Focal Attention for Long-Range Interactions in Vision Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recently, Vision Transformers and its variants have shown promise on various vision tasks. The ability to capture short- and long-range visual dependencies through self-attention is arguably the secret sauce for its success but by nature brings difficulty due to quadratic computational overhead, especially for the high-resolution tasks (*e.g.*, object detection). A variety of works have attempted to address this by applying either coarse-grained global attention or fine-grained local attention. However, both paradigms discount the original power of transformer layers. In this paper, we present a focal self-attention mechanism, which simultaneously models the local and global interactions in a transformer layer and helps to capture both short and long-range visual dependencies efficiently *and* effectively. With focal self-attention, our built vision transformer models, called Focal Transformers achieve superior performance over the state-of-the-art methods in various standard benchmark settings on image classification and high-resolution object detection tasks. Specifically, a focal transformer model with a moderate size of 51.1M achieves 83.5 Top-1 accuracy on ImageNet classification at 224×224 resolution. Using our Focal Transformers as the backbones, we demonstrate consistent and substantial improvements over the current state-of-the-art Swin Transformers [35] on 6 different detection methods trained at both 1x and 3x schedule.

1 Introduction

Nowadays, Transformer [52] has become a prevalent model architecture in natural language processing (NLP) [18, 6]. In the light of its success in NLP, there is an increasing effort on adapting it to computer vision [39, 42]. Since its promise was first demonstrated in Vision Transformer (ViT) [19], we have witnessed a flourish of full-Transformer models for image classification [49, 55, 58, 35, 68, 51], object detection [9, 76, 71, 16] and semantic segmentation [53, 56]. Beyond these static image tasks, it has also been applied on various temporal tasks, such as action recognition [32, 70, 10], object tracking [14, 54], scene flow estimation [31], etc.

In Transformers, self-attention is the key component making it unique from the widely used convolutional neural networks (CNNs) [30]. It enables the global content-dependent interactions among different image regions for modeling both short- and long-range dependencies at each Transformer layer. Through the visualization of self-attention in DeiT-Tiny model[1], we indeed observe on the left side of Fig. 1 that it learns to attend local surroundings (like CNN) and the global contexts at the same time. Nevertheless, when it comes to high-resolution images for dense predictions such as object detection or segmentation, a global and fine-grained self-attention becomes non-trivial due to the quadratic computational cost with respect to the dimension of feature map. Recent works alternatively exploited either a coarse-grained global self-attention [55, 58] or a fine-grained local

[1] Pre-trained checkpoint downloaded from `https://github.com/facebookresearch/deit`.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

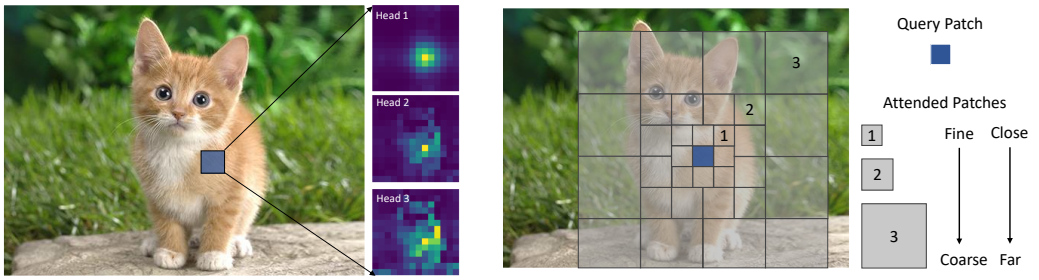


Figure 1: Left: Visualization of the attention maps of the three heads at the given query patch (blue) in the first layer of the DeiT-Tiny model. Right: An illustrative depiction of focal self-attention mechanism. Three granularity levels are used to compose the attention region for the blue query.

self-attention [35, 68, 51] to reduce the computational burden. However, both of them hurt the power of original self-attention after sacrificing one of its two merits, *i.e.*, the ability to simultaneously model short- and long-range visual dependencies.

In this paper, we present a new self-attention mechanism to capture both local and global interactions in Transformer layers. As discussed above, it is difficult to model the fine-grained global interactions for high-resolution inputs. We instead propose to perform self-attention at fine-grain locally whereas coarse-grain globally. As depicted in the right side of Fig. 1, for a query token in the feature map, it attends its closest surrounding at the finest grain as itself. When it goes to farther surroundings, we use summarized tokens to represent the coarser regions. Depending on how many levels we want to extend, we can further cover the whole feature map using more coarser grain at even farther distance. We call this mechanism focal self-attention considering each token attends other tokens in a focal manner. This focal self-attention has the ability to capture both short- and long-range dependencies that approximates the original self-attention while introducing much less computational overhead after coarsening the tokens in the feature map.

Based on the proposed focal self-attention, we develop focal vision Transformers for image classification and object detection. It shares some common properties as previous works for dense prediction tasks: 1) We exploit a multi-stage architecture to maintain a reasonable computational cost for high-resolution images, following [55, 58, 35, 68]; 2) Instead of performing focal self-attention for each token, we split the feature map into multiple windows in which tokens share the same surroundings, similar to the strategies used in [51, 68, 35]. As a result, we focus our study on the effectiveness of our proposed focal self-attention. Extensive experiments show that our Focal Transformers with similar sizes and complexities outperform the current state-of-the-art Transformer models consistently across various settings, particularly on object detection. These results demonstrate the focal self-attention stand-alone is an effective strategy for modeling the local-global interactions in vision Transformers.

2 Related work

Vision Transformers. The Vision Transformer (ViT) [19] 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 [27], achieving state-of-the-art performance on multiple image classification benchmarks, where training data is sufficient. ViT has been improved from different perspectives, such as data-efficient training [49], improved patch embedding/encoding [15, 64, 25], introducing convolutional projects into transformers [58, 63], multi-stage ViT architecture and efficient attention mechanisms for high-resolution vision tasks [55, 58, 35, 68]. Our Focal ViT follows the multi-stage ViT architecture and introduces a new efficient attention mechanism – focal self-attention.

Local fine-grain and global coarse-grain attention. When there are large number of tokens in Transformer models, such as long document processing in NLP and high-resolution vision tasks in CV, efficient attention mechanisms are required to overcome the quadratic computational and memory increase in the vanilla self-attention mechanisms. Local fine-grain attention, i.e., attending

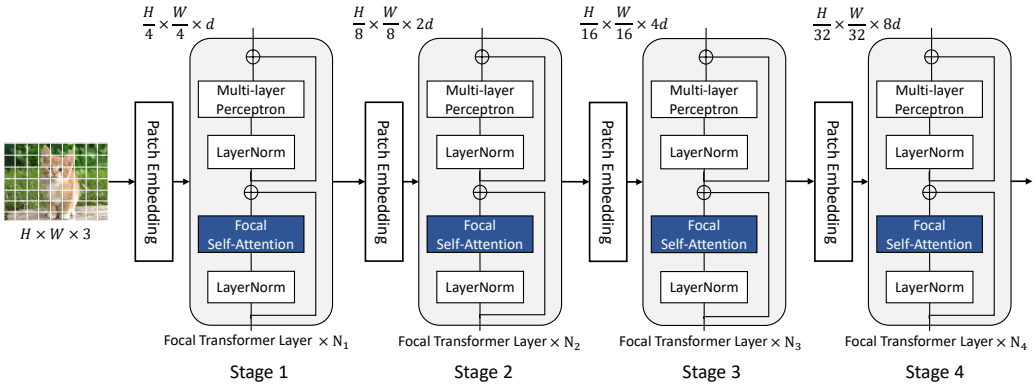


Figure 2: Model architecture for our focal vision Transformers. As highlighted by blue box, the main innovation part is the proposed focal self-attention module in each Transformer layer.

to neighboring tokens within a constant window size, is the most popular efficient attention in both NLP (see, e.g., [3, 66, 1]) and CV (see, e.g., [58, 35, 68]) scenarios. Global coarse-grain attention, i.e., attending to a small number of downsampled/summarized tokens, is another effective efficient attention in preserving the global communication among tokens in the vanilla self-attention; see, e.g., [41, 38] in NLP and [55, 58, 25] in CV. We argue that both types of attentions are important and the vanilla ViT model indeed had learned both of them, as shown in Fig. 1 left. Our proposed focal self-attention captures both types of attentions and approximates the vanilla full attention effectively. Finally, we refer to [48, 47, 68] for a comprehensive survey and benchmarks of various efficient attention mechanims in NLP and CV applications.

