Multi-Scale Vision Longformer:
A New Vision Transformer for High-Resolution Image Encoding

Pengchuan Zhang$^\heartsuit$ Xiyang Dai$^{\heartsuit\dagger}$ Jianwei Yang$^{\heartsuit\dagger}$ Bin Xiao$^\heartsuit$ Lu Yuan$^\heartsuit$

Lei Zhang$^\heartsuit$ Jianfeng Gao$^\heartsuit$

Abstract

*This paper presents a new Vision Transformer (ViT)
architecture* Multi-Scale Vision Longformer*, which significantly enhances the ViT of [11] for encoding high-resolution images using two techniques. The first is the
multi-scale model structure, which provides image encodings at multiple scales with manageable computational
cost. The second is the attention mechanism of vision Longformer, which is a variant of Longformer [2], originally
developed for natural language processing, and achieves a
linear complexity w.r.t. the number of input tokens. A comprehensive empirical study shows that the new ViT significantly outperforms several strong baselines, including the
existing ViT models and their ResNet counterparts, and the
Pyramid Vision Transformer from a concurrent work [43],
on a range of vision tasks, including image classification,
object detection, and segmentation. The models and source
code used in this study will be released to public soon.*

1. Introduction

Vision Transformer (ViT) [11] has shown promising results on image classification tasks for its strong capability of long range context modeling. But its quadratic increase of both computational and memory complexity hinders its application on many vision tasks that require high-resolution feature maps computed on high-resolution images[1], like object detection [31, 23], segmentation [25, 5], and human pose estimation [45, 34]. Vision-language tasks, like VQA, image captioning, and image-text retrieval, also benefit from high-resolution feature maps [15, 49], which are extracted with pre-trained CNN models. Developing a vision Transformer that can process high-resolution feature maps is a critical step toward the goal of unifying the model architecture of vision and language modalities and improv-

ing multi-modal representation learning.

In this paper, we propose a new vision Transformer architecture *Multi-Scale Vision Longformer*, which significantly enhances the baseline ViT [11] for encoding high-resolution images using two techniques: (1) the multi-scale model structure, and (2) the attention mechanism of vision Longformer.

Models with multi-scale (pyramid, hierarchical) structure provide a comprehensive encoding of an image at multiple scales, while keeping the computation and memory complexity manageable. Deep convolutional networks are born with such multi-scale structure, which however is not true for the conventional ViT architecture. To obtain a multi-scale vision Transformer, we stack multiple (e.g., four) vision Transformers (ViT stages) sequentially. The first ViT stage operates on a high-resolution feature map but has a small hidden dimension. As we go to later ViT stages, the feature map resolution reduces while the hidden dimension increases. The resolution reduction is achieved by performing patching embedding at each ViT stage. In our experiments, we find that with the same number of model parameters and the same model FLOPs, the multi-scale ViT achieves a significantly better accuracy than the vanilla ViT on image classification task. The results show that the multi-scale structure not only improves the computation and memory efficiency, but also boosts the classification performance. The proposed multi-scale ViT has the same network structure as conventional (multi-scale) CNN models such as ResNet [13], and can serve as a replace-and-plug-in choice for almost all ResNet applications. In this paper, we demonstrate this plausible property in image classification, object detection and instance segmentation.

The multi-scale structure alone is not sufficient to scale up ViT to process high-resolution images and feature maps, due to the quadratic increase of the computation and memory complexity with respect to the number of tokens in the self-attention layers. Compared to natural language tasks where data is 1-D, this problem is more severe in vision tasks where the increase in complexity is quartic (fourth order) with the increase of image resolution. For example, the

---

$^\heartsuit$Microsoft Corporation   † indicates equal contributions.

[1]In this paper, encoding a high-resolution image means generating high-resolution feature maps for high-resolution images.

computational complexity of a $4\times$ higher resolution multi-head self attention (MSA) layer (hidden dimension reduced by 4, i.e., $4H \times 4W \times \frac{D}{4}$) equals to that of 64 layers in the original size (i.e., $H \times W \times D$). To address this challenge, we develop a 2-D version of Longformer[2], called *Vision Longformer*, to achieve a linear complexity w.r.t. the number of tokens (quadratic w.r.t. resolution). Our experiments show that compared to the baseline ViT, Vision Longformer shows no performance drop while significantly reduces the computational and memory cost in encoding images. The result indicates that the "local attention + global memory" structure in Vision Longformer is a desirable inductive bias for vision Transformers. We also compare Vision Longformer with other efficient attention mechanisms. The result again validates its superior performance on both image classification and object detection tasks.

The main contributions of this paper are two-fold: (1) We propose a new vision Transformer that uses the multi-scale model structure and the attention mechanism of 2-D Longformer for efficient high-resolution image encoding. (2) We perform a comprehensive empirical study to show that the proposed ViT significantly outperforms strong baselines, including previous ViT models, their ResNet counterparts, and a model from a concurrent work, on image classification, object detection and segmentation tasks.

2. Related Work

The Vision Transformer (ViT) [11] applies a standard Transformer, originally developed for natural language processing (NLP), for image encoding by treating an image as a word sequence, i.e., splitting an image into patches (words) and using the linear embeddings of these patches as an input sequence. ViT has shown to outperform convolution neural network (CNN) models such as the ResNet [13], achieving state-of-the-art performance on multiple image classification benchmarks, where training data is sufficient. DeiT [40] is another computer vision model that leverages Transformer. It uses a teacher-student strategy specific to Transformers to improve data efficiency in training. Thus, compared to ViT, it requires much less training data and computing resources to produce state-of-the-art image classification results. In addition to image classification, Transformers have also been applied to other compute vision tasks, including object detection [3, 54, 50, 10], segmentation [41, 44], image enhancement [4, 46], image generation [28, 6], video processing [48, 53], and vision-language tasks [26, 35, 7, 33, 21, 20, 52, 22].

Developing an efficient attention mechanism for high-resolution image encoding is the focus of this work. Our model is inspired by the efficient attention mechanisms developed for Transformers, most of which are for NLP tasks. These mechanisms can be grouped into four categories. The first is the sparse attention mechanism, including content-independent sparsity [28, 8, 29, 14] and content-dependent sparsity [17, 32, 37, 51]. Axial Transformer [14] and Image Transformer [28] are among few sparsity-based efficient attentions that are developed for image generation. The second is the memory-based mechanism, including Compressive Transformers [30] and Set Transformer [19]. These models use some extra global tokens as static memory and allow all the other tokens to attend only to those global tokens. The third is the low-rank based mechanism. For example the Linformer [42] projects the input key-value pairs into a smaller chunk, and performs cross-attention between the queries and the projected key-value pairs. The fourth is the (generalized) kernel-based mechanism, including Performer[9] and Linear Transformers[16]. Many models utilize hybrid attention mechanisms. For example, Longformer[2], BigBird[47] and ETC[1] combine the sparsity and memory mechanisms; Synthesizers[36] combines the sparsity and low-rank mechanisms. Readers may refer to [39] and [38] for a comprehensive survey and benchmarks, respectively.

In this paper, we developed a 2-D version of Longformer[2], called Vision Longformer, which utilizes both the sparsity and memory mechanisms. Its conv-like sparsity mechanism is conceptually similar to the sparsity mechanism used in the Image Transformer[28].

The multi-scale vision Transformer architecture is another technique we use in our proposed high-resolution vision Longformer. The hierarchical Transformers [27] for NLP contain two stages, with the first stage processing overlapping segments and the second stage using the embeddings of the CLS tokens from all segments as input. In our proposed Vision Longformer, size reduction is performed by the patch embedding at the beginning of each stage, by merging all tokens in a patch from previous stage into a single token at the current stage. We typically use 4 stages for our model since we have empirically verified that using 4 stages is better than using 2 or 3 stages, especially for object detection tasks. Informer[51] takes a similar stacked multi-stage approach to encoding long sequences, where the size reduction between stages is achieved by max-pooling.

Pyramid Vision Transformer (PVT) [43] is a concurrent work of ours. Both PVT and our model use a multi-scale architecture where multiple (slightly modified) ViTs are stacked. The authors of PVT propose the spatial-reduction attention (SRA) to alleviate the cost increase in self-attention layers. However, the computation and memory complexity of PVT still increases quartically w.r.t. resolution (with a much smaller constant). Therefore, it becomes prohibitively expensive for PVT to encode images with resolution above 800. We have compared our method to PVT and validated its superior performance
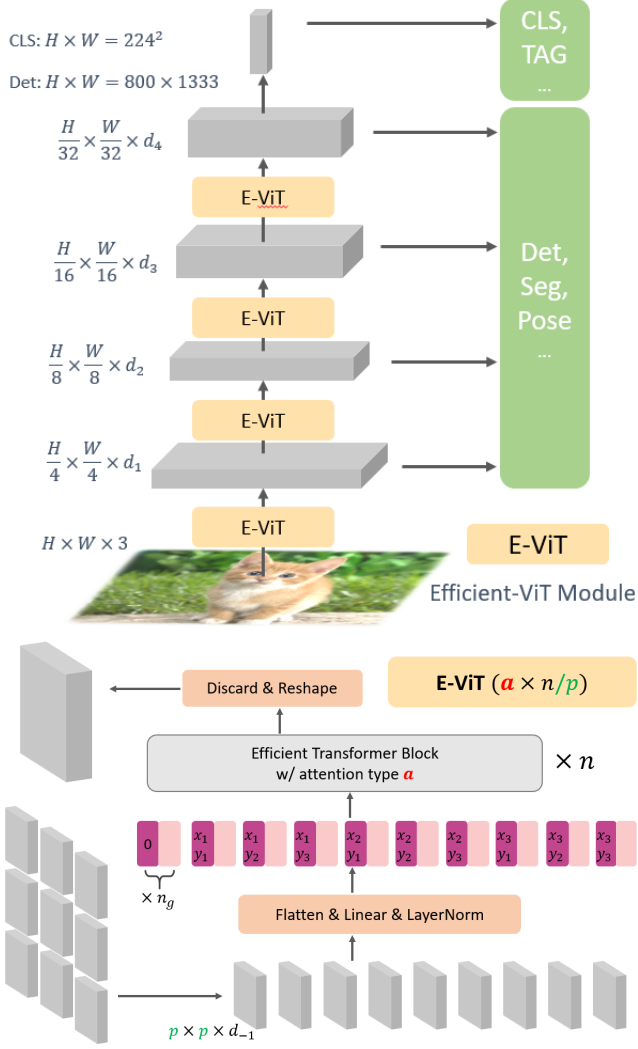
Figure 1. A Multi-scale vision Transformers (bottom) by stacking 4 E-ViT modules (Top). An E-ViT ($a \times n$ / p) module is a ViT encoder with an efficient attention mechanism a, n efficient transformer blocks, input patch size p. We add a LayerNorm after the patch embedding and change the original 1-D positional embedding to a (x, y)-type 2-D embedding. We add n_g extra global tokens, as a form of global memory, and simply throw them away when going to the next stage.

