Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

Ze Liu[†*] Yutong Lin[†*] Yue Cao[*] Han Hu[*‡] Yixuan Wei[†]
Zheng Zhang Stephen Lin Baining Guo
Microsoft Research Asia
{v-zeliu1,v-yutlin,yuecao,hanhu,v-yixwe,zhez,stevelin,bainguo}@microsoft.com

Abstract

This paper presents a new vision Transformer, called Swin Transformer, that capably serves as a general-purpose backbone for computer vision. Challenges in adapting Transformer from language to vision arise from differences between the two domains, such as large variations in the scale of visual entities and the high resolution of pixels in images compared to words in text. To address these differences, we propose a hierarchical Transformer whose representation is computed with ***S**hifted* ***win**dows. The shifted windowing scheme brings greater efficiency by limiting self-attention computation to non-overlapping local windows while also allowing for cross-window connection. This hierarchical architecture has the flexibility to model at various scales and has linear computational complexity with respect to image size. These qualities of Swin Transformer make it compatible with a broad range of vision tasks, including image classification (87.3 top-1 accuracy on ImageNet-1K) and dense prediction tasks such as object detection (58.7 box AP and 51.1 mask AP on COCO test-dev) and semantic segmentation (53.5 mIoU on ADE20K val). Its performance surpasses the previous state-of-the-art by a large margin of +2.7 box AP and +2.6 mask AP on COCO, and +3.2 mIoU on ADE20K, demonstrating the potential of Transformer-based models as vision backbones. The hierarchical design and the shifted window approach also prove beneficial for all-MLP architectures. The code and models are publicly available at* `https://github.com/microsoft/Swin-Transformer`.

1. Introduction

Modeling in computer vision has long been dominated by convolutional neural networks (CNNs). Beginning with AlexNet [39] and its revolutionary performance on the ImageNet image classification challenge, CNN architectures have evolved to become increasingly powerful through

[*]Equal contribution. [†]Interns at MSRA. [‡]Contact person.

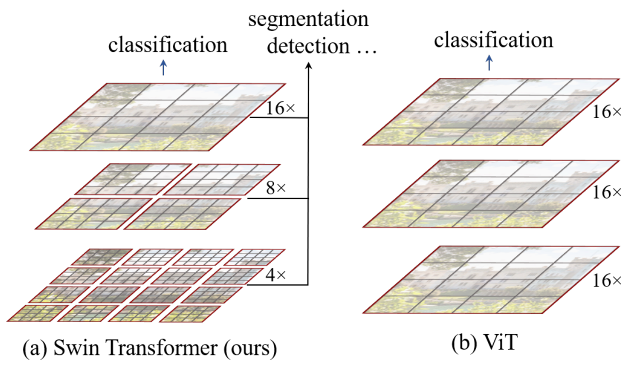


Figure 1. (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and has linear computation complexity to input image size due to computation of self-attention only within each local window (shown in red). It can thus serve as a general-purpose backbone for both image classification and dense recognition tasks. (b) In contrast, previous vision Transformers [20] produce feature maps of a single low resolution and have quadratic computation complexity to input image size due to computation of self-attention globally.

greater scale [30, 76], more extensive connections [34], and more sophisticated forms of convolution [70, 18, 84]. With CNNs serving as backbone networks for a variety of vision tasks, these architectural advances have led to performance improvements that have broadly lifted the entire field.

On the other hand, the evolution of network architectures in natural language processing (NLP) has taken a different path, where the prevalent architecture today is instead the Transformer [64]. Designed for sequence modeling and transduction tasks, the Transformer is notable for its use of attention to model long-range dependencies in the data. Its tremendous success in the language domain has led researchers to investigate its adaptation to computer vision, where it has recently demonstrated promising results on certain tasks, specifically image classification [20] and joint vision-language modeling [47].

In this paper, we seek to expand the applicability of Transformer such that it can serve as a general-purpose backbone for computer vision, as it does for NLP and as CNNs do in vision. We observe that significant challenges in transferring its high performance in the language domain to the visual domain can be explained by differences between the two modalities. One of these differences involves scale. Unlike the word tokens that serve as the basic elements of processing in language Transformers, visual elements can vary substantially in scale, a problem that receives attention in tasks such as object detection [42, 53, 54]. In existing Transformer-based models [64, 20], tokens are all of a fixed scale, a property unsuitable for these vision applications. Another difference is the much higher resolution of pixels in images compared to words in passages of text. There exist many vision tasks such as semantic segmentation that require dense prediction at the pixel level, and this would be intractable for Transformer on high-resolution images, as the computational complexity of its self-attention is quadratic to image size. To overcome these issues, we propose a general-purpose Transformer backbone, called Swin Transformer, which constructs hierarchical feature maps and has linear computational complexity to image size. As illustrated in Figure 1(a), Swin Transformer constructs a hierarchical representation by starting from small-sized patches (outlined in gray) and gradually merging neighboring patches in deeper Transformer layers. With these hierarchical feature maps, the Swin Transformer model can conveniently leverage advanced techniques for dense prediction such as feature pyramid networks (FPN) [42] or U-Net [51]. The linear computational complexity is achieved by computing self-attention locally within non-overlapping windows that partition an image (outlined in red). The number of patches in each window is fixed, and thus the complexity becomes linear to image size. These merits make Swin Transformer suitable as a general-purpose backbone for various vision tasks, in contrast to previous Transformer based architectures [20] which produce feature maps of a single resolution and have quadratic complexity.