3 Method

3.1 Model Architecture

To accommodate the high-resolution vision tasks, our model architecture takes similar multi-scale design used in [55, 68, 35] which allows us to obtain high-resolution feature maps at earlier stages. As shown in Fig. 2, an image $I \in \mathcal{R}^{H \times W \times 3}$ is first partitioned into patches of size 4×4, resulting in $\frac{H}{4} \times \frac{W}{4}$ visual tokens with dimension $4 \times 4 \times 3$. Then, we use a patch embedding layer which consists of a convolutional layer with filter size and stride both equal to 4, to project these patches into hidden features with dimension d. Given this spatial feature map, we then pass it to four stages of focal Transformer blocks. At each stage $i \in \{1, 2, 3, 4\}$, the focal Transformer block consists of N_i focal Transformer layers. After each stage, we use another patch embedding layer to reduce the spatial size of feature map by factor 2, while the feature dimension is increased by 2. For image classification task, we take the average of the output from last stage and send it to a classification layer. For object detection, the feature maps from last 3 or all 4 stages are fed to the detector head depending on the particular detection method we use. The model capacity can be customized by varying the input feature dimension d and the number of focal Transformer layers at each stage $\{N_1, N_2, N_3, N_4\}$.

Standard self-attention can model both short- and long-range interactions at fine-grain, but it suffers from high computational cost when it performs the attention on high-resolution feature maps. Take stage 1 in Fig. 2 as the example. For the feature map of size $\frac{H}{4} \times \frac{W}{4} \times d$, the complexity of self-attention is $\mathcal{O}((\frac{H}{4} \times \frac{W}{4})^2 d)$, resulting in an explosion of time and memory cost considering $\min(H, W) = 800$ typically for object detection. In the next, we will explain how we address this by using focal self-attention.

3.2 Focal Self-Attention

In this paper, we propose focal self-attention to make Transformer layers scalable to high-resolution inputs. Instead of attending all tokens at fine-grain, we propose to attend the fine-grain tokens only locally but the summarized ones globally. As such, it can cover as much region as standard

self-attention but with much less cost. In Fig. 3, we show the area of receptive field for standard self-attention and our focal self-attention. As we can see, for a query position, when we use gradually coarser-grain for its far surroundings, it can have significantly larger receptive field at the cost of attending the same number of visual tokens.

Theoretically, our focal mechanism enables global self-attention with much less time and memory cost because it attends much less number of surrounding (summarized) tokens. In practice, however, extracting the surrounding tokens for each query position suffers from high time and memory cost since we need to duplicate each token for all queries that can get access to it. This practical issue has been exhaustively noted by a number of previous works [51, 68, 35] and the common solution is to partition the input feature map into windows. Inspired by them, we resort to perform focal attention at the window level. Given a feature map of $x \in \mathcal{R}^{M \times N \times d}$ with spatial size $M \times N$, we first partition it into a grid of windows with size $s_p \times s_p$. Then, we find the surroundings for each window rather than individual tokens. In the following, we elaborate the window-wise focal self-attention.

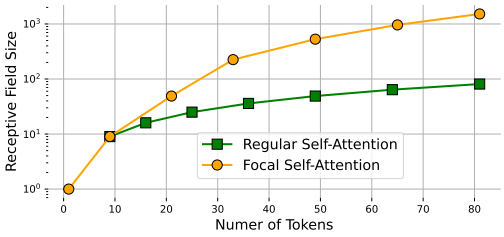


Figure 3: The receptive field size with the increase of tokens for standard self-attention and our focal self-attention. For focal self-attention, we increase the window granularity by factor 2 gradually but no more than 8. Note that the y-axis is logarithmic.

3.2.1 Window-wise Attention

An illustration of the proposed window-wise focal self-attention is shown in Fig. 4. Before we explain the details, we first define three terms for clarity:

- **Focal levels** L – the number of granularity levels we extract the tokens for our focal attention. In, Fig. 1, we show 3 focal levels in total for example.
- **Focal window size** s_w^l – the size of sub-window on which we get the summarized tokens at level $l \in \{1, ..., L\}$, which are 1, 2 and 4 for the three levels in Fig. 1.
- **Focal region size** s_r^l – the number of sub-windows horizontally and vertically in attended region at level l, and they are 3, 4 and 4 from level 1 to 3 in Fig. 1.

With the above three terms $\{L, s_w, s_r\}$, we can specify our focal self-attention module. Below we explain its two main steps:

Sub-window pooling. Given the input feature map $x \in \mathcal{R}^{M \times N \times d}$, we perform sub-window pooling for all L levels. For the focal level l, we first split the input feature map x into a grid of sub-windows with size $s_w^l \times s_w^l$. Then we use a simple linear layer f_p^l to pool the sub-windows spatially by:

$$x^l = f_p^l(\hat{x}) \in \mathcal{R}^{\frac{M}{s_w^l} \times \frac{N}{s_w^l} \times d}, \quad \hat{x} = Reshape(x) \in \mathcal{R}^{(\frac{M}{s_w^l} \times \frac{N}{s_w^l} \times d) \times (s_w^l \times s_w^l)}, \tag{1}$$

The pooled feature maps $\{x^l\}_1^L$ at different levels l provide rich information at both fine-grain and coarse-grain. Since we set $s_w^l = 1$ for the first focal level which has the same granularity as the input feature map, there is no need to perform any sub-window pooling. Considering the focal window size is usually very small (7 maximally in our settings), the number of extra parameters introduced by these sub-window pooling are fairly neglectable.

Attention computation. Once we obtain the pooled feature maps $\{x^l\}_1^L$ at all L levels, we compute the query at the first level and key and value for all levels using three linear projection layers f_q, f_k and f_v:

$$Q = f_q(x^1), \quad K = \{K^l\}_1^L = f_k(\{x^1, ..., x^L\}), \quad V = \{V^l\}_1^L = f_v(\{x^1, ..., x^L\}) \tag{2}$$

To perform focal self-attention, we need to first extract the surrounding tokens for each query token in the feature map. As we mentioned earlier, tokens inside a window partition $s_p \times s_p$ share the same set of surroundings. For the queries inside the i-th window $Q_i \in \mathcal{R}^{s_p \times s_p \times d}$, we extract the $s_r^l \times s_r^l$ keys and values from K^l and V^l around the window where the query lies in, and then gather the keys and values from all L to obtain $K_i = \{K_i^1, ..., K_i^L\} \in \mathcal{R}^{s \times d}$ and $V_i = \{V_i^1, ..., V_i^L\} \in \mathcal{R}^{s \times d}$, where s is the sum of focal region from all levels, *i.e.*,, $s = \sum_{l=1}^{L} (s_r^l)^2$. Note that a strict version

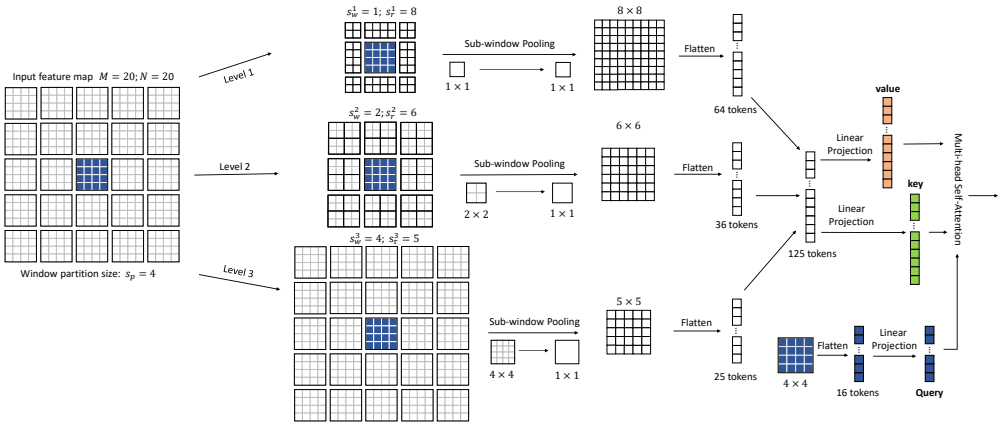


Figure 4: An illustration of our focal self-attention at window level. We suppose an input feature map of 20×20 and 4×4 window size. Each of the finest square cell represents a visual token either from the original feature map or the squeezed ones. Suppose we have an input feature map of size 20×20. We first partition it into 5×5 windows of size 4×4. Take the 4×4 blue window in the middle as the query, we extract its surroundings tokens at multiple granularity levels as its keys and values. For the first level, we extract the 8×8 tokens which are closest to the blue window at the finest grain. Then at the second level, we expand the attention region and pool the surrounding 2×2 sub-windows, which results in 6×6 pooled tokens. At the final level, we attend even larger region covering the whole feature map and pool 4×4 sub-windows. Finally, these three levels of tokens are concatenated to compute the keys and values for the $4 \times 4 = 16$ tokens (queries) in the blue window.

of focal self-attention following Fig. 1 requires to exclude the overlapped regions across different levels. In our model, we intentionally keep them in order to capture the pyramid information for the overlapped regions. Finally, we follow [35] to include a relative position bias and compute the focal self-attention for Q_i by:

$$Attention(Q_i, K_i, V_i) = Softmax(\frac{Q_i K_i^T}{\sqrt{d}} + B)V_i, \tag{3}$$

where $B = \{B^l\}_1^L$ is the learnable relative position bias. It consists of L subsets for L focal levels. Similar to [35], for the first level, we parameterize it to $B^1 \in \mathcal{R}^{(2s_p-1)\times(2s_p-1)}$, considering the horizontal and vertical position range are both in $[-s_p + 1, s_p - 1]$. For the other focal levels, considering they have different granularity to the queries, we treat all the queries inside a window equally and use $B^l \in \mathcal{R}^{s_r^l \times s_r^l}$ to represent the relative position bias between the query window and each of $s_r^l \times s_r^l$ pooled tokens. Since the focal self-attention for each window is independent of others, we can compute Eq. (3) in parallel. Once we complete it for the whole input feature map, we send it to the MLP block for proceeding computation as usual.