3. Multi-Scale Stacked Vision Transformers

3.1. Multi-Scale Model Architecture

Efficient ViT (E-ViT). As shown in Figure 1 (Bottom), we improve the encoding efficiency of vision Transformer by making the following modifications to the vanilla ViT. The modified ViT is referred to as Efficient ViT (E-ViT).

1. We add a Layer Normalization (LayerNorm) after the patch embedding.

2. We define a number of *global tokens*, including the

CLS token. Correspondingly, the tokens associated with image and feature patches are referred to as *local tokens* afterwards.

3. We change the 1-D positional embedding of the local tokens to a 2-D positional embedding by separately encoding x and y coordinates, and concatenating them.

4. We replace the vanilla full self-attention with an efficient attention mechanism, denoted by a, which will be described in detail in Sections 3.2 and 3.3.

Except for attention a, E-ViT has the following architecture parameters inherited from the vanilla ViT : input patch size p, number of attention blocks n, hidden dimension d and number of heads h, denoted as E-ViT($a \times n/p$; h, d, n_g). Using the full attention mechanism (i.e., a = full) and one global token (i.e., the CLS token with $n_g = 1$), the deficient E-ViT(full $\times 12/16$; $h, d, 1$) models still achieve better image classification performance on ImageNet than the baseline ViT for both tiny ($h = 3, d = 192$) and small ($h = 6, d = 384$) model sizes, as shown in Table 2. The performance gain is attributed to the added LayerNorm, as we show in the Supplementary. The 2-D positional embedding is used mainly for saving parameters for generating high-resolution feature maps.

Mathematically, an E-ViT($a \times n/p$; h, d, n_g) encoding module can be written as:

$$z_0 = [x_g^1; \ldots; x_g^{n_g}; LN(x_p^1 E); \ldots; LN(x_p^{n_l} E)] + E_{ops},$$
$$\tag{1}$$

$$z'_k = MSA_a(LN(z_{k-1})) + z_{k-1}, \quad k = 1, .., n \tag{2}$$

$$z_k = MLP(LN(z'_{k-1})) + z'_{k-1}, \quad k = 1, .., n, \tag{3}$$

where LN is the added Layer Normalization after the patch embedding E, $E_{ops} \in \mathbb{R}^{(n_l+n_g) \times d}$ contains the 2-D positional embedding of n_l local tokens and the 1-D positional embedding of n_g global tokens, MSA_a is the multi-head self-attention with attention type a, and MLP is the feedforward block in a standard Transformer.

Stack multiple E-ViT modules as multi-scale vision Transformers. As illustrated in Figure 1 (Top), a multi-scale Vision Transformer is built by stacking multiple E-ViT modules (or stages). In what follows, we describe several design choices we have made when building the multi-scale ViT.

What are the patch size and hidden dimension at each stage? As required in object detection and human pose estimation, for models with 4-scale feature maps, the first feature map needs to down-sample the image by 4 and thus stage 1 can be written as E-ViT($a_1 \times n_1/4$; $h_1, d_1, n_{g,1}$). We typically use only one attention block, i.e., $n_1 = 1$. The first stage generates the highest-resolution feature map, which consumes lots of memory. We also construct several

3-stage models, whose first stage patch size is 8. For later stages, the patch sizes are set to 2, which downsizes the feature map resolution by 2. Following the practice in ResNet, we increase the hidden dimension twice when downsizing the feature map resolution by 2. We list a few representative model configurations in Table 1. Different attention types (a) have different choices of number of global tokens n_g. But they share the same model configurations. Thus we do not specify a and n_g in Table 1. Readers refer to the Supplementary for the complete list of model configurations used in this paper,

Size	Stage1 n,p,h,d	Stage2 n,p,h,d	Stage3 n,p,h,d	Stage4 n,p,h,d
Tiny	1,4,1,48	1,2,3,96	9,2,3,192	1,2,6,384
Small	1,4,3,96	2,2,3,192	8,2,6,384	1,2,12,768
Medium-D	1,4,3,96	4,2,3,192	16,2,6,384	1,2,12,768
Medium-W	1,4,3,192	2,2,6,384	8,2,8,512	1,2,12,768
Base-D	1,4,3,96	8,2,3,192	24,2,6,384	1,2,12,768
Base-W	1,4,3,192	2,2,6,384	8,2,12,768	1,2,16,1024
Tiny-3stage	2,8,3,96		9,2,3,192	1,2,6,384
Small-3stage	2,8,3,192		9,2,6,384	1,2,12,768

Table 1. Model architecture for multi-scale stacked ViTs. Architecture parameters for each E-ViT stageE-ViT($a \times n/p$; h, d): number of attention blocks n, input patch size p, number of heads h and hidden dimension d. See the meaning of these parameters in Figure 1 (Bottom).

How to connect global tokens between consecutive stages? The choice varies at different stages and among different tasks. For the tasks in this paper, e.g., classification, object detection, instance segmentation, we simply discard the global tokens and only reshape the local tokens as the input for next stage. In this choice, global tokens only plays a role of an efficient way to globally communicate between distant local tokens, or can be viewed as a form of global memory. These global tokens are useful in vision-language tasks, in which the text tokens serve as the global tokens and will be shared across stages.

Should we use the average-pooled layer-normed features or the LayerNormed `CLS` token's feature for image classification? The choice makes no difference for flat models. But the average-pooled feature performs better than the `CLS` feature for multi-scale models, especially for the multi-scale models with only one attention block in the last stage as shown in Table 1. Readers refer to the Supplementary for an ablation study.

As reported in Table 2, the multi-scale models perform better than the flat models even in low-resolution classification problems. This shows the importance of multi-scale structure on classification tasks. However, the full self-attention mechanism suffers from the quartic computation/memory complexity w.r.t. the resolution of feature

maps, as shown in Table 2. Thus, it is impossible to train 4-stage multi-scale ViTs with full attention using the same setting (batch size and hardware) used for DeiT training.

Model	#Params (M)	FLOPs (G)	Memory (M)	Top-1 (%)
Ti-DeiT / 16 [40]	5.7	1.3	33.4	72.2
Ti-E-ViT(full/16)	5.7	1.3	33.4	73.2/73.1
Ti-Full-1,10,1	7.12	1.35	45.8	75.9
Ti-Full-2,9,1	6.78	1.45	60.6	75.8
Ti-Full-1,1,9,1	6.71	2.29	155.0	76.1
Ti-Full-1,2,8,1	6.37	2.39	170.5	75.6
Ti-ViL-1,10,1	7.12	1.27	38.3	75.6±0.23
Ti-ViL-2,9,1	6.78	1.29	45.5	75.9±0.08
Ti-ViL-1,1,9,1	6.71	1.33	52.7	76.2±0.12
Ti-ViL-1,2,8,1	6.37	1.35	60.0	76.0±0.10
S-DeiT / 16 [40]	22.1	4.6	67.1	79.9
S-E-ViT(full/16)	22.1	4.6	67.1	80.4/80.7
S-Full-1,10,1	27.58	4.84	78.5	81.7
S-Full-2,9,1	26.25	5.05	93.8	81.7
S-Full-1,1,9,1	25.96	6.74	472.9	–
S-Full-1,2,8,1	24.63	6.95	488.3	–
S-ViL-1,10,1	27.58	4.67	73.0	81.6
S-ViL-2,9,1	26.25	4.71	81.4	81.8
S-ViL-1,1,9,1	25.96	4.82	108.5	81.8
S-ViL-1,2,8,1	24.63	4.86	116.8	82.0

Table 2. Flat vs Multi-scale Models with full self-attention: Number of paramers, FLOPS, training time, memory per image (with Pytorch Automatic Mixed Precision enabled), and ImageNet accuracy with image size 224. "Ti-Full-2,9,1" stands for a tiny-scale 3-stage multiscale ViT with a = full attention and with 2,9,1 number of attention blocks in each stage, respectively. Similarly, small models start with "S-". Since all our multi-scale models use average-pooled feature from the last stage for classification, we report Top-1 accuracy of E-ViT(full/16) both with the CLS feature (first) and with the average-pooled feature (second). The multi-scale models consistently outperform the flat models, but the memory usage of full attention quickly blows up when only one high-resolution block is introduced. The Vision Longformer ("ViL-") saves FLOPs and memory, without performance drop.

3.2. Vision Longformer: An Efficient Attention Mechanism

We propose to use the "local attention + global memory" efficient mechanism, as illustrated in Figure 2 (Left), to reduce the computational and memory cost when using E-ViT to generate high-resolution feature maps. The 2-D Vision Longformer is an extension of the 1-D Longformer [2] originally developed for NLP tasks. We add n_g global tokens (including the CLS token) that are allowed to attend to all other tokens, serving as global memory. We also allow local tokens to only attend to global tokens and their local 2-D neighbors within a window size, and thus limit the increase of the attention cost to be linear w.r.t. number of input tokens. In summary, there are four components in this "local
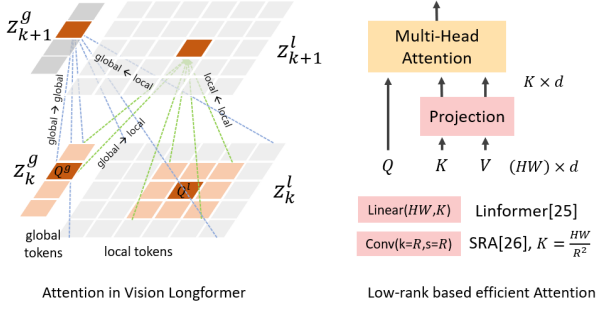
Figure 2. Left: the Vision Longformer attention mechanism. Right: the Low-rank based attention mechanism. Without "local→local" attentions in Vision Longformer, we get the Global Former. With a linear layer as the projection, we get Linformer[42]. With a conv layer with equal kernel size and stride, we get Spatial Reduction Attention (SRA)[43].

attention + global memory" mechanism, namely global-to-global, local-to-global, global-to-local, and local-to-local, as illustrated in Figure 2 (Left). In Equation 2, a Multi-head Self-Attention (MSA) block with the vision Longformer attention mechanism is denoted as MSA_{ViL}, i.e., $a = \text{ViL}$ in Equation 2.

