standard MSA layer, where N is the length of the input sequence and D refers to the number of hidden dimensions. Each self-attention head calculates the query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$ matrices with a linear transformation from $\mathbf{X}$,

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \mathbf{K} = \mathbf{X}\mathbf{W}_k, \mathbf{V} = \mathbf{X}\mathbf{W}_v, \tag{1}$$

where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{D \times D_h}$ are learnable parameters and D_h is the number of hidden dimensions for a head. Next, the output of a self-attention head is a weighted sum over N value vectors,

$$\mathrm{SA}_h(\mathbf{X}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_h}})\mathbf{V}. \tag{2}$$

For an MSA layer with N_h heads, the final output is computed by a linear projection of the concatenated outputs from each self-attention head, which can be formulated by

$$\mathrm{MSA}(\mathbf{X}) = \underset{h \in [N_h]}{\mathrm{concat}}[\mathrm{SA}_h(\mathbf{X})]\mathbf{W}_o, \tag{3}$$

where $\mathbf{W}_o \in \mathbb{R}^{(N_h \times D_h) \times D}$ is a learnable parameter. In practice, D is usually equal to $N_h \times D_h$. Overall, a standard MSA layer have the computational cost of $4ND^2 + 2N^2D$, where $2N^2D$ comes from Eq. (2), $3ND^2$ and ND^2 comes from Eq. (1) and Eq. (3), respectively.

Transformer blocks. A standard vision Transformer as described in [13] consists of a patch embedding layer, several blocks and a prediction head. Let l be the index of a block. Then each block contains an MSA layer and a position-wise feed-forward network (FFN), which can expressed as

$$\mathbf{X}^{'}_{l-1} = \mathbf{X}_{l-1} + \mathrm{MSA}(\mathrm{LN}(\mathbf{X}_{l-1})), \tag{4}$$

$$\mathbf{X}_{l} = \mathbf{X}^{'}_{l-1} + \mathrm{FFN}(\mathrm{LN}(\mathbf{X}^{'}_{l-1})), \tag{5}$$

where LN denotes the LayerNorm [1] and an FFN consists of two FC layers with GELU [18] non-linearity in between. Recent works on ViT have proposed to divide the blocks into several stages (typically 4 stages) to generate pyramid feature maps for dense prediction tasks. Furthermore, to reduce the computational cost on high-resolution feature maps in the early stages, the MSA in Eq. (4) has been replaced with efficient alternatives, such as SRA [40] and W-MSA [25].

Bottlenecks of LITv1. Recent studies have shown that the MSA layers in the early stages in a model still focus on local patterns [11]. With the same observation, LITv1 [29] removes all early MSAs (*i.e.*, exclude Eq. (4) in each block) while applying standard MSAs at the later stages. This design principle has achieved better efficiency with competitive performance on ImageNet compared to PVT [40] and Swin [25]. However, LITv1 still has two main bottlenecks in speed: 1) Given a high-resolution image, the standard MSAs in the later stages still result in huge computational cost. 2) The fixed relative positional encoding [25] dramatically slows down the speed when dealing with different image resolutions. This is due to interpolating the fixed-size positional encoding for each different image resolution. In the next section, we describe a novel attention mechanism with zero padding positional encoding to comprehensively accelerate LITv1.

4 Method

4.1 HiLo Attention

Natural images contain rich frequencies where high frequencies capture local fine details of objects (*e.g.*, lines and shapes) and low frequencies encode global structures (*e.g.*, textures and colors). However, the global self-attention in a typical MSA layer does not consider the characteristics of different underlying frequencies. To this end, we propose to separately process high/low frequencies in a feature map at an attention layer. We name the new attention mechanism as HiLo, which is depicted in Figure 1. As it shows, HiLo contains a **Hi**gh-**F**requency attent**i**on (Hi-Fi) and **Lo**w **F**requency attent**i**on (Lo-Fi) to model the relationship on different frequencies in feature maps. In the next, we describe the two attentions in detail.

High-frequency attention (Hi-Fi). Intuitively, as high frequencies encode local details of objects, it can be redundant and computationally expensive to apply global attention on a feature map. Therefore, we propose to design Hi-Fi to capture fine-grained high frequencies with local window self-attention (*e.g.*, 2×2 windows), which saves significant computational complexity. Furthermore, we employ the simple non-overlapping window partition in Hi-Fi, which is more hardware-friendly compared to the time-consuming operations such as window shifting [25] or multi-scale window partition [47].