3.2.2 Complexity Analysis

We analyze the computational complexity for the two main steps discussed above. For the input feature map $x \in \mathcal{R}^{M\times N\times d}$, we have $\frac{M}{s_w^l} \times \frac{N}{s_w^l}$ sub-windows at focal level l. For each sub-window, the pooling operation in Eq.1 has the complexity of $\mathcal{O}((s_w^l)^2 d)$. Aggregating all sub-windows brings us $\mathcal{O}((MN)d)$. Then for all focal levels, we have the complexity of $\mathcal{O}(L(MN)d)$ in total, which is independent of the sub-window size at each focal level. Regarding the attention computation in Eq. 3, the computational cost for a query window $s_p \times s_p$ is $\mathcal{O}((s_p)^2 \sum_l (s_r^l)^2 d)$, and $\mathcal{O}(\sum_l (s_r^l)^2 (MN)d)$ for the whole input feature map. To sum up, the overall computational cost for our focal self-attention becomes $\mathcal{O}((L + \sum_l (s_r^l)^2)(MN)d)$. In an extreme case, one can set $s_r^L = 2\max(M, N)/s_w^L$ to ensure global receptive field for all queries (including both corner and middle queries) in this layer.

3.3 Model Configuration

We customize three different model capacities for our focal Transformers. Here, we simply follow the design strategy suggested by previous works [55, 58, 35], though we believe there should be a better

	Output Size	Layer Name	Focal-Tiny	Focal-Small	Focal-Base
	56×56	Patch Embedding	$p_1 = 4; c_1 = 96$	$p_1 = 4; c_1 = 96$	$p_1 = 4; c_1 = 128$
stage 1	56×56	Transformer Block	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 7\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 7\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 7\} \end{bmatrix} \times 2$
	28×28	Patch Embedding	$p_2 = 2; c_2 = 192$	$p_2 = 2; c_2 = 192$	$p_2 = 2; c_2 = 256$
stage 2	28×28	Transformer Block	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 5\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 5\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 5\} \end{bmatrix} \times 2$
	14×14	Patch Embedding	$p_3 = 2; c_3 = 384$	$p_3 = 2; c_3 = 384$	$p_3 = 2; c_3 = 512$
stage 3	14×14	Transformer Block	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 3\} \end{bmatrix} \times 6$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 3\} \end{bmatrix} \times 18$	$\begin{bmatrix} s^0_{w,r} = \{1, 13\} \\ s^1_{w,r} = \{7, 3\} \end{bmatrix} \times 18$
	7×7	Patch Embedding	$p_4 = 2; c_4 = 768$	$p_4 = 2; c_4 = 768$	$p_4 = 2; c_4 = 1024$
stage 4	7×7	Transformer Block	$\begin{bmatrix} s^0_{w,r} = \{1, 7\} \\ s^1_{w,r} = \{7, 1\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 7\} \\ s^1_{w,r} = \{7, 1\} \end{bmatrix} \times 2$	$\begin{bmatrix} s^0_{w,r} = \{1, 7\} \\ s^1_{w,r} = \{7, 1\} \end{bmatrix} \times 2$

Table 1: Model configurations for our focal Transformers. We introduce three configurations Focal-Tiny, Focal-Small and Focal-Base with different model capacities.

configuration specifically for our focal Transformers. Specifically, we use similar design to the tiny, small and base models in Swin Transformer [35], as shown in Table 1. Our models take 224×224 images as inputs and the window partition size is also set to 7 to make our models comparable to the Swin Transformers. For the focal self-attention layer, we introduce two levels, one for fine-grain local attention and one for coarse-grain global attention. Expect for the last stage, the focal region size is consistently set to 13 for the window partition size of 7, which means that we expand 3 tokens for each window partition. For the last stage, since the whole feature map is 7×7, the focal region size at level 0 is set to 7, which is sufficient to cover the entire feature map. For the coarse-grain global attention, we set its focal window size same to the window partition size 7, but gradually decrease the focal region size to get $\{7, 5, 3, 1\}$ for the four stages. For the patch embedding layer, the spatial reduction ratio p_i for four stages are all $\{4, 2, 2, 2\}$, while Focal-Base has a higher hidden dimension compared with Focal-Tiny and Focal-Small.

4 Experiments

4.1 Image Classification

We compare different methods on ImageNet-1K [17]. For fair comparison, we follow the training recipes in [49, 55]. All models are trained for 300 epochs with a batch size 1024. The initial learning rate is set to 10^{-3} with 20 epochs of linear warm-up starting from 10^{-5}. For optimization, we use AdamW [36] as the optimizer with a cosine learning rate scheduler. The weight decay is set to 0.05 and the maximal gradient norm is clipped to 5.0. We use the same set of data augmentation and regularization strategies used in [49] after excluding random erasing [73], repeated augmentation [4, 28] and exponential moving average (EMA) [40]. The stochastic depth drop rates are set to 0.2, 0.2 and 0.3 for our tiny, small and base models, respectively. During training, we crop images randomly to 224×224, while a center crop is used during evaluation on the validation set.

In Table 2, we summarize the results for baseline models and the current state-of-the-art models on image classification task. We can find our Focal Transformers consistently outperforms other methods with similar model size (#Params.) and computational complexity (GFLOPs). Specifically, Focal-Tiny improves over the Transformer baseline DeiT-Small/16 by 2.0%. Meanwhile, using the same model configure (2-2-6-2) and a few extra parameters and computations, our Focal-Tiny improves over Swin-Tiny by 1.0 points (81.2 $\rightarrow$ 82.2). When we increase the window size from 7 to 14 to match the settings in ViL-Small [68], the performance can be further improved to 82.5. For small and base models, our Focal Transformers still achieves slightly better performance than the others. Notably, our Focal-Small can even reach 83.5 which is better than all counterpart small and base models using much less parameters (51.1M). We refer the readers to our appendix for more detailed comparisons.

Model	#Params.	FLOPs	Top-1 (%)
ResNet-50 [27]	25.0	4.1	76.2
DeiT-Small/16 [49]	22.1	4.6	79.9
PVT-Small [55]	24.5	3.8	79.8
ViL-Small [68]	24.6	5.1	82.0
CvT-13 [58]	20.0	4.5	81.6
Swin-Tiny [35]	28.3	4.5	81.2
Focal-Tiny (Ours)	29.1	4.9	82.2
ResNet-101 [27]	45.0	7.9	77.4
PVT-Medium [55]	44.2	6.7	81.2
CvT-21 [58]	32.0	7.1	82.5
ViL-Medium [68]	39.7	9.1	83.3
Swin-Small [35]	49.6	8.7	83.1
Focal-Small (Ours)	51.1	9.1	**83.5**
ResNet-152 [27]	60.0	11.0	78.3
ViT-Base/16 [19]	86.6	17.6	77.9
DeiT-Base/16 [49]	86.6	17.5	81.8
PVT-Large [55]	61.4	9.8	81.7
ViL-Base [68]	55.7	13.4	83.2
Swin-Base [35]	87.8	15.4	83.4
Focal-Base (Ours)	89.8	16.0	**83.5**

Table 2: Comparison of image classification on ImageNet-1K for different models. Except for ViT-Base/16, all other models are trained and evaluated on 224×224 resolution.

Backbone	RetinaNet	Mask R-CNN	
	AP^b	AP^b	AP^m
ResNet-50 [27]	36.3	38.0	34.4
PVT-Small	40.4	40.4	37.8
ViL-Small [68]	41.6	41.8	38.5
Swin-Tiny [35]	42.0	43.7	39.8
Focal-Tiny (Ours)	**43.7** (+1.7)	**44.8** (+1.1)	**41.0** (+1.3)
ResNet-101 [27]	38.5	40.4	36.4
ResNeXt101-32x4d [60]	39.9	41.9	37.5
PVT-Medium [55]	41.9	42.0	39.0
ViL-Medium [68]	42.9	43.4	39.7
Swin-Small [35]	45.0	46.5	42.1
Focal-Small (Ours)	**45.6** (+0.6)	**47.4** (+0.9)	**42.8** (+0.7)
ResNeXt101-64x4d [60]	41.0	42.8	38.4
PVT-Large [55]	42.6	42.9	39.5
ViL-Base [68]	44.3	45.1	41.0
Swin-Base [35]	45.0	46.9	42.3
Focal-Base (Ours)	**46.3** (+1.3)	**47.8** (+0.9)	**43.2** (+0.9)

Table 3: Comparisons with CNN and Transformer baselines and state-of-the-art methods on COCO object detection. The box mAP (AP^b) and mask mAP (AP^m) are reported for RetinaNet and Mask R-CNN trained with 1x schedule. More detailed comparisons with 3x schedule are in Table 4.