Theoretical complexity. Given the numbers of global and local tokens, denoted by n_g and n_l respectively, and local attention window size w, the memory complexity of the MSA_{ViL} block is $\mathcal{O}(n_g(n_g + n_l) + n_l w^2)$. Although [2] points out that separating the attention parameters for global and local tokens is useful, we do not observe any gain in our experiments and thus simply let them share the same set of attention parameters. We empirically set the window size w to 15 for all E-ViT stages, which makes our model comparable with the global attention window size 14 of ViT/16 acted on 224×224 images. With such a window size, only attentions in the first two stages (in 4-stage multi-scale ViTs) are local. The attentions in the later two stages are equivalent[2] to full attention. In our experiments, we find that it is sufficient to use only one global token ($n_g = 1$) for ImageNet classification problems. So, the effective memory complexity of the MSA_{ViL} block is $\mathcal{O}((15^2 + 1)n_l)$, which is linear w.r.t. the number of tokens.

Results in Table 2 show that in comparison with the full attention models, the proposed multi-scale vision Longformer achieves a similar or slightly better performance, while saving significant memory and computation cost. It indicates that the "local attention + global memory" mechanism is a good inductive bias for vision Transformers. Although the savings are negligible for feature maps with resolution 28×28 (i.e., the feature maps in the first stage of 3-stage multi-scale), the savings are much more significant for feature maps with resolution 56×56 (i.e., the feature

maps in the first stage of a 4-stage multi-scale model). It turns out that the higher resolution the feature map is, the closer the practical memory usage is to the theoretical complexity.

Three implementations of Vision Longformer. Vision Longformer is conceptually similar to conv-like local attention. We have implemented vision Longformer in three ways: (1) using Pytorch's unfold function (nn.unfold or tensor.unfold), (2) using a customized CUDA kernel and (3) using a sliding chunk approach. The unfold implementation is simple but very slow, i.e., 24 times slower than full attention on $40 \times 40 \times 768$ feature map. The implementation using the customized CUDA kernel is about 20% faster than the full attention in the same setting, while achieving the theoretical memory complexity. The sliding-chunk approach is the fastest, which is 60% faster than the full attention with a cost of consuming 20% more memory than the theoretical complexity. Readers refer to the Supplementary for a detailed comparison of these implementations.

3.3. Other Efficient Attention Mechanisms

We compare Vision Longformer with the following alternative choices of efficient attention methods. We put detailed descriptions of these methods and their experimental setup in the Supplementary.

Pure global memory ($a = $ global). In Vision Longformer, see Figure 2 (Left), if we remove the local-to-local attention, then we obtain the pure global memory attention mechanism (called Global Attention hereafter). Its memory complexity is $\mathcal{O}(n_g(n_g + n_l))$, which is also linear w.r.t. n_l. However, for this pure global memory attention, n_g has to be much larger than 1. We gradually increase n_g (by 2 each time) and its performance gets nearly saturated at 128. Therefore, $n_g = 128$ is the default for this Global attention.

Linformer[42] ($a = $ LIN) projects the $n_l \times d$ dimensional keys and values to $K \times d$ dimensions using additional projection layers, where $K \ll n_l$. Then the n_l queries only attend to these projected K key-value pairs. The memory complexity of Linformer is $\mathcal{O}(Kn_l)$. We gradually increase K (by 2 each time) and its performance gets nearly saturated at 256. Therefore, $K = 256$ is the default for this Linformer attention, which turns out to be the same with the recommended value. Notice that Linformer's projection layer (of dimension $K \times n_l$) is specific to the current n_l, and cannot be transferred to higher-resolution tasks that have a different n_l.

Spatial Reduction Attention (SRA) [43] ($a = $ SRA) is similar to Linformer, but uses a convolution layer with kernel size R and stride R to project the key-value pairs, hence resulting in n_l/R^2 compressed key-value pairs. Therefore, The memory complexity of SRA is $\mathcal{O}(n_l^2/R^2)$, which is still quadratic w.r.t. n_l but with a much smaller constant $1/R^2$. When transferring the ImageNet-pretrained SRA-

---

[2]Equivalent in our sliding chunks implementation, which is our default choice.

models to high-resolution tasks, SRA still suffers from the quartic computation/memory blow-up w.r.t. the feature map resolution. Pyramid Vision Transformer [43] uses this SRA to build multi-scale vision transformer backbones, with different spatial reduction ratios ($R_1 = 8, R_2 = 4, R_3 = 2, R_4 = 1$) for each stage. With this PVT's setting, the key and value feature maps at all stages are essentially with resolution $H/32 \times W/32$.

Performer [9] ($a = $ **performer**) uses random kernels to approximate the Softmax computation in MSA, and achieves a linear computation/memory complexity with respect to n_l and the number of random features. We use the default 256 orthogonal random features (OR) for Performer, and provide other details in the Supplementary.

Multi-scale Models	Tiny-4stage / 4		Small-4stage / 4		Trans 2Det
	1,1,9,1	1,2,8,1	1,1,9,1	1,2,8,1	
Full	76.06	75.60	OOM	OOM	–
ViL	76.18	75.98	81.79	81.99	✓
Global	71.52	72.00	79.17	78.97	✓
Linformer [42]	74.71	74.74	81.19	80.98	✗
SRA/64[43]	69.08	68.78	76.35	76.37	✓
SRA/32[43]	73.22	73.2	79.96	79.9	–
Performer	71.12	73.09	78.81	78.72	✓
Par-Global	75.32	75.4	81.6	81.45	–
Par-Linformer	75.56	75.33	81.66	81.79	✗
Par-SRA/32	75.2	75.26	81.62	81.61	–
Par-Performer	75.34	75.93	81.72	81.72	–

Table 3. Overall comparison of different attention mechanisms on ImageNet classification top-1 accuracy (%), with input size 224. Tiny-4stage / 4 means that the model has a comparable size with DeiT-Tiny, has 4 stages and uses patch size 4x4 in the initial pixel space. "1,2,8,1" are the numbers of attention blocks in each stage. "Par-xformer" indicates multi-scale ViTs with multiple attention types: the first two stages utilize the "xformer" sparse attention and the last two stages still use full attention. In the "Trans2Det" columns, ✓ indicates that the ImageNet-pre-trained model can be used to initialize detection backbones, ✗ means not. − means that it can be transferred, but the corresponding detection models consumes prohibitively large memory due to the need of using high resolution feature maps. SRA/32 downsizes key/value feature maps with the same schedule in PVT[43], while SRA/64 downsizes more aggressively to make the memory manageable for downstream high-resolution tasks.

Compare vision longformer with other attention mechanisms. On ImageNet classification tasks in Table 3, all efficient attention mechanisms above show a large performance gap from Vision Longformer. Linformer performs very competitively. Global attention and Performer have a similar performance with the DeiT model (72.2 for tiny and 79.8 for small). We use spatial reduction ratios $16, 8, 4, 2$ from stage1 to stage4 for the multi-scale SRA model, which is different from the reduction ratios $8, 4, 2, 1$ in PVT [43].

This more aggressive spatial reduction makes the classification performance worse in Table 3, but makes the memory cost manageable when transfer to detection tasks for input image size 8000×1333. For a more complete comparison of these models, including model parameters, FLOPs and memory usage, please refer to the Supplementary.

Why is Longformer better? One possible reason is that the conv-like sparsity is a good inductive bias for vision transformers, compared with other attention mechanisms. This is supported by the visualization of the attention maps from pretrained DeiT models [40]. Another explanation is that Vision Longformer keeps the key and value feature maps high resolution. However, low resolution-based attention mechanims like Linformer and SRA and pure global attention lose the high-resolution information in the key and value feature maps. We will conduct deeper studies to better understand the effects of different efficient attention mechanisms for vision Transformers.

Mixed attention mechanisms (Partial -former) for classification tasks. For classification tasks with 224×224 image size as input, the feature map size at Stage3 in multi-scale ViTs is 14×14. This is the same as the feature map size in ViT and DeiT, which best suits for full attention. A natural choice is to use efficient attention in the first two stages (with high-resolution feature map but with small number of blocks) and to use full attention in the last two stages. Multi-scale ViTs with this mixed attention mechanisms are called "Parital X-former". We also report these Partial X-formers' performance in Table 3. All these Partial X-formers perform well on ImageNet classification, with very little (even no) gap between Full Attention and Vision Longformer. These Partial X-forms achieve very good accuracy-efficiency performance for low-resolution classification tasks. We do not have "Partial ViL" for classification because ViL's window size is 15, and thus its attention mechanism in the last two stages is equivalent to the full attention.

3.4. Transfer to High-resolution Vision Tasks

Similar to the transfer-ability of ImageNet-pretrained CNN weights to downstream high-resolution tasks, such as object detection and segmentation, multi-scale vision Longformer pretrained on ImageNet can also be transferred to such high-resolution tasks, as we will show in Section 4.3.

However, Linformer is not transferable because the weights of the linear projection layer is specific to a resolution. The Partial X-formers and Multi-scale ViT with full attention are not transferable due to its prohibitively large memory usage after transferred to high-resolution tasks. In Section 4.4, we also show the superior performance of Vision Longformer over other attention mechanisms, on the object detection and segmentation tasks.