A key design element of Swin Transformer is its *shift* of the window partition between consecutive self-attention layers, as illustrated in Figure 2. The shifted windows bridge the windows of the preceding layer, providing connections among them that significantly enhance modeling power (see Table 4). This strategy is also efficient in regards to real-world latency: all *query* patches within a window share the same *key* set[1], which facilitates memory access in hardware. In contrast, earlier *sliding window* based self-attention approaches [33, 50] suffer from low latency on general hardware due to different *key* sets for different *query* pixels[2]. Our experiments show that the proposed

[1]The *query* and *key* are projection vectors in a self-attention layer.

[2]While there are efficient methods to implement a sliding-window based convolution layer on general hardware, thanks to its shared kernel weights across a feature map, it is difficult for a sliding-window based self-attention layer to have efficient memory access in practice.

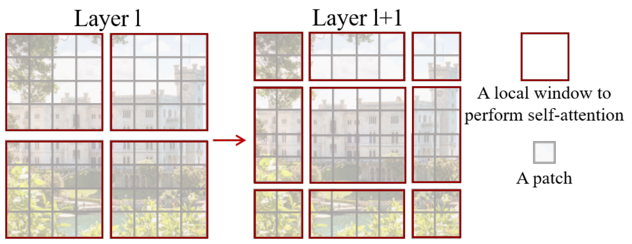


Figure 2. An illustration of the *shifted window* approach for computing self-attention in the proposed Swin Transformer architecture. In layer l (left), a regular window partitioning scheme is adopted, and self-attention is computed within each window. In the next layer $l + 1$ (right), the window partitioning is shifted, resulting in new windows. The self-attention computation in the new windows crosses the boundaries of the previous windows in layer l, providing connections among them.

shifted window approach has much lower latency than the *sliding window* method, yet is similar in modeling power (see Tables 5 and 6). The shifted window approach also proves beneficial for all-MLP architectures [61].

The proposed Swin Transformer achieves strong performance on the recognition tasks of image classification, object detection and semantic segmentation. It outperforms the ViT / DeiT [20, 63] and ResNe(X)t models [30, 70] significantly with similar latency on the three tasks. Its 58.7 box AP and 51.1 mask AP on the COCO test-dev set surpass the previous state-of-the-art results by +2.7 box AP (Copy-paste [26] without external data) and +2.6 mask AP (DetectoRS [46]). On ADE20K semantic segmentation, it obtains 53.5 mIoU on the val set, an improvement of +3.2 mIoU over the previous state-of-the-art (SETR [81]). It also achieves a top-1 accuracy of 87.3% on ImageNet-1K image classification.

It is our belief that a unified architecture across computer vision and natural language processing could benefit both fields, since it would facilitate joint modeling of visual and textual signals and the modeling knowledge from both domains can be more deeply shared. We hope that Swin Transformer's strong performance on various vision problems can drive this belief deeper in the community and encourage unified modeling of vision and language signals.

2. Related Work

CNN and variants CNNs serve as the standard network model throughout computer vision. While the CNN has existed for several decades [40], it was not until the introduction of AlexNet [39] that the CNN took off and became mainstream. Since then, deeper and more effective convolutional neural architectures have been proposed to further propel the deep learning wave in computer vision, e.g., VGG [52], GoogleNet [57], ResNet [30], DenseNet [34], HRNet [65], and EfficientNet [58]. In addition to these architectural advances, there has also been much work on improving individual convolution layers, such as depth-wise convolution [70] and deformable convolution [18, 84]. While the CNN and its variants are still the primary backbone architectures for computer vision applications, we highlight the strong potential of Transformer-like architectures for unified modeling between vision and language. Our work achieves strong performance on several basic visual recognition tasks, and we hope it will contribute to a modeling shift.

Self-attention based backbone architectures Also inspired by the success of self-attention layers and Transformer architectures in the NLP field, some works employ self-attention layers to replace some or all of the spatial convolution layers in the popular ResNet [33, 50, 80]. In these works, the self-attention is computed within a local window of each pixel to expedite optimization [33], and they achieve slightly better accuracy/FLOPs trade-offs than the counterpart ResNet architecture. However, their costly memory access causes their actual latency to be significantly larger than that of the convolutional networks [33]. Instead of using sliding windows, we propose to *shift* windows between consecutive layers, which allows for a more efficient implementation in general hardware.