Low-frequency attention (Lo-Fi). Recent studies have shown that the global attention in MSA helps to capture low frequencies [31]. However, directly applying MSA to high-resolution feature maps requires huge computational cost. As averaging is a low-pass filter [38], Lo-Fi firstly applies average pooling to each window to get low-frequency signals in the input $\mathbf{X}$. Next, the average-pooled feature maps are projected into keys $\mathbf{K} \in \mathbb{R}^{N/s^2 \times D_h}$ and values $\mathbf{V} \in \mathbb{R}^{N/s^2 \times D_h}$, where s is the window size. The queries $\mathbf{Q}$ in Lo-Fi still comes from the original feature map $\mathbf{X}$. We then apply the standard attention to capture the rich low-frequency information in feature maps. Note that due to the spatial reduction of $\mathbf{K}$ and $\mathbf{V}$, Lo-Fi simultaneously reduces the complexity for both Eq. (1) and Eq. (2).

Head splitting. A naive solution for head assignment is to allocate both Hi-Fi and Lo-Fi the same number of heads as the standard MSA layer. However, doubling heads results in more computational cost. In order to achieve better efficiency, HiLo separates the same number of heads in an MSA into two groups with a split ratio α, where $(1-\alpha)N_h$ heads will be employed for Hi-Fi and the other αN_h heads are used for Lo-Fi. By doing so, as each attention has a lower complexity than a standard MSA, the entire framework of HiLo guarantees a low complexity and ensures high throughput on GPUs. Moreover, another benefit of head splitting is that the learnable parameter $\mathbf{W}_o$ can be decomposed into two smaller matrices, which helps to reduce model parameters. Finally, the output of HiLo is a concatenation of the outputs from each attention

$$\text{HiLo}(\mathbf{X}) = [\text{Hi-Fi}(\mathbf{X}); \text{Lo-Fi}(\mathbf{X})], \tag{6}$$

where $[\cdot]$ denotes the concatenation operation.

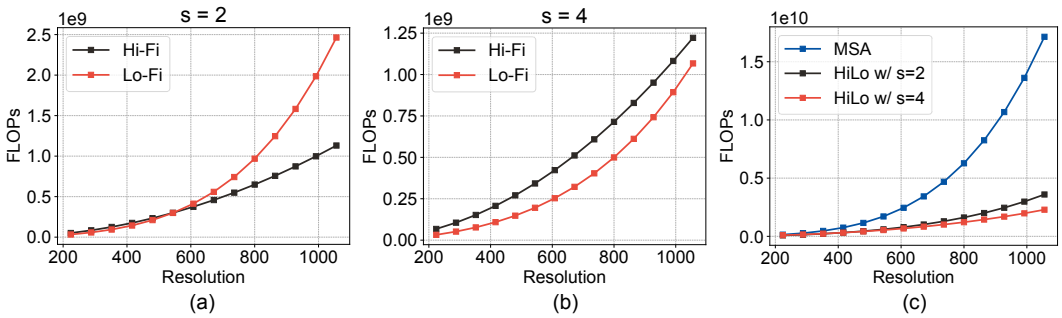

Figure 2: FLOPs comparison for Hi-Fi and Lo-Fi under different image resolutions and equal number of heads (Figures a and b). A larger window size helps HiLo achieve better efficiency on high-resolution images (Figure c).

Complexity Analysis. Without loss of generality, we assume Hi-Fi and Lo-Fi have an equal number of heads (*i.e.*, $\alpha = 0.5$) and the feature map has equal width and height. Then, Hi-Fi and Lo-Fi have a computational cost of $\frac{7}{4}ND^2 + s^2ND$ and $(\frac{3}{4} + \frac{1}{s^2})ND^2 + \frac{1}{s^2}N^2D$, respectively. Derivation for this result can be found in the supplementary material. As shown in Figure 2-(a) and (b), under a small input image resolution and a small value of s (*e.g.*, $s = 2$), both Hi-Fi and Lo-Fi are comparably efficient. However, with a much higher resolution, Lo-Fi will result in a huge computational cost as it still has a quadratic complexity in terms of N in Eq. (2), *i.e.*, $\frac{1}{s^2}N^2D$. In this case, slightly increasing s (*e.g.*, $s = 4$) helps Lo-Fi achieve better efficiency while preserving the accuracy. Combining the two attentions together, a larger window size also helps the overall framework of HiLo to reduce more FLOPs on high-resolution images, as shown in Figure 2-(c). Thus, we suggest a practical guideline for adopting HiLo into existing frameworks: *increasing the window size in order to get better efficiency on high-resolution images.* We further show in Section 5.2 that this principle helps LITv2 achieve a better speed and accuracy trade-off on downstream tasks, *e.g.*, dense object detection.