4. Experiments

In this section, we show the final performance of Multi-scale Vision Longformer (short for ViL) on ImageNet classification in Section 4.1 and downstream high-resolution detection tasks in Section 4.3.We mainly follow the DeiT training configuration for ImageNet classification training, and use the standard $\times 1$ training schedule with the "AdamW" optimizer for detection tasks. We refer to the Supplementary for detailed experimental settings.

4.1. ImageNet Classification

Following DeiT[40] and PVT[43], we build multi-scale ViLs with four different sizes, i.e., tiny, small, medium and base. We experiment two strategies to scale the small model to medium and base sizes: deeper or wider, resulting in Medium/Base-Deep/Wide, respectively.The detailed model configuration is specified in Table 1.

We train multi-scale ViLs purely on ImageNet, following the setting in DeiT [40]. In Table 4, we report our results and compare with ResNets[13], ViT [11] and DeiT [40] trained in the same setting. Our models at all scales outperform other models in the same scale by a large margin. Moreover, our small model (size comparable with ResNet50) even outperforms X101-64x4d/ViT-Base/Deit-Base/PVT-Large models. Our medium-size models further improves based on the ViL-Small model by a large margin.

Model	#Params (M)	GFLOPs	Top-1 (%)
R18	11.7	1.8	69.8
DeiT-Tiny/16[40]	5.7	1.3	72.2
PVT-Tiny[43]	13.2	1.9	75.1
ViL-Tiny	6.7	1.3	76.3
R50	25.6	4.1	78.5
DeiT-Small/16[40]	22.1	4.6	79.9
PVT-Small[43]	24.5	3.8	79.8
ViL-Small	24.6	4.9	82.0
R101	44.7	7.9	79.8
ViT-Small/16[11]	48.8	9.9	80.8
PVT-Medium[43]	44.2	6.7	81.2
ViL-Medium-W	39.8	10.8	82.9
ViL-Medium-D	39.7	8.7	83.3
X101-64x4d	83.5	15.6	81.5
ViT-Base/16[11]	86.6	17.6	77.9
DeiT-Base/16[40]	86.6	17.6	81.8
PVT-Large[43]	61.4	9.8	81.7
ViL-Base-W	79.0	17.5	81.9
ViL-Base-D	55.7	13.4	83.2

Table 4. Number of paramers, FLOPS and ImageNet accuracy. Trained on ImageNet-1K with image size 224. Our ViL models are highlighted with gray background.

4.2. ImageNet-21K pretraining and ImageNet-1K finetuning

When trained purely on ImageNet-1K, there is no further performance increase from ViL-Medium to ViL-Base, even a significant decrease for the ViL-Base-W model. This is consistent with the observation in ViT: large pure transformer based models can be trained well when training data is sufficient. For ViL-Base size models, it is insufficient to only trained on ImageNet. In our supplementary material, we report ViL's performance on ImageNet, when pretrained on ImageNet-21k and finetuned on ImageNet. Therefore, we conducted experiments that ViL-Medium/Base models are first pretrained on ImageNet-21k with image size 224^2 and finetuned on ImageNet-1K with image size 384^2. For ViT models on image size 224^2, there are in total 24×24 tokens with full attention. For ViLmodels on image size 224^2, we set the window sizes to be $(13, 17, 25, 25)$ from Stage1 to Stage4. Therefore, in the last two stages, the ViLmodels' attention is equivalent to full attention.

In Table 5, we can see that the performance gets boosted significantly after ImageNet-21K pretraining for all ViLmedium and base models. Especially for the ViL-Base-W model, its ImageNet Top-1 accuracy is improved from 81.86 to 86.14. We want to point out that the performance of ViL-Medium-D model has surpassed that of ViT-Base/16, ViT-Large/16 and BiT-152x4-M, in the ImageNet-21K pretraining setting. The performance of ViL-Base models are even better. This shows the superior performance and parameter efficiency of ViLmodels.

Model	#Params (M)	No IN-21K		After IN-21K	
		GFLOPs	Top-1	GFLOPs	Top-1
ViT-Base/16[11]	86.6	17.6	77.9	49.3	84.0
ViT-Large/16[11]	307	61.6	76.5	191.1	85.2
BiT-152x4-M[18]	928	182	81.3	837	85.4
ViL-Medium-W	39.8	10.79	82.9	33.6	84.9
ViL-Medium-D	39.7	8.7	83.3	28.4	85.6
ViL-Base-W	79.0	17.5	81.9	54.3	86.2
ViL-Base-D	55.7	13.4	83.2	43.7	86.0

Table 5. Trained purely on ImageNet-1K with image size 224 (No IN-21K). Pretained on ImageNet-21K with image size 224 and Finetuned on ImageNet-1K with image size 384 (After IN-21K), except BiT-M [18] fine-tuned with image size 480. Our ViL models are highlighted with gray background.

4.3. Detection Tasks

We apply our ViL to two representative object detection pipelines including RetinaNet [23] and Mask-RCNN [12]. We follow the conventional setting to use our vision longformer as the backbone to generate feature maps for both detection pipelines. Similar to [43], we extract the features from all four scales and then feed them to the detection

Backbone	#Params (M)	FLOPs (G)	RetinaNet 1x schedule						RetinaNet 3x + MS schedule					
			AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet18	21.3	190.33	31.8	49.6	33.6	16.3	34.3	43.2	35.4	53.9	37.6	19.5	38.2	46.8
PVT-Tiny[43]	23.0	n/a	36.7	56.9	38.9	22.6	38.8	50.0	39.4	59.8	42.0	25.5	42.0	52.1
ViL-Tiny	16.64	171.91	38.8	59.0	41.3	23.8	41.5	51.8	40.7	61.2	43.2	26.4	43.9	53.6
ResNet50	37.7	239.32	36.3	55.3	38.6	19.3	40.0	48.8	39.0	58.4	41.8	22.4	42.8	51.6
PVT-Small[43]	34.2	n/a	40.4	61.3	43.0	25.0	42.9	55.7	42.2	62.7	45.0	26.2	45.2	57.2
ViL-Small	35.68	252.21	41.6	62.5	44.1	24.9	44.6	56.2	42.9	63.8	45.6	27.8	46.4	56.3
ResNet101	56.7	319.07	38.5	57.8	41.2	21.4	42.6	51.1	40.9	60.1	44.0	23.7	45.0	53.8
ResNeXt101-32x4d	56.4	319.07	39.9	59.6	42.7	22.3	44.2	52.5	41.4	61.0	44.3	23.9	45.5	53.7
PVT-Medium[43]	53.9	n/a	41.9	63.1	44.3	25.0	44.9	57.6	43.2	63.8	46.1	27.3	46.3	58.9
ViL-Medium	50.77	338.93	42.9	64.0	45.4	27.0	46.1	57.2	43.7	64.6	46.4	27.9	47.1	56.9
ResNeXt101-64x4d	95.5	483.59	41.0	60.9	44.0	23.9	45.2	54.0	41.8	61.5	44.4	25.2	45.4	54.6
PVT-Large[43]	71.1	n/a	42.6	63.7	45.4	25.8	46.0	58.4	43.4	63.6	46.1	26.1	46.0	59.5
ViL-Base	66.74	443.0	44.3	65.5	47.1	28.9	47.9	58.3	44.7	65.5	47.6	29.9	48.0	58.1

Table 6. Object detection performance on the COCO val2017 with RetinaNet. The FLOPs (G) are measured at resolution 800×1333, and FLOPs for PVT architecture are not available. Our ViL models are pre-trained on ImageNet-1K and highlighted with gray background.

Backbone	#Params (M)	FLOPs (G)	Mask R-CNN 1x schedule						Mask R-CNN 3x + MS schedule					
			AP^b	AP_{50}^b	AP_{75}^b	AP^m	AP_{50}^m	AP_{75}^m	AP^b	AP_{50}^b	AP_{75}^b	AP^m	AP_{50}^m	AP_{75}^m
ResNet18	31.2	131.03	34.0	54.0	36.7	31.2	51.0	32.7	36.9	57.1	40.0	33.6	53.9	35.7
PVT-Tiny[43]	32.9	n/a	36.7	59.2	39.3	35.1	56.7	37.3	39.8	62.2	43.0	37.4	59.3	39.9
ViL-Tiny	26.9	94.3	38.7	61.1	41.3	36.2	58.3	38.2	41.2	63.0	44.7	37.9	59.8	40.6
ResNet50	44.2	180.0	38.0	58.6	41.4	34.4	55.1	36.7	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small[43]	44.1	n/a	40.4	62.9	43.8	37.8	60.1	40.3	43.0	65.3	46.9	39.9	62.5	42.8
ViL-Small	45.0	174.3	41.8	64.1	45.1	38.5	61.1	41.4	43.4	64.9	47.0	39.6	62.1	42.4
ResNet101	63.2	259.77	40.4	61.1	44.2	36.4	57.7	38.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d	62.8	259.77	41.9	62.5	45.9	37.5	59.4	40.2	44.0	64.4	48.0	39.2	61.4	41.9
PVT-Medium[43]	63.9	n/a	42.0	64.4	45.6	39.0	61.6	42.1	44.2	66.0	48.2	40.5	63.1	43.5
ViL-Medium	60.1	261.1	43.4	65.9	47.0	39.7	62.8	42.1	44.6	66.3	48.5	40.7	63.8	43.7
ResNeXt101-64x4d	101.9	424.29	42.8	63.8	47.3	38.4	60.6	41.3	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large[43]	81.0	n/a	42.9	65.0	46.6	39.5	61.9	42.5	44.5	66.0	48.3	40.7	63.4	43.7
ViL-Base	76.1	365.1	45.1	67.2	49.3	41.0	64.3	44.2	45.7	67.2	49.9	41.3	64.4	44.5

Table 7. Object detection and instance segmentation performance on the COCO val2017 with Mask R-CNN. The FLOPs (G) are measured at resolution 800×1333, and FLOPs for PVT architecture are not available. Our ViL models are pre-trained on ImageNet-1K and highlighted with gray background.

and/or instance segmentation head. To adapt the learned positional embedding to the higher image resolution in detection, we perform bilinear interpolation on it prior to the training. In our experiments, all models are evaluated on COCO dataset [24], with 118k images for training and 5k images for evaluation. We report the results for both $1\times$ and $3\times$+MS training schedules, and compare them with two backbone architectures: ResNet [13] and the most recent PVT [12].