4.2 Object Detection and Segmentation

We benchmark our models on object detection with COCO 2017 [34]. The pretrained models are used as visual backbones and then plug into two representative pipelines, RetinaNet [33] and Mask R-CNN [26]. All models are trained on the 118k training images and results reported on 5K validation set. We follow the standard to use two training schedules, 1x schedule with 12 epochs and 3x schedule with 36 epochs. For 1x schedule, we resize image's shorter side to 800 while keeping its longer side no more than 1333. For 3x schedule, we use multi-scale training strategy by randomly resizing its shorter side to the range of $[480, 800]$. Considering this higher input resolution, we adaptively increase the focal sizes at four stages to $(15, 13, 9, 7)$, to ensures the focal attention covers more than half of the image region (first two stages) to the whole image (last two stages). With the focal size increased, the relative position biases are accordingly up-sampled to corresponding sizes using bilinear interpolation. During training, we use AdamW [36] for optimization with initial learning rate 10^{-4} and weight decay 0.05. Similarly, we use 0.2,0.2 and 0.3 stochastic depth drop rates to regularize the training for our tiny, small and base models. Since Swin Transformer does not report the numbers on RetinaNet, we train it by ourselves using their official code with the same hyperparameters to our Focal Transformers.

In Table 3, we show the performance for both CNN-based models and the current Transformer-based state-of-the-arts methods. The bbox mAP (AP^b) and mask mAP (AP^m) are reported. As we can see, our Focal Transformers outperform the CNN-based models consistently with the gap of 4.8-7.1 points. Compared with the other methods which also use multi-scale Transformer architectures, we still observe substantial gain across all settings and metrics. Particularly, our Focal Transformers brings 0.7-1.7 points of mAP against the current best approach Swin Transformer [35] at comparable settings. Different from the other multi-scale Transformer models, our method can simultaneously enable both short-range fine-grain and long-range coarse-grain interactions for each visual token, and thus capture richer visual contexts at each layer for better dense predictions. To get better understanding on the models, we further train them with 3x schedule and show the detailed numbers for RetinaNet and Mask R-CNN in Table 4. For comprehension, we also list the number of parameters and the associated computational cost for each model. As we can see, even for 3x schedule, our models can still achieve 0.3-1.1 gain over the best Swin Transformer models at comparable settings.

To further verify the effectiveness of our proposed Focal Transformers, we follow [35] to train four different object detectors including Cascade R-CNN [7], ATSS [69], RepPoints [61] and Sparse R-CNN [45]. We use Focal-Tiny as the backbone and training all four models using 3x schedule. The box mAPs on COCO validation set are reported in Table 5. As we can see, our Focal-Tiny

Backbone	#Params (M)	FLOPs (G)	RetinaNet 3x schedule + MS						Mask R-CNN 3x schedule + MS					
			AP^b	AP^b_{50}	AP^b_{75}	AP_S	AP_M	AP_L	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50 [27]	37.7/44.2	239/260	39.0	58.4	41.8	22.4	42.8	51.6	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small[55]	34.2/44.1	226/245	42.2	62.7	45.0	26.2	45.2	57.2	43.0	65.3	46.9	39.9	62.5	42.8
ViL-Small [68]	35.7/45.0	252/174	42.9	63.8	45.6	27.8	46.4	56.3	43.4	64.9	47.0	39.6	62.1	42.4
Swin-Tiny [35]	38.5/47.8	245/264	45.0	65.9	48.4	29.7	48.9	58.1	46.0	68.1	50.3	41.6	65.1	44.9
Focal-Tiny (Ours)	39.4/48.8	265/291	**45.5**	**66.3**	**48.8**	**31.2**	**49.2**	**58.7**	**47.2**	**69.4**	**51.9**	**42.7**	**66.5**	**45.9**
ResNet101 [27]	56.7/63.2	315/336	40.9	60.1	44.0	23.7	45.0	53.8	42.8	63.2	47.1	38.5	60.1	41.3
ResNeXt101-32x4d [60]	56.4/62.8	319/340	41.4	61.0	44.3	23.9	45.5	53.7	44.0	64.4	48.0	39.2	61.4	41.9
PVT-Medium [55]	53.9/63.9	283/302	43.2	63.8	46.1	27.3	46.3	58.9	44.2	66.0	48.2	40.5	63.1	43.5
ViL-Medium [68]	50.8/60.1	339/261	43.7	64.6	46.4	27.9	47.1	56.9	44.6	66.3	48.5	40.7	63.8	43.7
Swin-Small [35]	59.8/69.1	335/354	46.4	67.0	50.1	31.0	50.1	60.3	48.5	70.2	53.5	43.3	67.3	46.6
Focal-Small (Ours)	61.7/71.2	367/401	**47.3**	**67.8**	**51.0**	**31.6**	**50.9**	**61.1**	**48.8**	**70.5**	**53.6**	**43.8**	**67.7**	**47.2**
ResNeXt101-64x4d [60]	95.5/102	473/493	41.8	61.5	44.4	25.2	45.4	54.6	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large[55]	71.1/81.0	345/364	43.4	63.6	46.1	26.1	46.0	59.5	44.5	66.0	48.3	40.7	63.4	43.7
ViL-Base [68]	66.7/76.1	443/365	44.7	65.5	47.6	29.9	48.0	58.1	45.7	67.2	49.9	41.3	64.4	44.5
Swin-Base [35]	98.4/107	477/496	45.8	66.4	49.1	29.9	49.4	60.3	48.5	69.8	53.2	43.4	66.8	46.9
Focal-Base (Ours)	100.8/110.0	514/533	**46.9**	**67.8**	**50.3**	**31.9**	**50.3**	**61.5**	**49.0**	**70.1**	**53.6**	**43.7**	**67.6**	**47.0**

Table 4: COCO object detection and segmentation results with RetinaNet [33] and Mask R-CNN [27]. All models are trained with 3x schedule and multi-scale inputs (MS). The numbers before and after "/" at column 2 and 3 are the model size and complexity for RetinaNet and Mask R-CNN, respectively.

Method	Backbone	#Param	FLOPs	AP^b	AP^b_{50}	AP^b_{75}
Cascade Mask R-CNN [7]	R-50	82.0	739	46.3	64.3	50.5
	Swin-T	85.6	742	50.5	69.3	54.9
	Focal-T	86.7	770	**51.5** (+1.0)	**70.6**	**55.9**
ATSS [69]	R-50	32.1	205	43.5	61.9	47.0
	Swin-T	35.7	212	47.2	66.5	51.3
	Focal-T	36.8	239	**49.5** (+2.3)	**68.8**	**53.9**
RepPointsV2 [61]	R-50	43.4	431	46.5	64.6	50.3
	Swin-T	44.1	437	50.0	68.5	54.2
	Focal-T	45.4	491	**51.2** (+1.2)	**70.4**	**54.9**
Sparse R-CNN [45]	R-50	106.1	166	44.5	63.4	48.2
	Swin-T	109.7	172	47.9	67.3	52.3
	Focal-T	110.8	196	**49.0** (+1.1)	**69.1**	**53.2**

Table 5: Comparison with ResNet-50 and Swin-Tiny across different object detection methods. We use Focal-Tiny as the backbone and train all models using 3x schedule.

Model	W-Size	FLOPs	Top-1 (%)	AP^b	AP^m
Swin-Tiny	7	4.5	81.2	43.7	39.8
	14	4.9	82.1	44.0	40.5
Focal-Tiny	7	4.9	82.2	44.9	41.1
	14	5.2	82.3	45.5	41.5

Table 6: Model performance with different window sizes.

Model	W-Shift	Top-1 (%)	AP^b	AP^m
Swin-Tiny	-	80.2	38.8	36.4
	✓	81.2	43.7	39.8
Focal-Tiny	-	82.2	44.8	41.0
	✓	81.9	44.9	41.1

Table 7: Model performance without and with window shift.

exceeds Swin-Tiny by 1.0-2.3 points on all methods. These significant and consistent improvements over different detection methods in addition to RetinaNet and Mask R-CNN suggest that our Focal Transformer can be used as a generic backbone for a variety of object detection methods.

4.3 Ablation Studies

Above we have shown the superior performance of our Focal Transformer. Here we conduct ablation studies to inspect the model's capacity from different aspects. We use Focal-Tiny as the target and experiment it on both image classification and object detection.