4.2 Positional Encoding

Positional encoding is essential to self-attention due to its permutation-invariant property. In LITv1, the later MSAs adopt the same relative positional encoding (RPE) scheme as Swin [25]. This approach has significantly improves Swin by 0.7% in Top-1 accuracy on ImageNet compared to using absolute positional encoding [25]. However, on dense prediction tasks, the fixed RPE has to be interpolated for different image resolutions, which dramatically slows down the training/inference speed of LITv1. As a recent study [20] has shown that position information can be implicitly learned from zero-padding in CNNs, we propose to adopt one layer of 3×3 depthwise convolutional layer with zero-padding in each FFN to replace the time-consuming RPE. Notably, due to the elimination of early MSAs, the early blocks in LITv1 only have FFNs left, which results in a tiny receptive field of 1×1. To this end, we show in Section 5.4 that the 3×3 convolutional filters adopted in each FFN also improve LITv2 by simultaneously enlarging the receptive field in the early stages.

4.3 Model Architecture

LITv2 has three variants: LITv2-S, LITv2-M and LITv2-B, corresponding to the small, medium and base settings in LITv1, respectively. For a fair comparison, we keep the network width and depth as the same as LITv1. The overall modifications are simply in two steps: 1) Adding one layer of depthwise convolution with zero-padding in each FFN and removing all relative positional encodings in all MSAs. 2) Replacing all attention layers with the proposed HiLo attention. Detailed architecture configurations can be found in the supplementary material.

5 Experiment

In this section we conduct experiments to validate the effectiveness of the proposed LITv2. Following common practice [40, 25, 9, 47], we experiment LITv2 on three tasks, including image classification on ImageNet-1K [34], object detection and instance segmentation on COCO [24] and semantic segmentation on ADE20K [51].

Table 1: Image classification results on ImageNet-1K. By default, the FLOPs, throughput and memory consumption are measured based on the resolution 224×224. We report the throughput and training/test time memory consumption with a batch size of 64. Throughput is tested on one NVIDIA RTX 3090 GPU and averaged over 30 runs. ResNet results are from "ResNet Stikes Back" [42]. "$\uparrow 384$" means a model is finetuned at the resolution 384×384. "OOM" means "out-of-memory".

Model	Param (M)	FLOPs (G)	Throughput (imgs/s)	Train Mem (GB)	Test Mem (GB)	Top-1 (%)
ResNet-50 [42]	26	4.1	1,279	7.9	2.8	80.4
ConvNext-Ti [26]	28	4.5	1,079	8.3	1.7	82.1
PVT-S [40]	25	3.8	1,007	6.8	1.3	79.8
Swin-Ti [25]	28	4.5	961	6.1	1.5	81.3
CvT-13 [43]	20	4.5	947	6.1	1.5	81.6
Focal-Tiny [47]	29	4.9	384	12.2	3.3	**82.2**
Twins-PCPVT-S [9]	24	3.8	998	6.8	1.2	81.2
LITv1-S [29]	27	4.1	1,298	5.8	1.2	81.5
LITv2-S	28	3.7	**1,471**	**5.1**	**1.2**	82.0
ResNet-101 [42]	45	7.9	722	10.5	3.0	81.5
ConvNext-S [26]	50	8.7	639	12.3	1.8	83.1
PVT-M [40]	44	6.7	680	9.3	1.5	81.2
Twins-SVT-B [9]	56	8.3	621	9.8	1.9	83.2
Swin-S [25]	50	8.7	582	9.7	1.7	83.0
LITv1-M [29]	48	8.6	638	12.0	1.4	83.0
LITv2-M	49	7.5	**812**	**8.8**	**1.4**	**83.3**
ResNet-152 [42]	60	11.6	512	13.4	2.9	82.0
ConvNext-B [26]	89	15.4	469	16.9	2.9	**83.8**
Twins-SVT-L [9]	99	14.8	440	13.7	3.1	83.7
Swin-B [25]	88	15.4	386	13.4	2.4	83.3
LITv1-B [29]	86	15.0	444	16.4	2.1	83.4
LITv2-B	87	13.2	**602**	**12.2**	**2.1**	83.6
DeiT-B$\uparrow$ 384 [36]	86	55.4	159	39.9	**2.5**	83.1
Swin-B$\uparrow$ 384 [25]	88	47.1	142	OOM	6.1	84.5
LITv2-B$\uparrow$ 384	87	39.7	**198**	**35.8**	4.6	**84.7**