In Table 6, we can find our proposed ViL achieves significantly better performance than the ResNet baselines. Compared with the concurrent PVT architecture, our model also outperforms it with fairly large margin. Particularly, our ViL-Tiny has 1.9 points improvement over PVT-Tiny while using much less number of parameters (16.6M *v.s.* 23.0M). Note that in all other settings (Small, Medium and Base), our model also has less or comparable number of parame-

ters than the counterparts. The similar trend is also observed with Mask R-CNN pipeline. As shown in Table 7, our ViL backbone surpasses ResNet baseline and PVT on both object detection and instance segmentation, with less or comparable number of parameters. Using our largest model ViL-Base with only 76.1M parameters, we can achieve 45.7 AP^b and significantly outperforms ResNeXt101-64x4d and PVT-Large that use more parameters. Beside the model size, our ViL backbones also consume less computational cost (GFLOPs) than the ResNet counterparts. These consistent and significant improvements with both RetinaNet and Mask R-CNN demonstrate the promise of our proposed ViL when using it as the image encoder for high-resolution dense object detection tasks.

4.4. Ablation Study for Detection Tasks

Here, we answer the question that whether different efficient attention mechanisms can be equally applied to the

dense prediction tasks like object detection and instance segmentation. Similar to Sec 4.4, we study SRA [43], Global Transformer and Performer and their corresponding partial version with Mask R-CNN pipeline (trained with the $1\times$ schedule). As we can see in Table 8, when all use the efficient attention mechanisms, ViL achieves much better performance than the other three mechanisms. Specifically, our ViL achieves 41.2 AP^b while the other three are all around 35.0 AP^b. This is reasonable because except for ViL, all other efficient mechanisms have relatively worse pretrained model on ImageNet and thus poor representations for the downstream tasks. When all using partial version of efficient attention mechanisms, we can observe the gaps between different mechanisms shrink to around 1.0 point while our ViL still outperform all others. For Performer particularly, we further convert the its pretrained partial version to fully efficient version to investigate its transferrability. As we can see, it has even worse performance than the pure Performer. These results indicate that not all efficient mechanisms can be equally transferred to high-resolution dense prediction tasks, and the proposed ViL is the only choice which retains the good performance across different settings.

Attention	#Params (M)	AP^b	AP^m	FLOPs (G)	Memory (G)
SRA/64 [43]	73.3	35.7	33.6	153.9	7.1
SRA/32 [43]	51.5	39.8	36.8	176.6	13.6
Par-SRA/32	46.8	41.1	38.1	159.0	22.6
Global	45.2	34.1	32.5	177.1	7.6
Par-Global	45.1	41.3	38.2	158.5	20.1
Performer	45.0	35.0	33.1	160.7	8.4
Performer from Par-Performer	45.0	32.6	31.2	160.7	8.4
Par-Performer	45.0	41.7	38.4	160.7	20.0
ViL	45.0	41.3	38.1	174.3	7.4
Par-ViL	45.0	42.6	39.3	164.4	19.5

Table 8. Comparing different efficient attention mechanisms for object detection with Mask R-CNN. All use small model for the experiments. We report the run-time memory cost (G) when training each model in the last column.

5. Conclusions

In this paper, we have presented a new Vision Transformer (ViT) architecture *Multi-Scale Vision Longformer* to address the computational and memory efficiency which prevents the vanilla ViT model from applying to vision tasks requiring high-resolution feature maps. We mainly developed two techniques: 1) a multi-scale model structure designed for Transformers to provide image encoding at multiple scales with manageable computational cost, and 2) an efficient 2-D attention mechanism of vision Longformer for achieving a linear complexity w.r.t. the number of input tokens. The architecture design and the efficient attention mechanism are validated with comprehensive ablation studies. Our experimental results show that the new ViT architecture effectively addresses the computational and memory efficiency problem and outperforms several strong baselines on image classification and object detection.

References

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. Etc: Encoding long and structured data in transformers. *arXiv preprint arXiv:2004.08483*, 2020. 2

[2] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 1, 2, 4, 5, 15

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 2

[4] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. *arXiv preprint arXiv:2012.00364*, 2020. 2

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 1

[6] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pre-training from pixels. In *International Conference on Machine Learning*, pages 1691–1703. PMLR, 2020. 2

[7] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*, 2019. 2

[8] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2

[9] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2020. 2, 6, 17

[10] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *arXiv preprint arXiv:2011.09094*, 2020. 2

[11] 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, 2020. 1, 2, 7, 12

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7, 8

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 7, 8

[14] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019. 2

[15] Huaizu Jiang, Ishan Misra, Marcus Rohrbach, Erik Learned-Miller, and Xinlei Chen. In defense of grid features for visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10267–10276, 2020. 1

[16] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. 2

[17] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. 2

[18] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6, 2019. 7

[19] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019. 2

[20] Gen Li, Nan Duan, Yuejian Fang, Daxin Jiang, and Ming Zhou. Unicoder-VL: A universal encoder for vision and language by cross-modal pre-training. *arXiv preprint arXiv:1908.06066*, 2019. 2

[21] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019. 2

[22] Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020. 2

[23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 7

[24] 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 *ECCV*, 2014. 8

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[26] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. ViL-BERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*, 2019. 2

[27] R. Pappagari, P. Zelasko, J. Villalba, Y. Carmiel, and N. Dehak. Hierarchical transformers for long document classification. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 838–844, 2019. 2

[28] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018. 2, 15, 16

[29] Jiezhong Qiu, Hao Ma, Omer Levy, Scott Wen-tau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. *arXiv preprint arXiv:1911.02972*, 2019. 2

[30] Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1

[32] Aurko Roy*, Mohammad Taghi Saffar*, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers, 2020. 2

[33] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. VL-BERT: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 2

[34] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 1

[35] Hao Tan and Mohit Bansal. LXMERT: Learning cross-modality encoder representations from transformers. *EMNLP*, 2019. 2

[36] Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Synthesizer: Rethinking self-attention in transformer models. *arXiv preprint arXiv:2005.00743*, 2020. 2

[37] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International Conference on Machine Learning*, pages 9438–9447. PMLR, 2020. 2

[38] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020. 2

[39] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020. 2

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 2, 4, 6, 7, 12, 13, 14

[41] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020. 2

[42] Sinong Wang, Belinda Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2, 5, 6, 14, 17

[43] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2, 5, 6, 7, 8, 9, 14, 17

[44] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020. 2

[45] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *Proceedings of the European conference on computer vision (ECCV)*, pages 466–481, 2018. 1

[46] Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo. Learning texture transformer network for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5800, 2020. 2

[47] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020. 2

[48] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision*, pages 528–543. Springer, 2020. 2

[49] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2021. 1

[50] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020. 2

[51] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv preprint arXiv:2012.07436*, 2020. 2

[52] Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. Unified vision-language pretraining for image captioning and VQA. *AAAI*, 2020. 2

[53] Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[54] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 2

A. Settings

A.1. Model configurations

We listed the model configuration of all models used in this paper in Table 9. We do not specify the attention mechanism here, because the model configuration is the same for all attention mechanisms and the attention-specific parameters are specified in Table 15.

Model	Stage1 n,p,h,d	Stage2 n,p,h,d	Stage3 n,p,h,d	Stage4 n,p,h,d
Tiny	1,4,1,48	1,2,3,96	9,2,3,192	1,2,6,384
Small	1,4,3,96	2,2,3,192	8,2,6,384	1,2,12,768
Medium-D	1,4,3,96	4,2,3,192	16,2,6,384	1,2,12,768
Medium-W	1,4,3,192	2,2,6,384	8,2,8,512	1,2,12,768
Base-D	1,4,3,96	8,2,3,192	24,2,6,384	1,2,12,768
Base-W	1,4,3,192	2,2,6,384	8,2,12,768	1,2,16,1024
Tiny 1-10-1	1,8,3,96		10,2,3,192	1,2,6,384
Tiny 2-9-1	2,8,3,96		9,2,3,192	1,2,6,384
Tiny 1-9-2	1,8,3,96		9,2,3,192	2,2,6,384
Tiny 2-8-2	2,8,3,96		8,2,3,192	2,2,6,384
Tiny 1-1-9-1	1,4,1,48	1,2,3,96	9,2,3,192	1,2,6,384
Tiny 1-2-8-1	1,4,1,48	2,2,3,96	8,2,3,192	1,2,6,384
Small 1-10-1	1,8,3,192		10,2,6,384	1,2,12,768
Small 2-9-1	2,8,3,192		9,2,6,384	1,2,12,768
Small 1-9-2	1,8,3,192		9,2,6,384	2,2,12,768
Small 2-8-2	2,8,3,192		8,2,6,384	2,2,12,768
Small 1-1-9-1	1,4,3,96	1,2,3,192	9,2,6,384	1,2,12,768
Small 1-2-8-1	1,4,3,96	2,2,3,192	8,2,6,384	1,2,12,768

Table 9. Model architecture for multi-scale stacked ViTs. Architecture parameters for each E-ViT module E-ViT($a \times n/p$; h, d): number of attention blocks n, input patch size p, number of heads h and hidden dimension d. See the meaning of these parameters in Figure 1 (Bottom).

A.2. Experimental settings

Table 10 summarizes our training setups for our different models.

For the ImageNet classification task, our setting mainly follow that in DeiT [40]. For example, we do not use dropout but use random path. We use all data augmentations in DeiT [40], except that we apply Repeated Augmentation only on Medium and Base models. When fine-tuning from a ImageNet-21K pretrained checkpoint, we mainly follow the practice of ViT [11], train on image size 384×384, use SGD with momentum 0.9, use no weight decay, and use only random cropping for data augmentation.

For COCO object detection/segmentation tasks, we follow the standard "1×" and "3 × +MS" schedules. We only change the optimizer from SGD to AdamW and search for good initial learning rate and weight decay. For the "1×" schedule, the input image scale is fixed to be $(800, 1333)$ for the min and max sizes, respectively. For the "3 × +MS"

schedule, the input image is randomly resized to have min size in $\{640, 672, 704, 736, 768, 800\}$. We found that there is obvious over-fitting in Training ViL-Medium and ViL-Base models on COCO, mainly because that these two models are relatively large but they are only pretrained on ImageNet. Therefore, we are taking the best checkpoint (one epoch per checkpoint) along the training trajectory to report the performance.