Contributions of short- and long-range interaction. We attempt to factorize the effect of short-range fine-grain and long-range coarse-grain interactions in our Focal Transformers. We ablate the original Focal-Tiny model to: a) Focal-Tiny-Window merely performing attention inside each window; b) Focal-Tiny-Local attending the additional fine-grain surrounding tokens and c) Focal-Tiny-Global attending the extra coarse-grain squeezed tokens. We train them using the same setting as Focal-Tiny and report their performance on image classification and object detection using Mask R-CNN 1x schedule. As we can see from Fig. 5, Focal-Tiny-Window suffers from significant drop on both image classification (82.2→80.1) and object detection (44.8→38.3). This is expected since the communication across windows are totally cut off at each Transformer layer. After we enable either the local fine-grain or global coarse-grain interactions (middle two columns), we observe significant jumps. Though they prompt richer interactions from different paths, finally both of them enable the

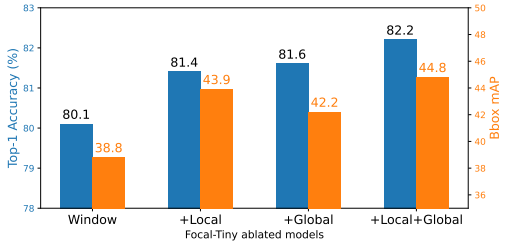


Figure 5: Ablating Focal-Tiny model by removing local, global and both interactions.

Depths	Model	#Params.	FLOPs	Top-1 (%)	AP^b	AP^m
2-2-2-2	Swin	21.2	3.1	78.7	38.2	35.7
	Focal	21.7	3.4	79.9	40.5	37.6
2-2-4-2	Swin	24.7	3.8	80.2	41.2	38.1
	Focal	25.4	4.1	81.4	43.3	39.8
2-2-6-2	Swin	28.3	4.5	81.2	43.7	39.8
	Focal	29.1	4.9	82.2	44.8	41.0

Table 8: Model performance with the change of model depth.

model to capture more contextual information. When we combine them together, we observe further improvements on both tasks. This implies that these two type of interactions are complementary to each other and both of them should be enabled in our model. Another observation is that adding long-range tokens can bring more relative improvement for image classification than object detection and vice versa for local tokens. We suspect that dense predictions like object detection more rely on fine-grained local context while image classification favors more the global information.

Effect of varying window size. Above we have demonstrated that both short- and long-range interactions are necessary. Based on this, a natural question is that whether increasing the window size can further help the model learning giving an enlarged receptive field. In Table 6, we show the performance of Swin-Tiny and Focal-Tiny with window size 7 and 14. Clearly, a larger window size brings gain for both methods on all three metrics and our Focal-Tiny model consistently outperforms Swin-Tiny using both window sizes. Comparing the second and third row, we find our model beats Swin even using much smaller window size (7 *v.s.* 14). We suspect the long-range interactions in our model is the source of this gain.

Model capacity against model depth. Considering our focal attention prompts local and global interactions at each Transformer layer, one question is that whether it needs less number of layers to obtain similar modeling capacity as those without global interactions. To answer this, we conduct the experiments by reducing the number of Transformer layers at stage 3 in Swin-Tiny and Focal-Tiny from the original 6 to 4 and 2. In Table 8, we show the performance and model complexity for each variant. First, we can find our model outperforms Swin model consistently with the same depth. More importantly, using two less layers, our model achieves comparable performance to Swin Transformer. Particularly, Focal-Tiny with 4 layers achieves 81.4 on image classification which is even better than original Swin-Tiny model with 6 layers (highlighted by blue color). Though we do not explore different architectures for our Focal Transformer, these results suggest that we can potential find even more efficient *and* effective architectures.

The necessary of window shift. In [35], the authors proposed window shift operation to enable the cross-window interactions using two successive layers. In contrast, the visual tokens in our Focal Transformer can always communicate with those in other windows at both fine- and coarse-grain. Then a natural question is whether adding the window shift to our Focal Transformers can further bring us improvements. To investigate, we remove the window shift from Swin Transformer while adding it to our Focal Transformer. As shown in Table 7, Swin Transformer has a severe degradation after removing the window shift. However, our Focal Transformer is even hurt on classification task. These results indicate that the window shift is not a necessary ingredient in our model. As such, our model can get rid of the constraint in Swin Transformer that there should be even number of layers at each stage for the alternative window shift operation.

5 Conclusion

In this paper, we presented a new attention mechanism called focal attention to enable efficient long-range interactions in vision transformers. Our design performs the local attention at fine-grain but global attention at coarse-grain which results in an effective way to capture richer context at a reasonable cost. By plugging this to a multi-scale vision transformer, we designed a new focal transformer and demonstrated its superiority over the state-of-the-art methods on both 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] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3286–3295, 2019.

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

[4] Maxim Berman, Hervé Jégou, Andrea Vedaldi, Iasonas Kokkinos, and Matthijs Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

[5] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE international conference on computer vision*, pages 5561–5569, 2017.

[6] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

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

[8] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[9] 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.

[10] Shuning Chang, Pichao Wang, Fan Wang, Hao Li, and Jiashi Feng. Augmented transformer with adaptive graph for temporal action proposal generation. *arXiv preprint arXiv:2103.16024*, 2021.

[11] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification, 2021.

[12] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019.

[13] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[14] Xin Chen, Bin Yan, Jiawen Zhu, Dong Wang, Xiaoyun Yang, and Huchuan Lu. Transformer tracking. *arXiv preprint arXiv:2103.15436*, 2021.

[15] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *Arxiv preprint 2102.10882*, 2021.

[16] 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.

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

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11592–11601, 2020.

[21] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 682–691, 2019.

[22] Jun Fu, Jing Liu, Yuhang Wang, Yong Li, Yongjun Bao, Jinhui Tang, and Hanqing Lu. Adaptive context network for scene parsing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6748–6757, 2019.

[23] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D Cubuk, Quoc V Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. *arXiv preprint arXiv:2012.07177*, 2020.

[24] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. *arXiv preprint arXiv:22104.01136*, 2021.

[25] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer, 2021.

[26] 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.

[27] 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.

[28] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020.

[29] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[30] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[31] Bing Li, Cheng Zheng, Silvio Giancola, and Bernard Ghanem. Sctn: Sparse convolution-transformer network for scene flow estimation. *arXiv preprint arXiv:2105.04447*, 2021.

[32] Xiangyu Li, Yonghong Hou, Pichao Wang, Zhimin Gao, Mingliang Xu, and Wanqing Li. Trear: Transformer-based rgb-d egocentric action recognition. *arXiv preprint arXiv:2101.03904*, 2021.

[33] 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.

[34] 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.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[37] Hyeonseob Nam, Jung-Woo Ha, and Jeonghee Kim. Dual attention networks for multimodal reasoning and matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 299–307, 2017.

[38] 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.

[39] 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.

[40] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[41] 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.

[42] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

[43] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition, 2021.

[44] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *CVPR*, 2019.

[45] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. *arXiv preprint arXiv:2011.12450*, 2020.

[46] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[47] 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.

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

[49] 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.

[50] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers, 2021.

[51] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. *arXiv preprint arXiv:2103.12731*, 2021.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[53] 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.

[54] Ning Wang, Wengang Zhou, Jie Wang, and Houqaing Li. Transformer meets tracker: Exploiting temporal context for robust visual tracking. *arXiv preprint arXiv:2103.11681*, 2021.

[55] 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.

[56] 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.

[57] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[58] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021.

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

[60] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[61] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9657–9666, 2019.

[62] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *European Conference on Computer Vision*, pages 191–207. Springer, 2020.

[63] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021.

[64] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[65] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019.

[66] 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.

[67] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Haibin Lin, Zhi Zhang, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

[68] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. *arXiv preprint arXiv:2103.15358*, 2021.

[69] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9759–9768, 2020.

[70] Jiaojiao Zhao, Xinyu Li, Chunhui Liu, Shuai Bing, Hao Chen, Cees GM Snoek, and Joseph Tighe. Tuber: Tube-transformer for action detection. *arXiv preprint arXiv:2104.00969*, 2021.

[71] 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.

[72] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020.

[73] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.

[74] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.

[75] Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

[76] 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.

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]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (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]
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]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

A Appendix

A.1 Image Classification

We present the exhaustive comparison with previous works in Table 9. We compare our method with both CNN-based and Transformer-based methods. We categorize different methods into groups based on two properties:

- **Scale** – the scale of feature maps in a model. It can be either a single-scale or multi-scale. In single-scale models, all feature maps have the same size across different stages. For multi-scale models, there are usually feature maps with different resolutions with the proceeding stages.
- **Locality** – the locality of operations in a model. It can be either global or local. Local operations can be a convoluional layer in CNN models or a transformer layer which conducts local self-attention. However, global operations such as the standard self-attention, produce the output feature map by gather information from all inputs.

Based on this criterion, all CNN models are natural multi-scale because their feature map sizes gradually decrease at different stages. Recently, a number of works attempt to integrate the global operations into CNNs by introducing squeeze-and-excitation (SE) layer [29], channel-wise attention layer [57] and even self-attention layer [2, 43]. As we can see, the combination of local and global operations significantly improve the performance for image classification. Particularly, BotNet-S1-110 achieves 82.8 top-1 accuracy with moderate number of parameters (61.6M).