5.1 Image Classification on ImageNet-1K

We conduct image classification experiments on ImageNet-1K [34], a large-scale image dataset which contains $\sim$1.2M training images and 50K validation images from 1K categories. We measure the model performance by Top-1 accuracy. Furthermore, we report the FLOPs, throughput, as well as training/test memory consumption on GPUs. We compare with two CNN-based models [42, 26] and several representative SoTA ViTs [40, 25, 43, 47, 9]. Note that this paper does not consider mobile-level architectures [6, 28]. Instead, we focus on models with the similar model size. Besides, we are also not directly comparable with NAS-based methods [2, 4] as LITv2 is manually designed.

Implementation details. All models are trained for 300 epochs from scratch on 8 V100 GPUs. At training time, we set the total batch size as 1,024. The input images are resized and randomly cropped into 224×224. The initial learning rate is set to 1×10^{-3} and the weight decay is set to 5×10^{-2}. We use AdamW optimizer with a cosine decay learning rate scheduler. All training strategies including the data augmentation are same as in LITv1. For HiLo, the window size s is set to 2. The split ratio α is set to 0.9, which is chosen from a simple grid search on ImageNet-1K. The depthwise convolutional layers in FFNs are set with a kernel size of 3×3, stride of 1 and zero padding size of 1.

Results. In Table 1, we report the experiment results on ImageNet-1K. First, compared to LITv1 baselines, LITv2 achieves consistent improvement on Top-1 accuracy while using less FLOPs. Moreover, benefit from HiLo, LITv2 achieves faster throughput and significant training time memory reduction (*e.g.*, 13%, 27%, 36% inference speedup for the small, medium and base settings, respectively) compared to LITv1. Second, compared to CNNs, LITv2 models outperform all counterparts of ResNet and ConvNext in terms of FLOPs, throughput and memory consumption while achieving comparable performance. Last, compared to SoTA ViTs, LITv2 surpasses many models in terms of throughput and memory consumption with competitive performance. For example, under the

Table 2: Object detection and instance segmentation performance on the COCO `val2017` split using the RetinaNet [23] and Mask R-CNN [16] framework. AP^b and AP^m denote the bounding box AP and mask AP, respectively. "*" indicates the model adopts a local window size of 4 in HiLo.

Backbone	RetinaNet				Mask R-CNN				
	Params (M)	FLOPs (G)	FPS	AP^b	Params (M)	FLOPs (G)	FPS	AP^b	AP^m
ResNet-50 [17]	38	239	18.5	36.3	44	260	27.1	38.0	34.4
PVT-S [40]	34	273	13.0	40.4	44	292	16.2	40.4	37.8
Swin-T [25]	38	251	17.0	41.5	48	270	21.1	42.2	39.1
Twins-SVT-S [9]	34	225	15.5	43.0	44	244	20.4	43.4	40.3
LITv1-S [29]	39	305	3.3	41.6	48	324	3.2	42.9	39.6
LITv2-S	38	242	18.7	**44.0**	47	261	18.7	**44.9**	**40.8**
LITv2-S*	38	230	20.4	**43.7**	47	249	21.9	**44.7**	**40.7**
ResNet-101 [17]	57	315	15.2	38.5	63	336	20.9	40.4	36.4
PVT-M [40]	54	348	10.5	41.9	64	367	10.8	42.0	39.0
Swin-S [25]	60	343	13.3	44.5	69	362	15.8	44.8	40.9
Twins-SVT-B [9]	67	358	10.8	45.3	76	377	12.7	45.2	41.5
LITv2-M	59	348	12.2	**46.0**	68	367	12.6	**46.8**	**42.3**
LITv2-M*	59	312	14.8	**45.8**	68	315	16.0	**46.5**	**42.0**

similar amount of FLOPs, LITv2-S achieves faster inference speed than PVT-S and Twins-PCPVT-S with better performance. Although Focal-Tiny achieves better Top-1 accuracy than LITv2-S, it runs much slower (*i.e.*, 384 vs. 1,471 images/s) and requires a large amount of memory to train. Besides, when finetuning on a higher resolution, LITv2-B outperforms both DeiT-B and Swin-B with a faster throughput and lower complexity.