B. More experimental results

B.1. Ablation study on the architecture design of multi-scale vision Longformer

In this section, we present two ablation studies on the model architecture of multi-scale vision Longformer.

Ablation of the effects of LayerNorm and 2-D positional embedding in the patch embedding. In Table 2, we show that our flat model E-ViT(full $\times$ 12/16), which only differs from the standard ViT/DeiT model by an newly-added LayerNorm after the patch embedding and the 2-D positional embedding, has better performance than the standard ViT/DeiT model. In Table 11, we show that this better performance comes from the newly-added LayerNorm.

Feature from the CLS token or from average pooling? As shown in Table 12, for ViL models that has only one attention block in the last stage (ViL 1-2-8-1), the average pooled feature from all tokens works better than the feature of the CLS token. However, when there are more than 2 attention blocks in the last stage (ViL 1-1-8-2), the difference between these two features disappears. The ViL 1-1-8-2 model has better performance than the ViL 1-2-8-1 model because it has more trainable parameters.

B.2. A comprehensive comparison of different attention mechanisms on ImageNet classification

We compare different attention mechanisms with different model sizes and architectures in Table 13 and Table 14. In Table 13, we show their performance on ImageNet-1K classification problem, measured by Top-1 accuracy. In Table 14, we show their number of parameters and FLOPs. We would like to comment that FLOPs is just a theoretical estimation of computation complexity, and it may not fit well the space/time cost in practice.

C. Implementations and Efficiency of Vision Longformer In Practice

There is a trivial implementation of the conv-like sliding window attention, in which we compute the full quadratic attention and then mask out non-neighbor tokens. This approach suffers from the quadratic complexity w.r.t. number of tokens (quartic w.r.t. feature map size), and is impractical for real use, as shown by the blue curve in Figure 4. We

Model	Dataset	Epoch	Base Lr	LR decay	Weight decay	Batch size
MsViT-Tiny	ImageNet	300	1e-3	cosine	0.1	1024
MsViT-Small	ImageNet	300	1e-3	cosine	0.1	1024
MsViT-Meidum-W	ImageNet	300	8e-4	cosine	0.1	1024
MsViT-Meidum-D	ImageNet	300	8e-4	cosine	0.1	1024
MsViT-Base-W	ImageNet	150	4e-4	cosine	0.1	1024
MsViT-Base-D	ImageNet	150	8e-4	cosine	0.1	1024
MsViT-Meidum-W	ImageNet-21k	90	5e-4	cosine	0.1	1024
MsViT-Meidum-D	ImageNet-21k	90	5e-4	cosine	0.1	1024
MsViT-Base-W	ImageNet-21k	90	3e-4	cosine	0.1	1024
MsViT-Base-D	ImageNet-21k	90	5e-4	cosine	0.1	1024
MsViT-Meidum-W	ImageNet-384	10	[2, 4]*e-2	cosine	0.	512
MsViT-Meidum-D	ImageNet-384	10	[2, 4]*e-2	cosine	0.	512
MsViT-Base-W	ImageNet-384	10	[2, 4]*e-2	cosine	0.	512
MsViT-Base-D	ImageNet-384	10	[2, 4]*e-2	cosine	0.	512
Model	Dataset	iterations	Base Lr	LR decay	Weight decay	Batch size
MsViT-Tiny-1x	COCO	60k-80k-90k	1e-4	multi-step	0.1	16
MsViT-Small-1x	COCO	60k-80k-90k	1e-4	multi-step	0.1	16
MsViT-Meidum-D-1x	COCO	60k-80k-90k	1e-4	multi-step	0.1	16
MsViT-Base-D-1x	COCO	60k-80k-90k	8e-5	multi-step	0.1	16
MsViT-Tiny-3x+ms	COCO	180k-240k-270k	1e-4	multi-step	0.1	16
MsViT-Small-3x+ms	COCO	180k-240k-270k	1e-4	multi-step	0.1	16
MsViT-Meidum-D-3x+ms	COCO	180k-240k-270k	1e-4	multi-step	0.1	16
MsViT-Base-D-3x+ms	COCO	180k-240k-270k	8e-5	multi-step	0.1	16

Table 10. Hyperparameters for training. We use MsViT to represent the multi-scale vision transformers with different kinds of attention mechanisms, including our Vision Longformer (ViL). For the experiments trained on COCO, MsViT is combined with the Retinanet or Mask R-CNN. The training configs for Retinanet or Mask R-CNN are the same, and we still use MsViT for their unified short name. We do not apply gradient clipping for all ImageNet classification training and apply gradient clipping at global norm 1 for COCO object detection/segmentation. We use AdamW for all our experiments, except that we use SGD with momentum 0.9 for the ImageNet-384 fine-tuning experiments.

Model	Tiny		Small	
	CLS	Ave Pool	CLS	Ave Pool
DeiT/16[40]	72.2	-	79.8	-
+Layernorm	73.04	73.16	80.54	80.32
+2D Pos	73.21	73.09	80.44	80.75

Table 11. Ablation of the effects of LayerNorm and 2-D positional embedding in the patch embedding of the E-ViT module, with ImageNet Top-1 accuracy. The improvement over DeiT [40] comes from the added LayerNorm. The 2-D positional embedding is mainly for saving parameters for high-resolution feature maps. The column names of "CLS" and "Ave Pool" indicate how the image feature is obtained for the linear classification head.

Model	Tiny		Small	
	CLS	Ave Pool	CLS	Ave Pool
ViL 1-2-8-1	75.72	75.98	81.65	81.99
ViL 1-1-8-2	76.18	76.25	82.12	82.08

Table 12. For ViL models that has only one attention block in the last stage (ViL 1-2-8-1), the average pooled feature from all tokens works better than the feature of the CLS token. When there are more than 2 attention blocks in the last stage (ViL 1-1-8-2), the difference between these two features disappears.

only use it to verify the correctness of our other implementations.

We have implemented Vision Longformer in three ways:

1. Using Pytorch's unfold function. We have two sub-versions: one using nn.functional.unfold (denoted as "unfold/nn.F") and the other using tensor.unfold (denoted as "unfold/tensor"). As shown in Figure 4, the "unfold/tensor" version (red solid line) is more efficient both in time and memory than the "unfold/nn.F" version (red dotted line). However, both of them are even slower and use more memory than the full attention!

Flat Models	Tiny				Small			
DeiT / 16 [40]	72.2				79.8			
E-ViT(full $\times$ 12/16)	73.21 (CLS) / 73.09 (AVG)				80.44 (CLS) / 80.75 (AVG)			
Multi-scale Models	Tiny-3stage / 8		Tiny-4stage / 4		Small-3stage / 8		Small-4stage / 4	
	1-10-1	2-9-1	1-1-9-1	1-2-8-1	1-10-1	2-9-1	1-1-9-1	1-2-8-1
Full Attention	75.86	75.79	76.06	75.60	81.66	81.73	81.93*	81.91*
Vision Longformer	$75.63_{\pm0.23}$	$75.88_{\pm0.08}$	$76.18_{\pm0.12}$	$75.98_{\pm0.10}$	81.57	81.81	81.79	81.99
Linformer [42]	74.54	74.72	74.71	74.74	80.76	80.99	81.19	80.98
Partial Linformer	75.64	75.82	75.56	75.33	81.66	81.63	81.66	81.79
SRA/64 [43]	68.71	68.84	69.08	68.78	75.9	76.18	76.35	76.37
SRA/32 [43]	73.16	73.46	73.22	73.2	79.82	79.8	79.96	79.9
Partial SRA/32	75.17	75.8	75.2	75.26	81.63	81.59	81.62	81.61
Global	70.93	71.62	71.52	72.00	79.04	79.08	79.17	78.97
Partial Global	75.55	75.61	75.32	75.4	81.39	81.42	81.6	81.45
Performer	71.28	71.87	71.12	73.09	78.17	78.58	78.81	78.72
Partial Performer	75.65	75.74	75.34	75.93	81.59	81.86	81.72	81.72

Table 13. Overall comparison in ImageNet top-1 accuracy, with input size 224. Tiny-4stage / 4 means that the model has comparable size with DeiT-Tiny, has 4 stages and uses patch size 4x4 in the initial pixel space. 1-2-8-1 means that the model contains 4 stages, each stage has 1/2/8/1 MSA-FFN blocks, respectively. *Partial* means that the last two stages, which contain most of the attention blocks, still use full attention. Vision Longformer does not have *Partial* version because its window size is set as 15 (comparable with the ViT(DeiT)/16 feature map size 14), and its attention mechanism in the last two stages is equivalent to full attention. * indicates that the training batch size is 256 (with learning rate linearly scaled down), different from all other experiments with batch size 1024 in this table.