Self-attention/Transformers to complement CNNs Another line of work is to augment a standard CNN architecture with self-attention layers or Transformers. The self-attention layers can complement backbones [67, 7, 3, 71, 23, 74, 55] or head networks [32, 27] by providing the capability to encode distant dependencies or heterogeneous interactions. More recently, the encoder-decoder design in Transformer has been applied for the object detection and instance segmentation tasks [8, 13, 85, 56]. Our work explores the adaptation of Transformers for basic visual feature extraction and is complementary to these works.

Transformer based vision backbones Most related to our work is the Vision Transformer (ViT) [20] and its follow-ups [63, 72, 15, 28, 66]. The pioneering work of ViT directly applies a Transformer architecture on non-overlapping medium-sized image patches for image classification. It achieves an impressive speed-accuracy trade-off on image classification compared to convolutional networks. While ViT requires large-scale training datasets (i.e., JFT-300M) to perform well, DeiT [63] introduces several training strategies that allow ViT to also be effective using the smaller ImageNet-1K dataset. The results of ViT on image classification are encouraging, but its architecture is unsuitable for use as a general-purpose backbone network on dense vision tasks or when the input image resolution is high, due to its low-resolution feature maps and the quadratic increase in complexity with image size. There are a few works applying ViT models to the dense vision tasks of object detection and semantic segmentation by direct upsampling or deconvolution but with relatively lower performance [2, 81]. Concurrent to our work are some that modify the ViT architecture [72, 15, 28] for better image classification. Empirically, we find our Swin Transformer architecture to achieve the best speed-accuracy trade-off among these methods on image classification, even though our work focuses on general-purpose performance rather than specifically on classification. Another concurrent work [66] explores a similar line of thinking to build multi-resolution feature maps on Transformers. Its complexity is still quadratic to image size, while ours is linear and also operates locally which has proven beneficial in modeling the high correlation in visual signals [36, 25, 41]. Our approach is both efficient and effective, achieving state-of-the-art accuracy on both COCO object detection and ADE20K semantic segmentation.

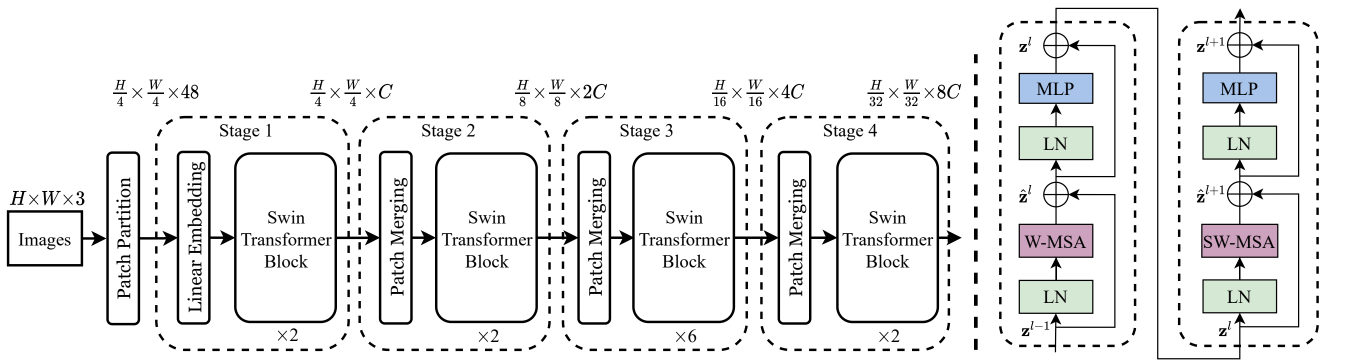
3. Method

3.1. Overall Architecture

An overview of the Swin Transformer architecture is presented in Figure 3, which illustrates the tiny version (Swin-T). It first splits an input RGB image into non-overlapping patches by a patch splitting module, like ViT. Each patch is treated as a "token" and its feature is set as a concatenation of the raw pixel RGB values. In our implementation, we use a patch size of 4×4 and thus the feature dimension of each patch is $4 \times 4 \times 3 = 48$. A linear embedding layer is applied on this raw-valued feature to project it to an arbitrary dimension (denoted as C).

Several Transformer blocks with modified self-attention computation (*Swin Transformer blocks*) are applied on these patch tokens. The Transformer blocks maintain the number of tokens ($\frac{H}{4} \times \frac{W}{4}$), and together with the linear embedding are referred to as "Stage 1".