5.2 Object Detection and Instance Segmentation on COCO

In this section, we conduct experiments on COCO 2017, a common benchmark for object detection and instance segmentation which contains ∼118K images for the training set and ∼5K images for the validation set. Following common practice [9, 40], we experiment with two detection frameworks: RetinaNet [23] and Mask R-CNN [16]. We measure model performance by Average Precision (AP).

Implementation details. All backbones are initialized with pretrained weights on ImageNet-1K. We train each model on 8 GPUs with $1\times$ schedule (12 epochs) and a total batch size of 16. For a fair comparison, we adopt the same training strategy and hyperparameter settings as in LITv1 [29]. Note that we pretrain LITv2 with a local window size of 2 and $\alpha = 0.9$ on ImageNet-1K. Under the same α, a larger window size helps to achieve lower complexity and thus improves the speed at high resolution, as explained in Section 4.1. In this case, we also train models with a slightly larger window size of $s = 4$ for better efficiency, which we denote with "*". By default, FLOPs is evaluated based on the input resolution of 1280×800. FPS is measured on one RTX 3090 GPU based on the mmdetection [3] framework.

Results. In Table 2, we report the experimental results on COCO. In general, LITv2 outperforms LITv1 by a large margin in almost all metrics. Besides, our LITv2 significantly surpasses ResNet in terms of AP, though it runs slightly slower in some cases. More importantly, our LITv2 beats all the compared SoTA ViTs, achieving the best AP with compelling fast inference speed. Furthermore, by adopting a larger window size (*i.e.*, $s = 4$), LITv2 achieves better efficiency with a slightly performance drop.

5.3 Semantic Segmentation on ADE20K

In this section, we evaluate LITv2 on the semantic segmentation task. We conduct experiments on ADE20K [51], a widely adopted dataset for semantic segmentation which has ∼20K training images, ∼2K validation images and ∼3K test images. Following prior works, we adopt the framework of Semantic FPN [22] and measure the model performance by mIoU. We train each model on 8 GPUs with a total batch size of 16 with 80K iterations. All backbones are initialized with pretrained weights on ImageNet-1K. The stochastic depth for the small, medium and base models of LITv2 are 0.2, 0.2 and 0.3, respectively. All other training strategies are the same as in LITv1 [29].

Results. In Table 3, we compare LITv2 with ResNet and representative ViTs on ADE20K. In general, LITv2 achieves fast speed while outperforming many SoTA models. For example, our LITv2-S,

Table 4: Performance comparisons with other efficient attention mechanisms in ViTs based on LITv2-S. We report the Top-1 accuracy on ImageNet-1K.

Method	Param (M)	FLOPs (G)	Throughput (imgs/s)	Train Memory (GB)	Test Memory (GB)	Top-1 (%)
MSA	28	4.1	1,293	6.5	1.2	82.3
SRA [40]	32	4.0	1,425	5.1	1.3	81.7
W-MSA [25]	28	4.0	1,394	5.3	1.2	81.9
T-MSA [9]	30	4.0	1,462	**5.0**	1.3	81.8
HiLo	**28**	**3.7**	**1,471**	5.1	**1.2**	**82.0**

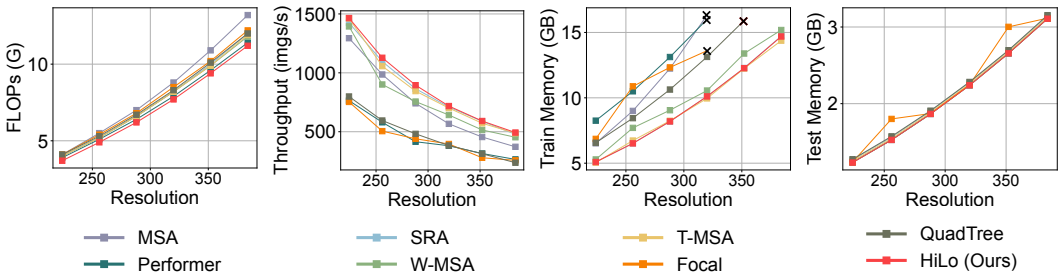

Figure 3: Comparison with other attention mechanisms based on LITv2-S. We report the FLOPs, throughput, and training/test time memory consumption. Evaluations are based on a batch size of 64 on one RTX 3090 GPU. The black cross symbol means "out-of-memory".