Flat Models	Tiny				Small			
DeiT / 16 [40]	5.7 (M) parameters, 1.3 GFLOPs				22.1 (M) parameters, 4.6 GFLOPs			
E-ViT(full $\times$ 12/16)	5.7 (M) parameters, 1.3 GFLOPs				22.1 (M) parameters, 4.6 GFLOPs			
Multi-scale Models	Tiny-3stage / 8		Tiny-4stage / 4		Small-3stage / 8		Small-4stage / 4	
	1-10-1	2-9-1	1-1-9-1	1-2-8-1	1-10-1	2-9-1	1-1-9-1	1-2-8-1
Full Attention	7.1, 1.35	6.8, 1.45	6.7, 2.29	6.4, 2.39	27.6, 4.84	26.3, 5.05	26.0, 6.74	24.6, 6.95
Vision Longformer	7.1, 1.27	6.8, 1.29	6.7, 1.33	6.4, 1.35	27.6, 4.67	26.3, 4.71	26.0, 4.82	24.6, 4.86
Linformer [42]	7.8, 1.57	7.7, 1.6	8.2, 1.69	8.0, 1.73	28.3, 5.27	27.1, 5.35	27.4, 5.55	26.3, 5.62
Partial Linformer	7.3, 1.31	7.2, 1.37	7.7, 1.46	7.6, 1.52	27.8, 4.76	26.7, 4.88	27.0, 5.08	25.8, 5.21
SRA/64 [43]	14.2, 0.99	13.9, 0.99	13.8, 1.0	13.5, 1.0	55.9, 3.92	54.6, 3.92	54.3, 3.97	52.9, 3.97
SRA/32 [43]	8.7, 1.09	8.4, 1.09	8.3, 1.1	8.0, 1.1	34.1, 4.23	32.7, 4.23	32.5, 4.28	31.1, 4.28
Partial SRA/32	7.3, 1.23	7.1, 1.22	7.0, 1.24	6.8, 1.22	28.2, 4.6	27.4, 4.56	27.1, 4.61	26.4, 4.57
Global	7.2, 1.69	6.9, 1.7	6.8, 1.75	6.5, 1.76	27.8, 6.65	26.4, 6.67	26.2, 6.76	24.9, 6.78
Partial Global	7.2, 1.6	6.9, 1.62	6.8, 1.68	6.5, 1.7	27.8, 6.07	26.4, 6.14	26.2, 6.23	24.9, 6.3
Performer	7.3, 1.81	7.0, 1.86	6.9, 1.99	6.6, 2.05	27.8, 5.75	26.5, 5.87	26.2, 6.14	24.8, 6.26
Partial Performer	7.1, 1.35	6.8, 1.45	6.7, 1.57	6.4, 1.67	27.6, 4.84	26.3, 5.04	26.0, 5.31	24.7, 5.52

Table 14. Overall comparison in number of parameters (M) and GFLOPs, with input size 224. Tiny-4stage / 4 means that the model has comparable size with DeiT-Tiny, has 4 stages and uses patch size 4x4 in the initial pixel space. 1-2-8-1 means that the model contains 4 stages, each stage has 1/2/8/1 MSA-FFN blocks, respectively. *Partial* means that the last two stages, which contain most of the attention blocks, still use full attention. Vision Longformer does not have *Partial* version because its window size is set as 15 (comparable with the ViT(DeiT)/16 feature map size 14), and its attention mechanism in the last two stages is equivalent to full attention.
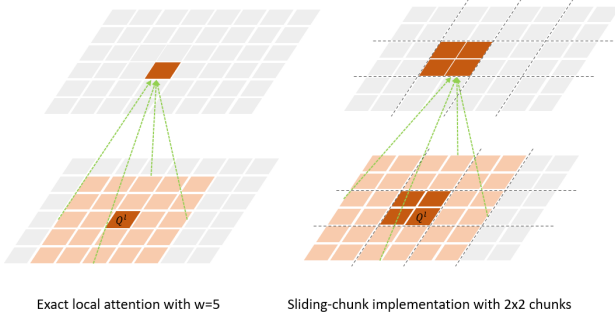
Figure 3. The sliding-chunk implementation of Vision Longformer. This implementation (Right) lets one token attends to more tokens than the exact conv-like local attention (Left). Our sliding-chunk implementation has the choice to be exactly the same with the conv-like local attention (Left), by masking out tokens that should not be attended to. For chunks on the boundaries, our implementation supports both zero padding and cyclic padding.

2. Using a customized CUDA kernel, denoted as "cuda_kernel". We make use of the TVM, like what has been done in Longformer [2], to write a customized CUDA kernel for vision longformer. As shown in Figure 4, the "cuda_kernel" (green line) achieves the theoretical optimal memory usage. Its time complexity is also reduced to linear w.r.t. number of tokens (quadratic w.r.t. feature map size). However, since it's not making use of the highly optimized matrix multiplication libraries in CUDA, it's speed is still slow in practice.

3. Using a sliding chunk approach, illustrated in Figure 3. For this sliding chunk approach, we have two subversions: one using Pytorch's autograd to compute backward step (denoted as "SCw/Autograd") and the other writing a customized torch.autograd.Function with hand-written backward function (denoted as "SCw/Handgrad"). Both sub versions of this sliding chunk approach are fully implemented with Pytorch functions and thus make use of highly optimized matrix multiplication libraries in CUDA. As shown in Figure 4, both of them are faster than the "cuda_kernel" implementation.

In the sliding chunk approach, to achieve a conv-like local attention mechanism with window size $2w + 1$, we split the feature map into chunks with size $w \times w$. Each chunk only attends to itself and its 8 neighbor chunks. The Pytorch Autograd will save 9 copies of the feature map (9 nodes in the computing graph) for automatic back-propogation, which is not time/memory efficient. The "SCw/Handgrad" version defines a customized torch.autograd.Function with hand-written backward function, which greatly saves the memory usage and also speeds up the algorithm, as shown
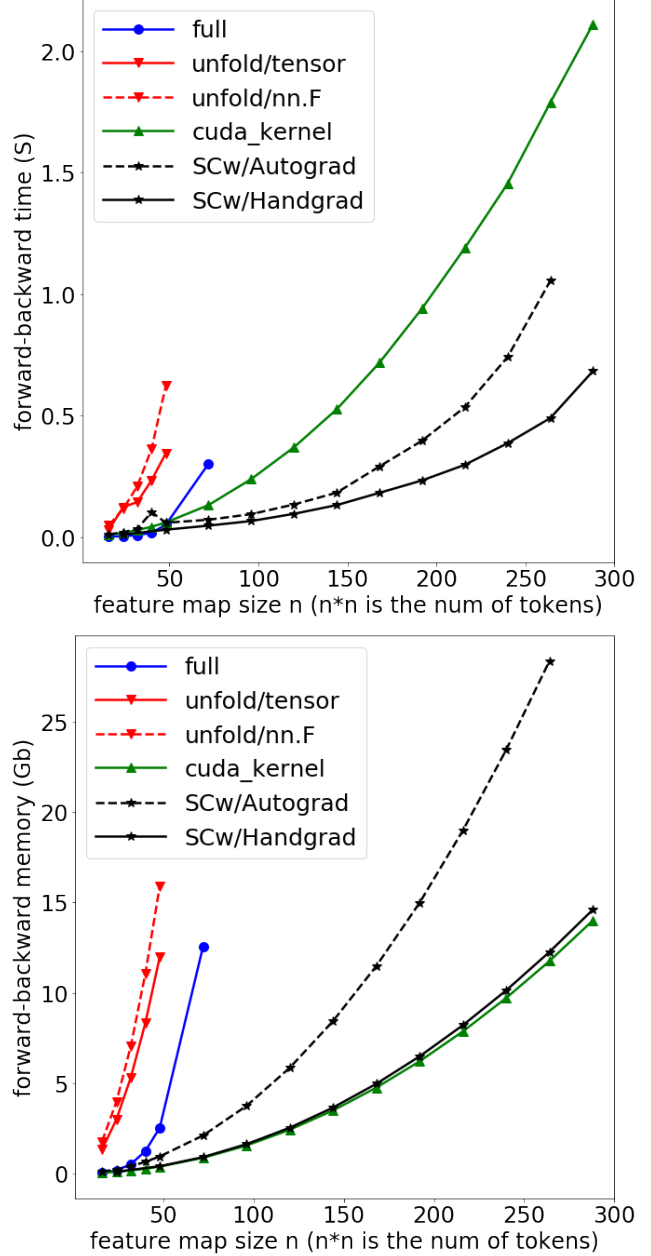

Figure 4. Compare of running time (including forward and backward) and memory usage of different implementations of the conv-like attention in vision longformer. All of these implementations shown in the figures are mathematically equivalent, doing the exact conv-like sliding window attention with window size 17.

in Figure 4. We would like to point out that the memory usage of the "SCw/Handgrad" version is nearly optimal (very close to that of the "cuda_kernel"). Similar speed-memory trade-off with different implementations of local attention mechanism has been observed in the 1-D Longformer [2], too; see Figure 1 in [2]. We would like to point out that Image Transformer [28] has an implementation of of 2-D
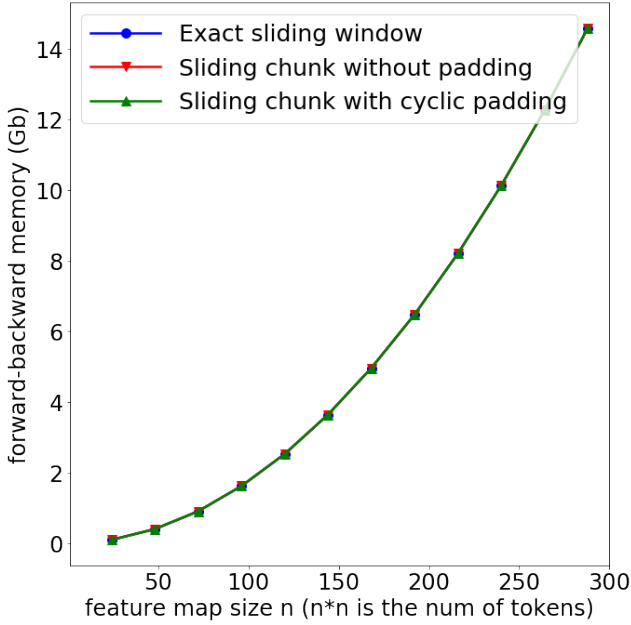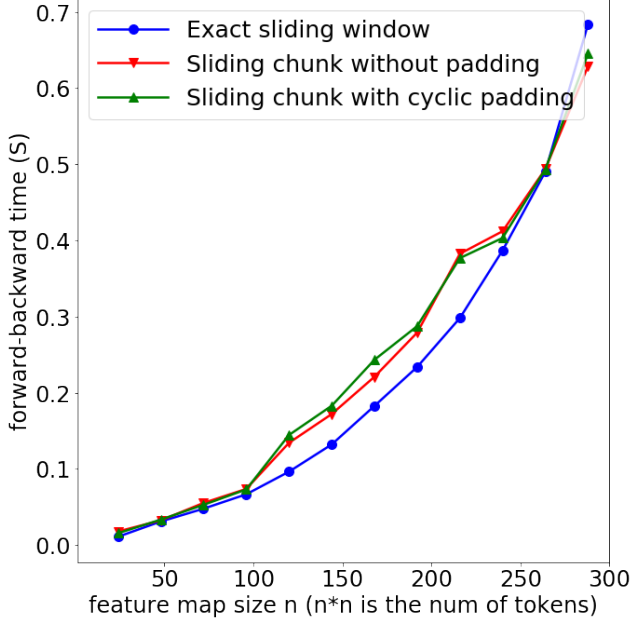
Figure 5. Compare of three modes of our "SCw/Handgrad" implementation of conv-like local attention: exact conv-like sliding window attention, sliding chunk attention without padding for boundary chunks, and sliding chunk attention with cyclic padding for boundary chunks. The are nearly the same in terms of running time (including forward and backward) and memory usage. The window size is 17 and thus chunk size is 8.