To produce a hierarchical representation, the number of tokens is reduced by patch merging layers as the network gets deeper. The first patch merging layer concatenates the features of each group of 2×2 neighboring patches, and applies a linear layer on the $4C$-dimensional concatenated features. This reduces the number of tokens by a multiple of $2 \times 2 = 4$ ($2\times$ downsampling of resolution), and the output dimension is set to $2C$. Swin Transformer blocks are applied afterwards for feature transformation, with the resolution kept at $\frac{H}{8} \times \frac{W}{8}$. This first block of patch merging and feature transformation is denoted as "Stage 2". The procedure is repeated twice, as "Stage 3" and "Stage 4", with output resolutions of $\frac{H}{16} \times \frac{W}{16}$ and $\frac{H}{32} \times \frac{W}{32}$, respectively. These stages jointly produce a hierarchical representation,



(a) Architecture (b) Two Successive Swin Transformer Blocks

Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks (notation presented with Eq. (3)). W-MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

with the same feature map resolutions as those of typical convolutional networks, e.g., VGG [52] and ResNet [30]. As a result, the proposed architecture can conveniently replace the backbone networks in existing methods for various vision tasks.

Swin Transformer block Swin Transformer is built by replacing the standard multi-head self attention (MSA) module in a Transformer block by a module based on shifted windows (described in Section 3.2), with other layers kept the same. As illustrated in Figure 3(b), a Swin Transformer block consists of a shifted window based MSA module, followed by a 2-layer MLP with GELU nonlinearity in between. A LayerNorm (LN) layer is applied before each MSA module and each MLP, and a residual connection is applied after each module.

3.2. Shifted Window based Self-Attention

The standard Transformer architecture [64] and its adaptation for image classification [20] both conduct global self-attention, where the relationships between a token and all other tokens are computed. The global computation leads to quadratic complexity with respect to the number of tokens, making it unsuitable for many vision problems requiring an immense set of tokens for dense prediction or to represent a high-resolution image.

Self-attention in non-overlapped windows For efficient modeling, we propose to compute self-attention within local windows. The windows are arranged to evenly partition the image in a non-overlapping manner. Supposing each window contains $M \times M$ patches, the computational complexity of a global MSA module and a window based one on an image of $h \times w$ patches are[3]:

$$\Omega(\text{MSA}) = 4hwC^2 + 2(hw)^2C, \tag{1}$$
$$\Omega(\text{W-MSA}) = 4hwC^2 + 2M^2hwC, \tag{2}$$

where the former is quadratic to patch number hw, and the latter is linear when M is fixed (set to 7 by default). Global self-attention computation is generally unaffordable for a large hw, while the window based self-attention is scalable.

Shifted window partitioning in successive blocks The window-based self-attention module lacks connections across windows, which limits its modeling power. To introduce cross-window connections while maintaining the efficient computation of non-overlapping windows, we propose a shifted window partitioning approach which alternates between two partitioning configurations in consecutive Swin Transformer blocks.

As illustrated in Figure 2, the first module uses a regular window partitioning strategy which starts from the top-left pixel, and the 8×8 feature map is evenly partitioned into 2×2 windows of size 4×4 ($M = 4$). Then, the next module adopts a windowing configuration that is shifted from that of the preceding layer, by displacing the windows by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels from the regularly partitioned windows.

With the shifted window partitioning approach, consecutive Swin Transformer blocks are computed as

$$\begin{aligned}
&\hat{\mathbf{z}}^{l} = \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1},\\
&\mathbf{z}^{l} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l},\\
&\hat{\mathbf{z}}^{l+1} = \text{SW-MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l},\\
&\mathbf{z}^{l+1} = \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},
\end{aligned} \tag{3}$$

where $\hat{\mathbf{z}}^{l}$ and $\mathbf{z}^{l}$ denote the output features of the (S)W-MSA module and the MLP module for block l, respectively;

[3]We omit SoftMax computation in determining complexity.

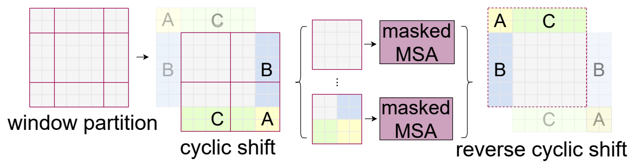


Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

W-MSA and SW-MSA denote window based multi-head self-attention using regular and shifted window partitioning configurations, respectively.

The shifted window partitioning approach introduces connections between neighboring non-overlapping windows in the previous layer and is found to be effective in image classification, object detection, and semantic segmentation, as shown in Table 4.

Efficient batch computation for shifted configuration An issue with shifted window partitioning is that it will result in more windows, from $\lceil \frac{h}{M} \rceil \times \lceil \frac{w}{M} \rceil$ to $(\lceil \frac{h}{M} \rceil + 1) \times (\lceil \frac{w}{M} \rceil + 1)$ in the shifted configuration, and some of the windows will be smaller than $M \times M$[4]. A naive solution is to pad the smaller windows to a size of $M \times M$ and mask out the padded values when computing attention. When the number of windows in regular partitioning is small, e.g. 2×2, the increased computation with this naive solution is considerable ($2 \times 2 \rightarrow 3 \times 3$, which is 2.25 times greater). Here, we propose a *more efficient batch computation approach* by cyclic-shifting toward the top-left direction, as illustrated in Figure 4. After this shift, a batched window may be composed of several sub-windows that are not adjacent in the feature map, so a masking mechanism is employed to limit self-attention computation to within each sub-window. With the cyclic-shift, the number of batched windows remains the same as that of regular window partitioning, and thus is also efficient. The low latency of this approach is shown in Table 5.