LITv2-M and LITv2-B surpass Swin-Ti, Swin-S and Swin-B by 2.8%, 0.5% and 1.2% in mIoU with higher FPS, respectively.

5.4 Ablation Study

In this section, we provide ablation studies for LITv2, including the comparison with other efficient attention variants, the effect of α in HiLo, as well as the effect of architecture modifications. By default, the throughput and memory consumption are measured on one RTX 3090 GPU with a batch size of 64 under the resolution of 224×224.

Table 3: Semantic segmentation performance of different backbones on the ADE20K validation set. FLOPs is evaluated based on the image resolution of 512×512.

Backbone	Params (M)	FLOPs (G)	FPS	mIoU (%)
ResNet-50 [17]	29	45	45.4	36.7
PVT-S [40]	28	40	38.7	39.8
Swin-Ti [25]	32	46	39.6	41.5
Twins-SVT-S [9]	28	37	34.5	43.2
LITv1-S [29]	32	46	18.1	41.7
LITv2-S	31	41	42.6	**44.3**
ResNet-101 [17]	48	66	36.7	38.8
PVT-M [40]	48	55	29.7	41.6
Swin-S [25]	53	70	24.4	45.2
Twins-SVT-B [9]	60	67	28.0	45.3
LITv2-M	52	63	28.5	**45.7**
PVT-L [40]	65	71	20.5	42.1
Swin-B [25]	107	107	25.5	46.0
Twins-SVT-L [9]	104	102	25.9	46.7
LITv2-B	90	93	27.5	**47.2**

Comparing HiLo with other attention mechanisms. Based on LITv2-S, we compare the performance of HiLo with other efficient attention mechanisms on ImageNet-1K, including spatial reduction attention (SRA) in PVT [40], shifted-window based attention (W-MSA) in Swin [25] and alternated local and global attention (T-MSA) in Twins [9]. In our implementation, we directly replace HiLo with each compared method. The results are reported in Table 4. In general, HiLo reduces more FLOPs while achieving better performance and faster speed than the compared methods. Furthermore, in Figure 3, we provide comprehensive benchmarks for more attention mechanisms based on different image resolutions, including Focal [47], QuadTree [35] and Performer [8]. Suffering from weak parallelizability, they are even slower than that of using standard MSAs on GPUs. Compared to them, HiLo achieves competitive results in terms of the FLOPs, throughput and memory consumption.

Effect of α. As shown in Figure 4, since the complexity of Lo-Fi is lower than Hi-Fi under the resolution of 224×224 and the window size of 2, a larger α helps to reduce more FLOPs as we allocate more heads to Lo-Fi. Moreover, we found HiLo performs badly with $\alpha = 0$, in which case only the Hi-Fi is left and HiLo only focuses on high frequencies. We speculate that low frequencies

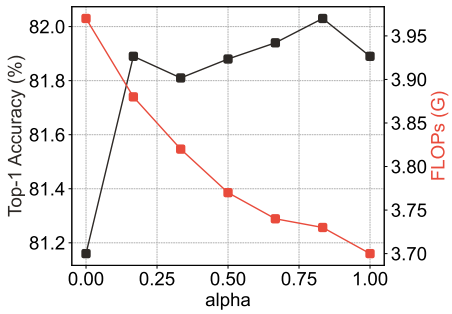

Figure 4: Effect of α based on LITv2-S.

Table 5: Effect of architecture modifications based on LITv1-S. "ConvFNN" means we add one layer of 3×3 depthwise convolutional layer into each FFN. "RPE" refers to relative positional encoding [25].