Figure 6. Running time (including forward and backward) and memory usage of our "SCw/Handgrad" implementation of conv-like local attention (sliding chunk attention without padding mode) with different window sizes. The speed is not sensitive to the window size for small window sizes (≤ 17) and the memory usage monotonically increases.

conv-like local attention mechanism, which is similar to our "SCw/Autograd" version. The Image Transformer [28] applies it to the image generation task.

This sliding-chunk implementation (Figure 3 Right) lets one token attends to more tokens than the exact conv-like local attention (Figure 3 Left). Our sliding-chunk imple-
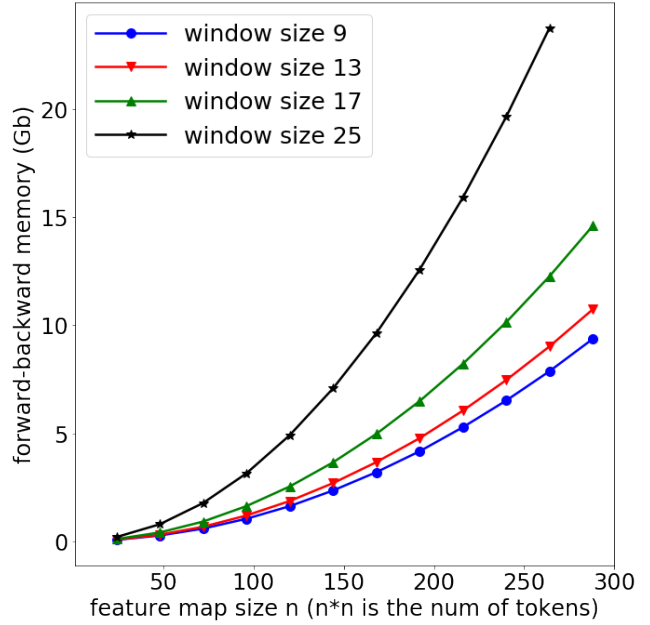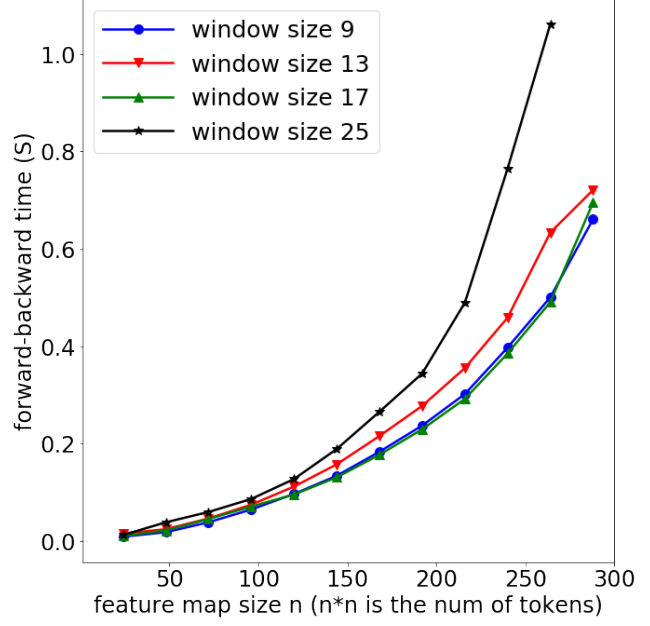
mentation has the choice to be

1. exactly the same with the conv-like local attention (Left), by masking out tokens that should not be attended to,

2. sliding chunk without padding, in which the chunks on the boundary have less chunks to attend to,

3. sliding chunk with cyclic padding, in which the chunks on the boundary still attend to 9 chunks with cyclic padded chunks.

. Since these three modes only differs by the attention masks to mask out invalid tokens, their speed and memory usage are nearly the same, as shown in Figure 5. For all the results reported in this paper, we use "sliding chunk without padding".

In Figure 6, we show the running time (including forward and backward) and memory usage of our "SCw/Handgrad" implementation of conv-like local attention (sliding chunk attention without padding mode) with different window sizes. We can see that the speed is not sensitive to the window size for small window sizes (≤ 17) and the memory usage monotonically increases.

Finally, both the "unfold/nn.F" and the "cuda_kernel" implementations support dilated conv-like attention. The customized CUDA kernel is even more flexible to support different dilations for different heads. The sliding-chunk implementation does not support this dilated conv-like attention. In this paper, we always use the sliding-chunk implementation due to its superior speed and nearly optimal memory complexity.

In Figure 4, 5 and 6, the evaluation is performed on a single multi-head self-attention module (MSA) with the conv-like local attention mechanism, instead of on the full multi-scale vision longformer. With this evaluation, we can clearly see the difference among different implementations of the conv-like local attention mechanism.

D. Other Efficient Attention Mechanisms utilized in this work

In this paper, we compare Vision Longformer with the following alternative choices of efficient attention methods.
Pure global memory (a = global). In Vision Longformer, see Figure 2 (Left), if we remove the local-to-local attention, then we obtain the pure global memory attention mechanism (called Global Attention hereafter). Its memory complexity is $\mathcal{O}(n_g(n_g + n_l))$, which is also linear w.r.t. n_l. However, for this pure global memory attention, n_g has to be much larger than 1. In practice, we set different numbers of global tokens for different stages, as shown in Table 15, with more global tokens in the first 2 stages and less in the last 2 stages. This setting makes the memory/computation complexity comparable with other attention mechanisms under the same model size.
Linformer[42] (a = LIN) projects the $n_l \times d$ dimensional keys and values to $K \times d$ dimensions using additional projection layers, where $K \ll n_l$. Then the n_l queries only attend to these projected K key-value pairs. The memory complexity of Linformer is $\mathcal{O}(Kn_l)$. We gradually increase K (by 2 each time) and its performance gets nearly satu-

rated at 256. Therefore, $K = 256$ is our choice for this Linformer attention, which turns out to be the same with the recommended value. Notice that Linformer's projection layer (of dimension $K \times n_l$) is specific to the current n_l, and cannot be transferred to higher-resolution tasks that have a different n_l. It is possible to transfer Linformer's weight by resizing feature maps of a different size to the original feature map size that Linformer is trained with and then applying the Linformer's projection. We do not explore this choice in this work.
Spatial Reduction Attention (SRA) [43] (a = SRA) is similar to Linformer, but uses a convolution layer with kernel size R and stride R to project the key-value pairs, hence resulting in n_l/R^2 compressed key-value pairs. Therefore, The memory complexity of SRA is $\mathcal{O}(n_l^2/R^2)$, which is still quadratic w.r.t. n_l but with a much smaller constant $1/R^2$. When transferring the ImageNet-pretrained SRA-models to high-resolution tasks, SRA still suffers from the quartic computation/memory blow-up w.r.t. the feature map resolution. Pyramid Vision Transformer [43] uses this SRA to build multi-scale vision transformer backbones, with different spatial reduction ratios ($R_1 = 8, R_2 = 4, R_3 = 2, R_4 = 1$) for each stage. With this PVT's setting, the key and value feature maps at all stages are essentially with resolution $\frac{H}{32} \times \frac{W}{32}$. This choice is able to scale up to image resolution 600×1000, but the memory usage is much larger than ResNet counterparts for 800×1333.

In this paper, we benchmarked the performance of SRA/32 with SR ratios $R_1 = 8, R_2 = 4, R_3 = 2, R_4 = 1$ (same as PVT [43]) and SRA/64 with SR ratios $R_1 = 16, R_2 = 8, R_3 = 4, R_4 = 2$ (two times more downsizing from that in PVT [43]), as shown in Table 15. The SRA/64 setting makes the memory usage comparable with other efficient attention mechanisms under the same model size, but introduces more parameters due to doubling the kernel size of the convolutional projection layer.

Performer [9] (a = performer) uses random kernel approximations to approximate the Softmax computation in MSA, and achieves a linear computation/memory complexity with respect to n_l and the number of random features K. We use the default $K = 256$ orthogonal random features (OR) for Performer. The memory/space complexity of performer is $\mathcal{O}(Kd + n_ld + Kn_l)$ while its computation/time complexity is $\mathcal{O}(Kn_ld)$. For the time complexity, we ignore the complexity of generating the orthogonal random features, which in practice cannot be ignored during training. We refer to Section B.3 in [9] for a detailed discussion of theoretical computation/memory complexity of Performer.

One important technique in training Performer is to redraw the random features during training. In our ImageNet classification training, we adopt a heuristic adaptive redrawing schedule: redraw every $1 + 5T$ iterations

in Epoch T ($T = 0, 1, ..., 299$). In our COCO object detection/segmentation training, the Performer is initialized from ImageNet pretrained checkpoint and thus there is no need to redraw very frequently in the initial training stage.Therefore, we redraw the random features every 1000 iterations in COCO object detection/segmentation training.

Model	Stage1	Stage2	Stage3	Stage4
	window size w			
ViL-3stage	15		15	15
ViL-4stage	15	15	15	15
ViL-4stage (384)	13	17	25	25
	# of global tokens n_g			
Global-3stage	256		64	16
Global-4stage	256	256	64	16
	Projection Dimension K			
Linformer-3stage	256		256	256
Linformer-4stage	256	256	256	256
	Spatial reduction ratio R			
SRA/64-3stage	8		4	2
SRA/64-4stage	16	8	4	2
SRA/32-3stage	4		2	1
SRA/32-4stage	8	4	2	1
	# of orthogonal random features K			
Performer-3stage	256		256	256
Performer-4stage	256	256	256	256

Table 15. Attention mechanism specific hyper-parameters. See Appendix D for the details of these attention mechanisms.