Relative position bias In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias $B \in \mathbb{R}^{M^2 \times M^2}$ to each head in computing similarity:

$$\text{Attention}(Q, K, V) = \text{SoftMax}(QK^T/\sqrt{d} + B)V, \quad (4)$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ are the *query*, *key* and *value* matrices; d is the *query*/*key* dimension, and M^2 is the number of patches in a window. Since the relative position along each axis lies in the range $[-M+1, M-1]$, we parameterize a smaller-sized bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$, and values in B are taken from $\hat{B}$.

[4]To make the window size (M, M) divisible by the feature map size of (h, w), bottom-right padding is employed on the feature map if needed.

We observe significant improvements over counterparts without this bias term or that use absolute position embedding, as shown in Table 4. Further adding absolute position embedding to the input as in [20] drops performance slightly, thus it is not adopted in our implementation.

The learnt relative position bias in pre-training can be also used to initialize a model for fine-tuning with a different window size through bi-cubic interpolation [20, 63].

3.3. Architecture Variants

We build our base model, called Swin-B, to have of model size and computation complexity similar to ViT-B/DeiT-B. We also introduce Swin-T, Swin-S and Swin-L, which are versions of about $0.25\times$, $0.5\times$ and $2\times$ the model size and computational complexity, respectively. Note that the complexity of Swin-T and Swin-S are similar to those of ResNet-50 (DeiT-S) and ResNet-101, respectively. The window size is set to $M = 7$ by default. The query dimension of each head is $d = 32$, and the expansion layer of each MLP is $\alpha = 4$, for all experiments. The architecture hyper-parameters of these model variants are:

- Swin-T: $C = 96$, layer numbers = $\{2, 2, 6, 2\}$
- Swin-S: $C = 96$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-B: $C = 128$, layer numbers =$\{2, 2, 18, 2\}$
- Swin-L: $C = 192$, layer numbers =$\{2, 2, 18, 2\}$

where C is the channel number of the hidden layers in the first stage. The model size, theoretical computational complexity (FLOPs), and throughput of the model variants for ImageNet image classification are listed in Table 1.

4. Experiments

We conduct experiments on ImageNet-1K image classification [19], COCO object detection [43], and ADE20K semantic segmentation [83]. In the following, we first compare the proposed Swin Transformer architecture with the previous state-of-the-arts on the three tasks. Then, we ablate the important design elements of Swin Transformer.

4.1. Image Classification on ImageNet-1K

Settings For image classification, we benchmark the proposed Swin Transformer on ImageNet-1K [19], which contains 1.28M training images and 50K validation images from 1,000 classes. The top-1 accuracy on a single crop is reported. We consider two training settings:

- *Regular ImageNet-1K training*. This setting mostly follows [63]. We employ an AdamW [37] optimizer for 300 epochs using a cosine decay learning rate scheduler and 20 epochs of linear warm-up. A batch size of 1024, an initial learning rate of 0.001, and a weight decay of 0.05 are used. We include most of the augmentation and regularization strategies of [63] in training, except for repeated augmentation [31] and EMA [45], which do not enhance performance. Note that this is contrary to [63] where repeated augmentation is crucial to stabilize the training of ViT.

- *Pre-training on ImageNet-22K and fine-tuning on ImageNet-1K.* We also pre-train on the larger ImageNet-22K dataset, which contains 14.2 million images and 22K classes. We employ an AdamW optimizer for 90 epochs using a linear decay learning rate scheduler with a 5-epoch linear warm-up. A batch size of 4096, an initial learning rate of 0.001, and a weight decay of 0.01 are used. In ImageNet-1K fine-tuning, we train the models for 30 epochs with a batch size of 1024, a constant learning rate of 10^{-5}, and a weight decay of 10^{-8}.

Results with regular ImageNet-1K training Table 1(a) presents comparisons to other backbones, including both Transformer-based and ConvNet-based, using regular ImageNet-1K training.

Compared to the previous state-of-the-art Transformer-based architecture, i.e. DeiT [63], Swin Transformers noticeably surpass the counterpart DeiT architectures with similar complexities: +1.5% for Swin-T (81.3%) over DeiT-S (79.8%) using 224^2 input, and +1.5%/1.4% for Swin-B (83.3%/84.5%) over DeiT-B (81.8%/83.1%) using 224^2/384^2 input, respectively.