On the contrary, Transformers [52] by nature performs global self-attention by which each visual token can interact with all others. Even without multi-scale design as in CNNs, a number of Transformer-based works such as TNT [25], DeepViT [75] and CaiT [50] achieve superior performance to CNN models with comparable model size and computational cost. To accommodate the high resolution feature maps, some recent works replace global self-attention with more efficient local self-attention and demonstrate comparable performance on image classification while much promising results on dense prediction tasks such as object detection and semantic segmentation [35].

In this paper, we present focal attention which is the first to combine global self-attention and local self-attention in an efficient way. Replacing either the global self-attention or the local self-attention with our focal self-attention, we achieve better performance than both. These results along with the CNN models augmented by local and global computations demonstrate that combining local and global interactions are more effective than either of them.

A.2 Object Detection and Segmentation

For completeness, we report the full metrics for RetinaNet and Mask R-CNN trained with 1x schedule in Table 10. As we can see, our Focal Transformers consistently outperform previous works including the state-of-the-art Swin Transformers on all metrics. We observe that our models trained with 1x schedule generally have more gain against the previous best models than 3x schedule (+1.2 *v.s.* +0.8 and +1.0 *v.s.* +0.7 box mAP for RetinaNet and Mask R-CNN, respectively). This indicates that our models have faster learning convergences compared with previous works. Compared with the local-attention based methods, *e.g.*, Swin Transformer, integrating the long-range interactions can help capture more visual dependencies and thus help the model to learn faster.

Comparing with system-level state-of-the-art methods. Giving the superior performance of our Focal Transformers on various standard benchmarks, we further increase the model capacity and compare it with previous state-of-the-art at system-level. Similar to Swin Transformers, we increase the hidden dimension in our Focal-Base from 128 to 196 while keep all the others the same.

Currently, the large models are usually pretrained on ImageNet-22K and then transferred to down stream tasks [58, 35]. However, due to the limited computational resources, we base our model on the pretrained Swin Transformer models[2], considering our model has similar architecture to Swin Transformers except for the window shift and global self-attention. Based on this, we reuse the parameters in Swin-Large model but remove the window shift operation and initialize our own window pooling layer and local-to-global relative position bias. This model is then used as the initial model for us to finetune on object detection and semantic segmentation tasks.

[2]Pretrained models are available at https://github.com/microsoft/Swin-Transformer

Architecture	Scale	Locality	Model	#Params. (M)	FLOPs (G)	Top-1 (%)
Convolutional Neural Network	Multiple	Local	ResNet-50 [27]	25.0	4.1	76.2
			ResNet-101 [27]	45.0	7.9	77.4
			ResNet-152 [27]	60.0	11.0	78.3
		Local +Global	SE-ResNet-50 [29]	28.1	8.2	77.5
			SE-ResNet-101 [29]	49.3	15.6	78.4
			SE-ResNet-152 [29]	66.8	23.1	78.9
			CBAM-ResNet-50 [57]	28.1	3.9	77.3
			CBAM-ResNet-101 [57]	49.3	7.6	78.5
			AttAug-ResNet-50 [2]	25.8	8.3	77.7
			AttAug-ResNet-101 [2]	45.4	16.1	78.1
			AttAug-ResNet-152 [2]	61.6	23.8	79.1
			BotNet-S1-59 [43]	33.5	7.3	81.7
			BotNet-S1-110 [43]	54.7	10.9	82.8
Transformer	Single	Global	ViT-B/16 [19]	86.6	17.6	77.9
			ViT-L/16 [19]	307.0	190.7	76.5
			DeiT-S/16 [49]	22.0	4.6	79.9
			DeiT-B/16 [49]	86.6	17.5	81.8
			TNT-S [25]	23.8	5.2	81.3
			TNT-B [25]	65.6	14.1	82.8
			CPVT-S [15]	23.0	4.6	81.5
			CPVT-B [15]	88.0	17.6	82.3
			DeepViT-S [75]	27.0	6.2	82.3
			DeepViT-L [75]	55.0	12.5	83.1
			CaiT-S36 [50]	68.0	13.9	83.3
			LeViT-256 [24]	18.9	1.1	81.6
			LeViT-384 [24]	39.1	2.3	82.6
	Multiple	Global	T2T-ViT-19 [64]	39.2	8.9	81.9
			T2T-ViT-24 [64]	64.1	14.1	82.3
			CrossViT-S [11]	26.7	5.6	81.0
			CrossViT-B [11]	104.7	21.2	82.2
			PVT-S [55]	24.5	3.8	79.8
			PVT-M [55]	44.2	6.7	81.2
			PVT-L [55]	61.4	9.8	81.7
			CvT-13 [58]	20.0	4.5	81.6
			CvT-21 [58]	32.0	7.1	82.5
		Local	ViL-S [68]	24.6	5.1	82.0
			ViL-M [68]	39.7	9.1	83.3
			ViL-B [68]	55.7	13.4	83.2
			Swin-T [35]	28.3	4.5	81.2
			Swin-S [35]	49.6	8.7	83.1
			Swin-B [35]	87.8	15.4	83.4
		Local +Global	Focal-T (Ours)	29.1	4.9	82.2
			Focal-S (Ours)	51.1	9.1	**83.5**
			Focal-B (Ours)	89.8	16.0	**83.5**

Table 9: Full comparison of image classification on ImageNet-1k for different model architectures.

For object detection on COCO, we follow Swin Transformer to also use HTC [12] as the detection method in that it has achieved state-of-the-art performance on COCO detection when using Swin Transformer as the backbone. For fair comparison, we also use soft-NMS [5], instaboost [21] and a multi-scale training strategy with shorter side in range $[400, 1400]$ while the longer side no more than 1600. We train the model using AdamW [36] with base learning rate 1e-4 and weight decay 0.1. The model is trained using standard 3x schedule. The box and mask mAPs on COCO validation set are reported in Table 11. We report both single-scale evaluation and multi-scale evaluation results. For single-scale, we resize the input images to $(1400, 2100)$, and use $(1200, 1800), (1300, 1950), (1400, 2100), (1450, 2200), (1500, 2250), (1600, 2400)$ for our multi-scale evaluation. As a reference, we also train HTC object detector with Swin-Large using the same regime as our Focal-Large model. As we can see, our Focal-Large model with multi-scale test achieve 57.9 box mAP and 50.8 mask mAP, which are on par and even better than the claimed numbers for Swin-Large in [35]. Note that because our model does not include global self-attention layer at the last stage, it has smaller model size and fewer FLOPs. Since the training script for

Backbone	#Params (M)	FLOPs (G)	RetinaNet 1x schedule						Mask R-CNN 1x schedule					
			AP^b	AP^b_{50}	AP^b_{75}	AP_S	AP_M	AP_L	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50 [27]	37.7/44.2	239/260	36.3	55.3	38.6	19.3	40.0	48.8	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small[55]	34.2/44.1	226/245	40.4	61.3	43.0	25.0	42.9	55.7	40.4	62.9	43.8	37.8	60.1	40.3
ViL-Small [68]	35.7/45.0	252/174	41.6	62.5	44.1	24.9	44.6	56.2	41.8	64.1	45.1	38.5	61.1	41.4
Swin-Tiny [35]	38.5/47.8	245/264	42.0	63.0	44.7	26.6	45.8	55.7	43.7	66.6	47.7	39.8	63.3	42.7
Focal-Tiny (Ours)	39.4/48.8	265/291	**43.7**	**65.2**	**46.7**	**28.6**	**47.4**	**56.9**	**44.8**	**67.7**	**49.2**	**41.0**	**64.7**	**44.2**
ResNet101 [27]	56.7/63.2	315/336	38.5	57.8	41.2	21.4	42.6	51.1	40.4	61.1	44.2	36.4	57.7	38.8
ResNeXt101-32x4d [60]	56.4/62.8	319/340	39.9	59.6	42.7	22.3	44.2	52.5	41.9	62.5	45.9	37.5	59.4	40.2
PVT-Medium [55]	53.9/63.9	283/302	41.9	63.1	44.3	25.0	44.9	57.6	42.0	64.4	45.6	39.0	61.6	42.1
ViL-Medium [68]	50.8/60.1	339/261	42.9	64.0	45.4	27.0	46.1	57.2	43.4	65.9	47.0	39.7	62.8	42.1
Swin-Small [35]	59.8/69.1	335/354	45.0	66.2	48.3	27.9	48.8	59.5	46.5	68.7	51.3	42.1	65.8	45.2
Focal-Small (Ours)	61.7/71.2	367/401	**45.6**	**67.0**	**48.7**	**29.5**	**49.5**	**60.3**	**47.4**	**69.8**	**51.9**	**42.8**	**66.6**	**46.1**
ResNeXt101-64x4d [60]	95.5/102	473/493	41.0	60.9	44.0	23.9	45.2	54.0	42.8	63.8	47.3	38.4	60.6	41.3
PVT-Large[55]	71.1/81.0	345/364	42.6	63.7	45.4	25.8	46.0	58.4	42.9	65.0	46.6	39.5	61.9	42.5
ViL-Base [68]	66.7/76.1	443/365	44.3	65.5	47.1	28.9	47.9	58.3	45.1	67.2	49.3	41.0	64.3	44.2
Swin-Base [35]	98.4/107	477/496	45.0	66.4	48.3	28.4	49.1	60.6	46.9	69.2	51.6	42.3	66.0	45.5
Focal-Base (Ours)	100.8/110.0	514/533	**46.3**	**68.0**	**49.8**	**31.7**	**50.4**	**60.8**	**47.8**	**70.2**	**52.5**	**43.2**	**67.3**	**46.5**

Table 10: COCO object detection and segmentation results with RetinaNet [33] and Mask R-CNN [27] trained with 1x schedule. This is a full version of Table 3. The numbers before and after "/" at column 2 and 3 are the model size and complexity for RetinaNet and Mask R-CNN, respectively.