Name	ImageNet-1K			COCO (RetinaNet)		
	FLOPs (G)	Mem (GB)	Top-1 (%)	FLOPs (G)	FPS	AP
LITv1-S [29]	4.1	5.8	81.5	305	3.3	41.6
+ ConvFFN	4.1	6.5	82.5	306	3.1	45.1
+ Remove RPE	4.1	6.5	82.3	306	13.3	44.7
+ HiLo	3.7	5.1	82.0	224	18.7	44.0

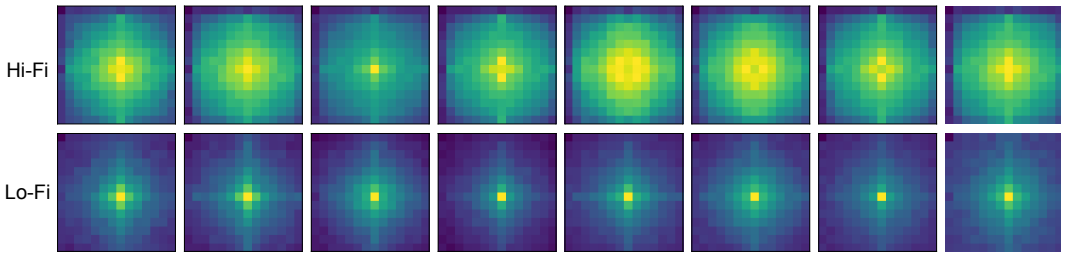

Figure 5: Frequency magnitude (14×14) from 8 output channels of Hi-Fi and Lo-Fi in LITv2-B. The magnitude is averaged over 100 samples. The lighter the color, the larger the magnitude. A pixel that is closer to the centre means a lower frequency.

play an important role in self-attention. For other values of α, we find the performance difference is around 0.2%, where $\alpha = 0.9$ achieves the best performance.

Effect of architecture modifications. Based on LITv2-S, we explore the effect of architecture modifications. As shown in Table 5, benefit from the enlarged receptive field in the early stages, the adoption of depthwise convolutions improves the performance on both ImageNet and COCO. Next, by removing the relative positional encoding, we significantly improve FPS on dense prediction tasks with a slightly performance drop on both datasets. Also note that since depthwise convolutions have encoded positional information by zero paddings [20], the elimination of RPE does not result in a significant performance drop compared to prior works [25]. Finally, benefit from HiLo, we achieve more gains in model efficiency on both ImageNet and COCO.

Spectrum analysis of HiLo. In Figure 5, we visualize the frequency magnitude of the output feature maps from Hi-Fi and Lo-Fi attentions, respectively. The visualisation indicates that Hi-Fi captures more high frequencies and Lo-Fi mainly focuses on low frequencies. This strongly aligns with our aim of disentangling high and low frequencies in feature maps at a single attention layer. We will provide more visualisation examples in the supplementary material.

6 Conclusion and Future Work

In this paper, we have introduced LITv2, a novel efficient vision Transformer backbone with fast speed on GPUs and outperforms most SoTA models on ImageNet and downstream tasks. We have also presented HiLo attention, the core of LITv2 which helps to achieve better efficiency especially on high-resolution images. With competitive performance, HiLo achieves great advantage over the existing attention mechanisms across FLOPs, throughput and memory consumption. Future work may include incorporating convolutional stem [45] and overlapping patch embedding [41] for better performance, or extending HiLo on more tasks such as speech recognition and video processing.

Limitations and societal impact. HiLo adopts a head splitting ratio to assign different numbers of heads into Hi-Fi and Lo-Fi. In our experiments, this ratio is determined by a grid search on ImageNet (*i.e.*, $\alpha = 0.9$). However, different tasks may have different importance on high and low frequencies. Thus, the optimal value of α is task-specific and needs to be set manually. Besides, our work potentially brings some negative societal impacts, such as the huge energy consumption and carbon emissions from large-scale training on GPU clusters.

References

[1] J. Ba, J. Kiros, and G. E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[2] B. Chen, P. Li, C. Li, B. Li, L. Bai, C. Lin, M. Sun, J. Yan, and W. Ouyang. Glit: Neural architecture search for global and local image transformer. In *ICCV*, 2021.

[3] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[4] M. Chen, H. Peng, J. Fu, and H. Ling. Autoformer: Searching transformers for visual recognition. In *ICCV*, 2021.

[5] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the frequency domain. In *KDD*, pages 1475–1484. ACM, 2016.

[6] Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan, and Z. Liu. Mobile-former: Bridging mobilenet and transformer. In *CVPR*, 2022.