Compared with the state-of-the-art ConvNets, i.e. RegNet [48] and EfficientNet [58], the Swin Transformer achieves a slightly better speed-accuracy trade-off. Noting that while RegNet [48] and EfficientNet [58] are obtained via a thorough architecture search, the proposed Swin Transformer is adapted from the standard Transformer and has strong potential for further improvement.

Results with ImageNet-22K pre-training We also pre-train the larger-capacity Swin-B and Swin-L on ImageNet-22K. Results fine-tuned on ImageNet-1K image classification are shown in Table 1(b). For Swin-B, the ImageNet-22K pre-training brings 1.8%∼1.9% gains over training on ImageNet-1K from scratch. Compared with the previous best results for ImageNet-22K pre-training, our models achieve significantly better speed-accuracy trade-offs: Swin-B obtains 86.4% top-1 accuracy, which is 2.4% higher than that of ViT with similar inference throughput (84.7 vs. 85.9 images/sec) and slightly lower FLOPs (47.0G vs. 55.4G). The larger Swin-L model achieves 87.3% top-1 accuracy, +0.9% better than that of the Swin-B model.

(a) Regular ImageNet-1K trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
RegNetY-4G [48]	224^2	21M	4.0G	1156.7	80.0
RegNetY-8G [48]	224^2	39M	8.0G	591.6	81.7
RegNetY-16G [48]	224^2	84M	16.0G	334.7	82.9
EffNet-B3 [58]	300^2	12M	1.8G	732.1	81.6
EffNet-B4 [58]	380^2	19M	4.2G	349.4	82.9
EffNet-B5 [58]	456^2	30M	9.9G	169.1	83.6
EffNet-B6 [58]	528^2	43M	19.0G	96.9	84.0
EffNet-B7 [58]	600^2	66M	37.0G	55.1	84.3
ViT-B/16 [20]	384^2	86M	55.4G	85.9	77.9
ViT-L/16 [20]	384^2	307M	190.7G	27.3	76.5
DeiT-S [63]	224^2	22M	4.6G	940.4	79.8
DeiT-B [63]	224^2	86M	17.5G	292.3	81.8
DeiT-B [63]	384^2	86M	55.4G	85.9	83.1
Swin-T	224^2	29M	4.5G	755.2	81.3
Swin-S	224^2	50M	8.7G	436.9	83.0
Swin-B	224^2	88M	15.4G	278.1	83.5
Swin-B	384^2	88M	47.0G	84.7	84.5

(b) ImageNet-22K pre-trained models

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
R-101x3 [38]	384^2	388M	204.6G	-	84.4
R-152x4 [38]	480^2	937M	840.5G	-	85.4
ViT-B/16 [20]	384^2	86M	55.4G	85.9	84.0
ViT-L/16 [20]	384^2	307M	190.7G	27.3	85.2
Swin-B	224^2	88M	15.4G	278.1	85.2
Swin-B	384^2	88M	47.0G	84.7	86.4
Swin-L	384^2	197M	103.9G	42.1	87.3

Table 1. Comparison of different backbones on ImageNet-1K classification. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

4.2. Object Detection on COCO

Settings Object detection and instance segmentation experiments are conducted on COCO 2017, which contains 118K training, 5K validation and 20K test-dev images. An ablation study is performed using the validation set, and a system-level comparison is reported on test-dev. For the ablation study, we consider four typical object detection frameworks: Cascade Mask R-CNN [29, 6], ATSS [79], RepPoints v2 [12], and Sparse RCNN [56] in mmdetection [10]. For these four frameworks, we utilize the same settings: multi-scale training [8, 56] (resizing the input such that the shorter side is between 480 and 800 while the longer side is at most 1333), AdamW [44] optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs). For system-level comparison, we adopt an improved HTC [9] (denoted as HTC++) with instaboost [22], stronger multi-scale training [7], 6x schedule (72 epochs), soft-NMS [5], and ImageNet-22K pre-trained model as initialization.

We compare our Swin Transformer to standard Con-

(a) Various frameworks							
Method	Backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	#param.	FLOPs	FPS
Cascade	R-50	46.3	64.3	50.5	82M	739G	18.0
Mask R-CNN	Swin-T	**50.5**	**69.3**	**54.9**	86M	745G	15.3
ATSS	R-50	43.5	61.9	47.0	32M	205G	28.3
	Swin-T	**47.2**	**66.5**	**51.3**	36M	215G	22.3
RepPointsV2	R-50	46.5	64.6	50.3	42M	274G	13.6
	Swin-T	**50.0**	**68.5**	**54.2**	45M	283G	12.0
Sparse	R-50	44.5	63.4	48.2	106M	166G	21.0
R-CNN	Swin-T	**47.9**	**67.3**	**52.3**	110M	172G	18.4