Method	#Param	FLOPs	AP^b	AP^m
X101-64x4d [60]	155M	1033G	52.3	46.0
EfficientNet-D7 [46]	77M	410G	54.4	-
GCNet* [8]	-	1041G	51.8	44.7
ResNeSt-200 [67]	-	-	52.5	-
Copy-paste [23]	185M	1440G	55.9	47.2
BoTNet-200 [43]	-	-	49.7	-
SpineNet-190 [20]	164M	1885G	52.6	52.8
Swin-L [35]	284M	1470G	57.1	49.5
Swin-L† [35]	284M	-	**58.0**	50.4
Swin-L (Our run)	284M	1470G	55.9	48.9
Swin-L† (Our run)	284M	-	57.1	50.0
Focal-L (Ours)	265M	1165G	56.9	49.8
Focal-L† (Ours)	265M	-	57.9	**50.8**

Table 11: Comparison with state-of-the-art methods on COCO object detection and instance segmentation. The numbers are reported on 5K val set. HTC [12] is used as the detection method. † means multi-scale evaluation with flip.

Backbone	Method	#Param	FLOPs	mIoU	+MS
ResNet-101	DANet [37]	69M	1119G	45.3	-
ResNet-101	ACNet [22]	-	-	45.9	-
ResNet-101	DNL [62]	69M	1249G	46.0	-
ResNet-101	UperNet [59]	86M	1029G	44.9	-
HRNet-w48 [44]	OCRNet [65]	71M	664G	45.7	-
ResNeSt-200 [67]	DLab.v3+ [13]	88M	1381G	48.4	-
T-Large†	SETR [72]	308M	-	50.3	-
Swin-T [35]	UperNet [59]	60M	945G	44.5	45.8
Swin-S [35]	UperNet [59]	81M	1038G	47.6	49.5
Swin-B [35]	UperNet [59]	121M	1188G	48.1	49.7
Swin-L† [35]	UperNet [59]	234M	3230G	52.1	53.5
Focal-T (Ours)	UperNet [59]	62M	998G	45.8	47.0
Focal-S (Ours)	UperNet [59]	85M	1130G	48.0	50.0
Focal-B (Ours)	UperNet [59]	126M	1354G	49.0	50.5
Focal-L† (Ours)	UperNet [59]	240M	3376G	**52.3**	**53.8**

Table 12: Comparison with state-of-the-art methods for semantic segmentation on ADE20K [74] val set. Both single- and multi-scale evaluations are reported at the last two columns. † means pretrained on ImageNet-22K.

Swin-Large with HTC is not available, there is a small gap between our reproduced numbers and those reported in [35]. Comparing with the numbers from our reproduced Swin-Large model, our Focal-Large model clearly achieved better performance on both box and mask mAP.

Besides the comparison on object detection, we further compare our method with previous state-of-the-art methods on semantic segmentation. We benchmark our method on ADE20K [74] so that we can compare with the current SoTA method. Specifically, we use UperNet [59] as the segmentation method and our Focal Transformers as the backbone. We train four models with Focal-Tiny, Focal-Small, Focal-Base and Focal-Large, respectively. For all models except for Focal-Large, we use a standard setting by setting the input size to 512×512 and train the model for 160k iterations with batch size 16. For Focal-Large, we change the input size to 640×640 as in [35], and keep the other settings the same. In Table 12, we show the comparisons to previous works. As we can see, our tiny, small and base models consistently outperforms Swin Transformers with similar size. More importantly, our Focal-Large model initialized from Swin-Large achieves better performance than Swin-Large, which presents a new SoTA for semantic segmentation on ADE20K.

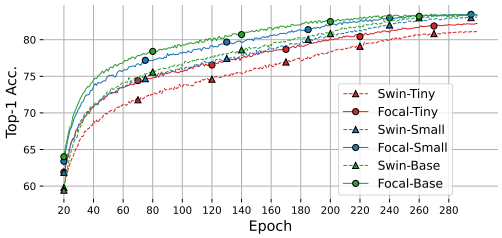


Figure 6: Training curves (Top-1 validation Acc.) for image classification with Swin Transformers and our Focal Transformers.

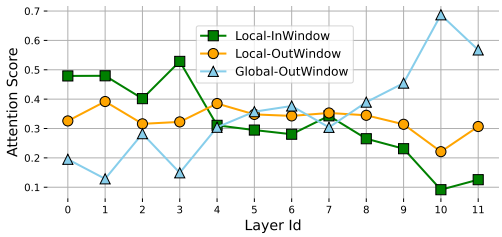


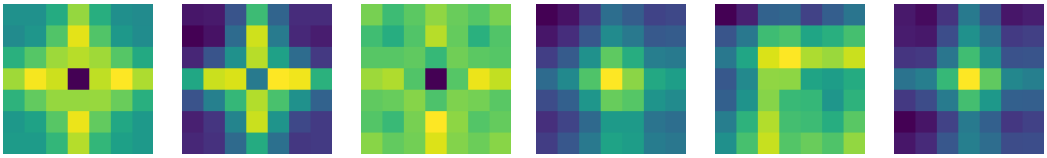
Figure 7: Summed up attention score at each layer for: a) local tokens inside window; b) local tokens surrounding window and c) global tokens.

A.3 Model Inspections

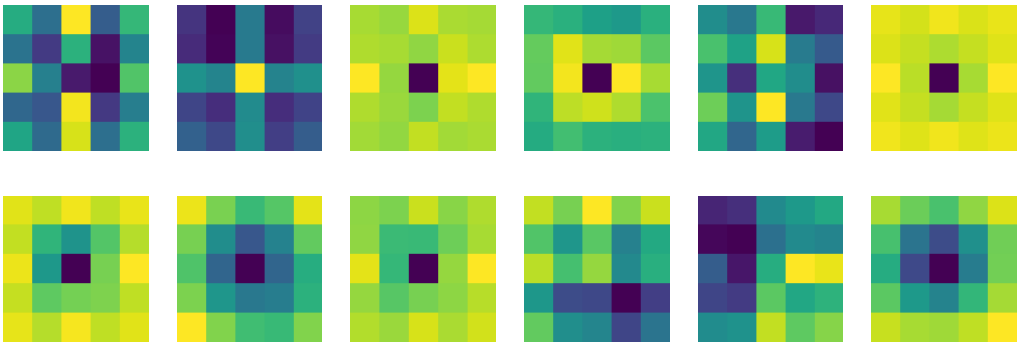
Learning speed comparison. As we briefly discussed earlier, our model shows faster learning speed on object detection task. In Fig. 6, we show the top-1 validation accuracy of our models and Swin Transformers for image classification task. Accordingly, our Focal Transformers have much faster learning speed as well. For example, Focal-Tiny has 75.7% top-1 accuracy at 100-th epoch while Swin-Tiny has 73.9% top-1 accuracy. Similarly, Focal-Small achieves 78.3% at 100-th epoch, which is 2.0 point higher than Swin-Small. Even for the base models, this gap is still maintained for a long duration until the end of the training. We attribute this faster learning speed to the long-range interactions introduce by our focal attention mechanism in that it can help to capture the global information at very beginning.

Attention scores for different token types. In our main submission, we have shown both local and global attentions are important. Here, we study how much local and global interactions occur at each layer. Using Focal-Tiny trained on ImageNet-1K as the target, we show in Fig. 7 the summed up attention scores for three type of tokens: 1) local tokens inside the window; 2) local tokens surrounding the window and 3) global tokens after the window pooling. To compute these scores, we average over the all local windows and then also take the average over all heads. Finally, we sum up the attention scores that belongs to the aforementioned three type of tokens. These attention scores are further averaged over the whole ImageNet-1K validation set. In Fig. 7, we can see a clear trend that the global attention becomes stronger when it goes to upper layers, while the local attention inside a window is weakened gradually. This indicates that: 1) our model heavily relies on both short- and long-range interactions. Neither of them are neglected in the model at all layers and stages; 2) the gradually strengthened global and weakened local attentions indicate that model tends to focus on more local details at earlier stages while on more global context at the later stages.