[7] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[8] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *ICLR*, 2021.

[9] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 2021.

[10] J. W. Cooley, P. A. W. Lewis, and P. D. Welch. The fast fourier transform and its applications. *IEEE Transactions on Education*, 12(1):27–34, 1969.

[11] J. Cordonnier, A. Loukas, and M. Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020.

[12] G. Deng and L. Cahill. An adaptive gaussian filter for noise reduction and edge detection. In *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, pages 1615–1619 vol.3, 1993.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[14] M. Fritsche, S. Gu, and R. Timofte. Frequency separation for real-world super-resolution. *ICCVW*, Oct 2019.

[15] M.-H. Guo, C.-Z. Lu, Z.-N. Liu, M.-M. Cheng, and S.-M. Hu. Visual attention network. *arXiv preprint arXiv:2202.09741*, 2022.

[16] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[19] J. Huang, D. Guan, A. Xiao, and S. Lu. Rda: Robust domain adaptation via fourier adversarial attacking. In *ICCV*, pages 8988–8999, October 2021.

[20] M. A. Islam, S. Jia, and N. D. B. Bruce. How much position information do convolutional neural networks encode? In *ICLR*, 2020.

[21] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, pages 5156–5165. PMLR, 2020.

[22] A. Kirillov, R. B. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In *CVPR*, pages 6399–6408, 2019.

[23] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *ICCV*, pages 2999–3007, 2017.

[24] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV*, pages 740–755, 2014.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

[26] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. In *CVPR*, 2022.

[27] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018.

[28] S. Mehta and M. Rastegari. Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer. In *ICLR*, 2022.

[29] Z. Pan, B. Zhuang, H. He, J. Liu, and J. Cai. Less is more: Pay less attention in vision transformers. In *AAAI*, 2022.

[30] Z. Pan, B. Zhuang, J. Liu, H. He, and J. Cai. Scalable visual transformers with hierarchical pooling. In *ICCV*, 2021.

[31] N. Park and S. Kim. How do vision transformers work? In *ICLR*, 2022.

[32] H. Peng, N. Pappas, D. Yogatama, R. Schwartz, N. A. Smith, and L. Kong. Random feature attention. In *ICLR*, 2021.

[33] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, pages 211–252, 2015.

[35] S. Tang, J. Zhang, S. Zhu, and P. Tan. Quadtree attention for vision transformers. *ICLR*, 2022.

[36] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[38] E. Voigtman and J. D. Winefordner. Low-pass filters for signal averaging. *Review of Scientific Instruments*, 57(5):957–966, 1986.

[39] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[40] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[41] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pvtv2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):1–10, 2022.

[42] R. Wightman, H. Touvron, and H. Jégou. Resnet strikes back: An improved training procedure in timm. *CoRR*, abs/2110.00476, 2021.

[43] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, 2021.

[44] M. Xiao, S. Zheng, C. Liu, Y. Wang, D. He, G. Ke, J. Bian, Z. Lin, and T. Liu. Invertible image rescaling. In *ECCV*, volume 12346, pages 126–144. Springer, 2020.

[45] T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollár, and R. B. Girshick. Early convolutions help transformers see better. In *NeurIPS*, 2021.

[46] K. Xu, M. Qin, F. Sun, Y. Wang, Y. Chen, and F. Ren. Learning in the frequency domain. In *CVPR*, pages 1737–1746. Computer Vision Foundation / IEEE, 2020.

[47] J. Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao. Focal self-attention for local-global interactions in vision transformers. In *NeurIPS*, 2021.

[48] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu. Incorporating convolution designs into visual transformers. In *ICCV*, 2021.

[49] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

[50] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In *ICCV*, 2021.

[51] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ADE20K dataset. *IJCV*, pages 302–321, 2019.

[52] Y. Zhou, W. Deng, T. Tong, and Q. Gao. Guided frequency separation network for real-world super-resolution. In *CVPRW*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Code and pretrained models are included in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We use the same random seed as in recent works for fair comparison.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 5.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.
   (b) Did you mention the license of the assets? [No] The license of the public datasets can be found in their websites.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Code and pretrained models are included in the supplementary material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] We use public datasets (*e.g.*, ImageNet [34] and ADE20K [51]).
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] No such concerns as we are using widely adopted public datasets.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]