(b) Various backbones w. Cascade Mask R-CNN									
	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	param	FLOPs	FPS
DeiT-S[†]	48.0	67.2	51.7	41.4	64.2	44.3	80M	889G	10.4
R50	46.3	64.3	50.5	40.1	61.7	43.4	82M	739G	18.0
Swin-T	**50.5**	**69.3**	**54.9**	**43.7**	**66.6**	**47.1**	86M	745G	15.3
X101-32	48.1	66.5	52.4	41.6	63.9	45.2	101M	819G	12.8
Swin-S	**51.8**	**70.4**	**56.3**	**44.7**	**67.9**	**48.5**	107M	838G	12.0
X101-64	48.3	66.4	52.3	41.7	64.0	45.1	140M	972G	10.4
Swin-B	**51.9**	**70.9**	**56.5**	**45.0**	**68.4**	**48.7**	145M	982G	11.6

(c) System-level Comparison						
Method	mini-val		test-dev		#param.	FLOPs
	AP^{box}	AP^{mask}	AP^{box}	AP^{mask}		
RepPointsV2* [12]	-	-	52.1	-	-	-
GCNet* [7]	51.8	44.7	52.3	45.4	-	1041G
RelationNet++* [13]	-	-	52.7	-	-	-
SpineNet-190 [21]	52.6	-	52.8	-	164M	1885G
ResNeSt-200* [78]	52.5	-	53.3	47.1	-	-
EfficientDet-D7 [59]	54.4	-	55.1	-	77M	410G
DetectoRS* [46]	-	-	55.7	48.5	-	-
YOLOv4 P7* [4]	-	-	55.8	-	-	-
Copy-paste [26]	55.9	47.2	56.0	47.4	185M	1440G
X101-64 (HTC++)	52.3	46.0	-	-	155M	1033G
Swin-B (HTC++)	56.4	49.1	-	-	160M	1043G
Swin-L (HTC++)	57.1	49.5	57.7	50.2	284M	1470G
Swin-L (HTC++)*	**58.0**	**50.4**	**58.7**	**51.1**	284M	-

Table 2. Results on COCO object detection and instance segmentation. [†]denotes that additional decovolution layers are used to produce hierarchical feature maps. * indicates multi-scale testing.

vNets, i.e. ResNe(X)t, and previous Transformer networks, e.g. DeiT. The comparisons are conducted by changing only the backbones with other settings unchanged. Note that while Swin Transformer and ResNe(X)t are directly applicable to all the above frameworks because of their hierarchical feature maps, DeiT only produces a single resolution of feature maps and cannot be directly applied. For fair comparison, we follow [81] to construct hierarchical feature maps for DeiT using deconvolution layers.

Comparison to ResNe(X)t Table 2(a) lists the results of Swin-T and ResNet-50 on the four object detection frameworks. Our Swin-T architecture brings consistent +3.4∼4.2 box AP gains over ResNet-50, with slightly larger model size, FLOPs and latency.

Table 2(b) compares Swin Transformer and ResNe(X)t under different model capacity using Cascade Mask R-CNN. Swin Transformer achieves a high detection accuracy of 51.9 box AP and 45.0 mask AP, which are significant gains of +3.6 box AP and +3.3 mask AP over ResNeXt101-64x4d, which has similar model size, FLOPs and latency. On a higher baseline of 52.3 box AP and 46.0 mask AP using an improved HTC framework, the gains by Swin Transformer are also high, at +4.1 box AP and +3.1 mask AP (see Table 2(c)). Regarding inference speed, while ResNe(X)t is built by highly optimized Cudnn functions, our architecture is implemented with built-in PyTorch functions that are not all well-optimized. A thorough kernel optimization is beyond the scope of this paper.

ADE20K		val	test	#param.	FLOPs	FPS
Method	Backbone	mIoU	score			
DANet [23]	ResNet-101	45.2	-	69M	1119G	15.2
DLab.v3+ [11]	ResNet-101	44.1	-	63M	1021G	16.0
ACNet [24]	ResNet-101	45.9	38.5	-		
DNL [71]	ResNet-101	46.0	56.2	69M	1249G	14.8
OCRNet [73]	ResNet-101	45.3	56.0	56M	923G	19.3
UperNet [69]	ResNet-101	44.9	-	86M	1029G	20.1
OCRNet [73]	HRNet-w48	45.7	-	71M	664G	12.5
DLab.v3+ [11]	ResNeSt-101	46.9	55.1	66M	1051G	11.9
DLab.v3+ [11]	ResNeSt-200	48.4	-	88M	1381G	8.1
SETR [81]	T-Large[‡]	50.3	61.7	308M	-	-
UperNet	DeiT-S[†]	44.0	-	52M	1099G	16.2
UperNet	Swin-T	46.1	-	60M	945G	18.5
UperNet	Swin-S	49.3	-	81M	1038G	15.2
UperNet	Swin-B[‡]	51.6	-	121M	1841G	8.7
UperNet	Swin-L[‡]	**53.5**	**62.8**	234M	3230G	6.2

Table 3. Results of semantic segmentation on the ADE20K val and test set. [†] indicates additional deconvolution layers are used to produce hierarchical feature maps. ‡ indicates that the model is pre-trained on ImageNet-22K.