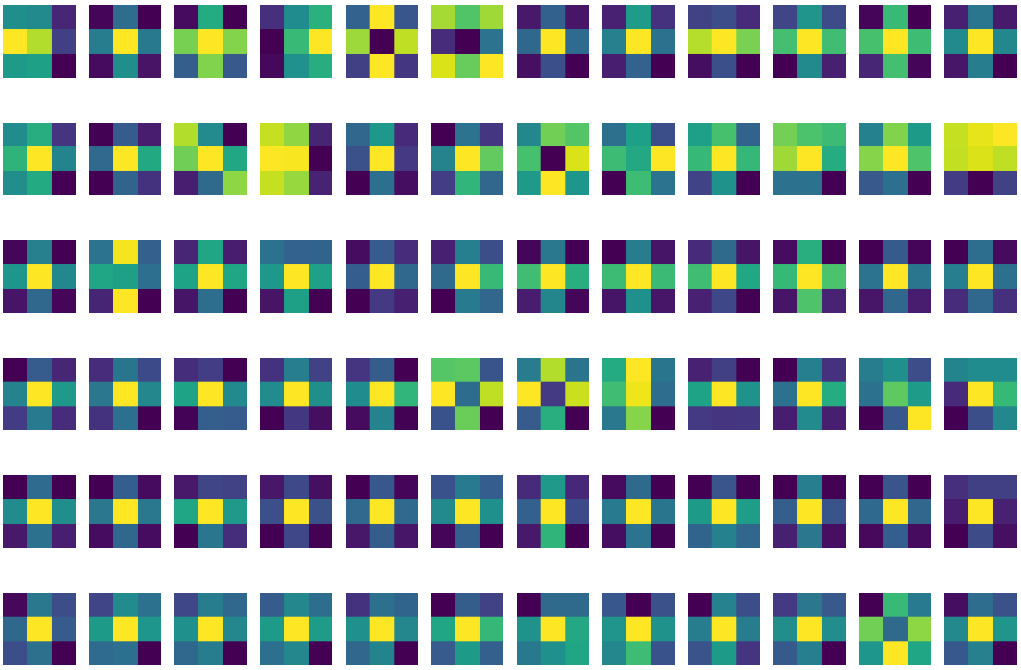
Local-to-global relative position bias. We further inspect what our model learns for the local to global relative position bias introduced in Eq. (3). This relative position bias is a good indicator on how the model put its attention weight on local and global regions. In our Focal Transformers, the focal region sizes at four stages are $(7, 5, 3, 1)$ and $(15, 13, 9, 7)$ for image classification and object detection, respectively. In Fig. 8 and Fig. 9, we visualize the learned relative position bias matrices for all heads and all layers in our Focal-Tiny model trained on ImageNet-1K and COCO, respectively. Surprisingly, though all are randomly initialized, these relative position biases exhibit some interesting patterns. At the first stage of image classification model, all three heads learn to put much less attention on the center window at first layer while focus more on the center at the second layer. For object detection model, however, they are swapped so that the first layer focus more on the center part while the second layer learns to extract the global context from surrounding. As a result, these the two layers cooperate with each other to extract both local and global information. At the second stage of both models, we observe similar property that the two consecutive layers have both local and global interactions. Compared with image classification model, the object detection model has more focus on the center regions. We suspect this is because object detection needs to extract more fine-grained information at local regions to predict the object category and location. At the third stage, we can see there is a fully mixture of local and global attentions in both models. Surprisingly, though randomly initialized, some of the heads automatically learn to disregard the center window pooled token which has much redundancy with the fine-grained tokens inside the center window.



(a) Stage 1, left 3 for first layer, right 3 for second layer, size=7 $\times$ 7

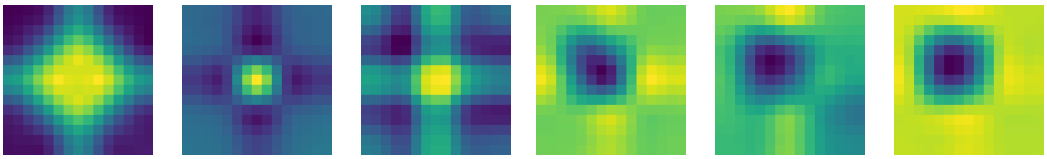


(b) Stage 2, top row for first layer and bottom row for second layer, 6 heads, size=5 $\times$ 5

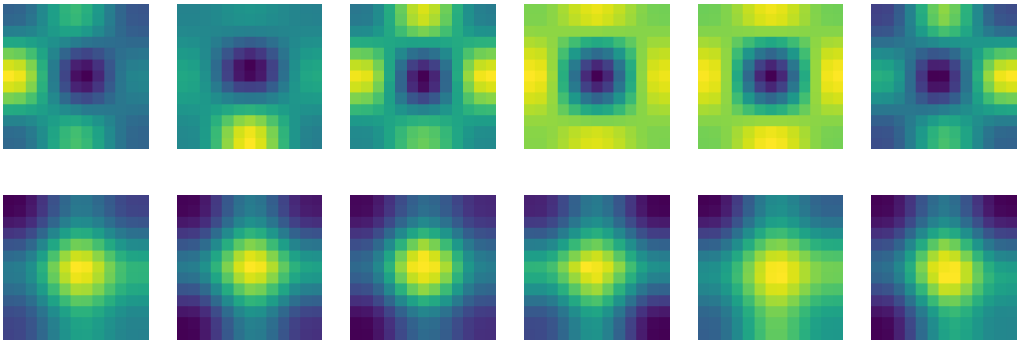


(c) Stage 3, 6 layers from top to bottom row, 12 heads, size=3 $\times$ 3

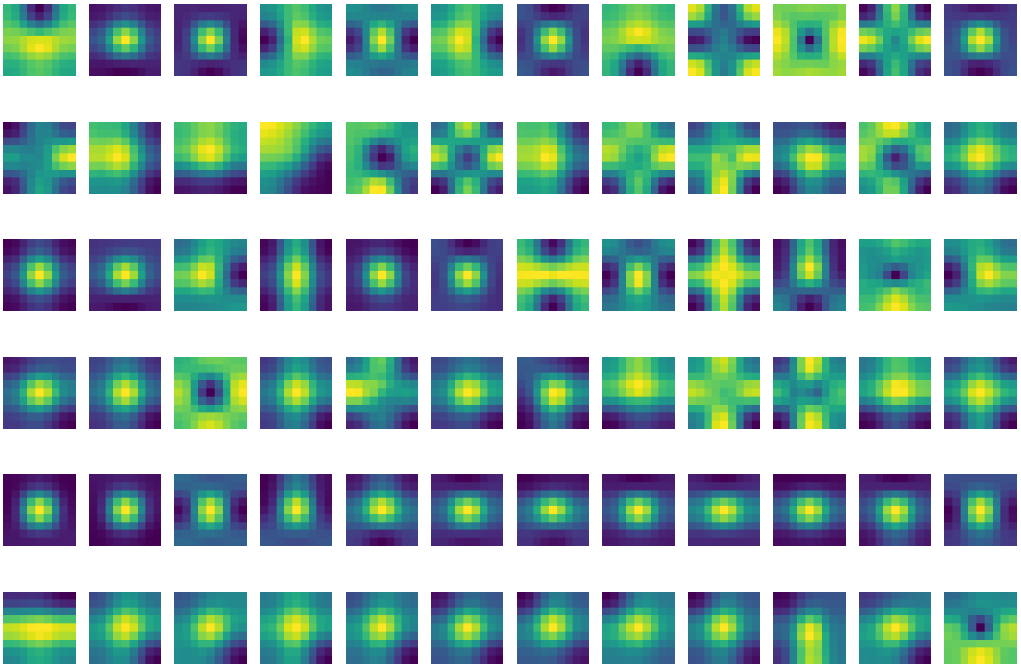
Figure 8: Learned relative position bias between local window and the global tokens in Focal-Tiny trained on ImageNet-1K. From top to bottom, we show the learned relative position bias for all heads at (a) stage 1, (b) stage 2 and (c) stage 3. Since the focal region size is 1 for stage 4 in classification models, we only show the first three stages.



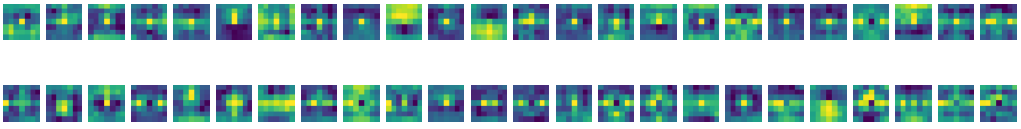
(a) Stage 1, left 3 for first layer and right 3 for second layer, 3 heads, size=15×15



(b) Stage 2, top row for first layer and bottom row for second layer, 6 heads, size=13×13



(c) Stage 3, 6 layers from top to bottom row, 12 heads, size=9×9



(d) Stage 4, top row for first layer and bottom row for second layer, 24 heads, size=7×7

Figure 9: Learned relative position bias between local window and the global tokens in Focal-Tiny for object detection trained on COCO. From top to bottom, we show the relative position bias for different heads at (a) stage 1, (b) stage 2, (c) stage 3 and (d) stage 4.