Comparison to DeiT The performance of DeiT-S using the Cascade Mask R-CNN framework is shown in Table 2(b). The results of Swin-T are +2.5 box AP and +2.3 mask AP higher than DeiT-S with similar model size (86M vs. 80M) and significantly higher inference speed (15.3 FPS vs. 10.4 FPS). The lower inference speed of DeiT is mainly due to its quadratic complexity to input image size.

Comparison to previous state-of-the-art Table 2(c) compares our best results with those of previous state-of-the-art models. Our best model achieves 58.7 box AP and 51.1 mask AP on COCO test-dev, surpassing the previous best results by +2.7 box AP (Copy-paste [26] without external data) and +2.6 mask AP (DetectoRS [46]).

4.3. Semantic Segmentation on ADE20K

Settings ADE20K [83] is a widely-used semantic segmentation dataset, covering a broad range of 150 semantic

	ImageNet		COCO		ADE20k
	top-1	top-5	AP^{box}	AP^{mask}	mIoU
w/o shifting	80.2	95.1	47.7	41.5	43.3
shifted windows	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**
no pos.	80.1	94.9	49.2	42.6	43.8
abs. pos.	80.5	95.2	49.0	42.4	43.2
abs.+rel. pos.	81.3	95.6	50.2	43.4	44.0
rel. pos. w/o app.	79.3	94.7	48.2	41.9	44.1
rel. pos.	**81.3**	**95.6**	**50.5**	**43.7**	**46.1**

Table 4. Ablation study on the *shifted windows* approach and different position embedding methods on three benchmarks, using the Swin-T architecture. w/o shifting: all self-attention modules adopt regular window partitioning, without *shifting*; abs. pos.: absolute position embedding term of ViT; rel. pos.: the default settings with an additional relative position bias term (see Eq. (4)); app.: the first scaled dot-product term in Eq. (4).

categories. It has 25K images in total, with 20K for training, 2K for validation, and another 3K for testing. We utilize UperNet [69] in mmseg [16] as our base framework for its high efficiency. More details are presented in the Appendix.

Results Table 3 lists the mIoU, model size (#param), FLOPs and FPS for different method/backbone pairs. From these results, it can be seen that Swin-S is +5.3 mIoU higher (49.3 vs. 44.0) than DeiT-S with similar computation cost. It is also +4.4 mIoU higher than ResNet-101, and +2.4 mIoU higher than ResNeSt-101 [78]. Our Swin-L model with ImageNet-22K pre-training achieves 53.5 mIoU on the val set, surpassing the previous best model by +3.2 mIoU (50.3 mIoU by SETR [81] which has a larger model size).

4.4. Ablation Study

In this section, we ablate important design elements in the proposed Swin Transformer, using ImageNet-1K image classification, Cascade Mask R-CNN on COCO object detection, and UperNet on ADE20K semantic segmentation.

Shifted windows Ablations of the *shifted window* approach on the three tasks are reported in Table 4. Swin-T with the shifted window partitioning outperforms the counterpart built on a single window partitioning at each stage by +1.1% top-1 accuracy on ImageNet-1K, +2.8 box AP/+2.2 mask AP on COCO, and +2.8 mIoU on ADE20K. The results indicate the effectiveness of using shifted windows to build connections among windows in the preceding layers. The latency overhead by *shifted window* is also small, as shown in Table 5.

Relative position bias Table 4 shows comparisons of different position embedding approaches. Swin-T with relative position bias yields +1.2%/+0.8% top-1 accuracy on ImageNet-1K, +1.3/+1.5 box AP and +1.1/+1.3 mask AP

method	MSA in a stage (ms)				Arch. (FPS)		
	S1	S2	S3	S4	T	S	B
sliding window (naive)	122.5	38.3	12.1	7.6	183	109	77
sliding window (kernel)	7.6	4.7	2.7	1.8	488	283	187
Performer [14]	4.8	2.8	1.8	1.5	638	370	241
window (w/o shifting)	2.8	1.7	1.2	0.9	770	444	280
shifted window (padding)	3.3	2.3	1.9	2.2	670	371	236
shifted window (cyclic)	3.0	1.9	1.3	1.0	755	437	278

Table 5. Real speed of different self-attention computation methods and implementations on a V100 GPU.

on COCO, and +2.3/+2.9 mIoU on ADE20K in relation to those without position encoding and with absolute position embedding, respectively, indicating the effectiveness of the relative position bias. Also note that while the inclusion of absolute position embedding improves image classification accuracy (+0.4%), it harms object detection and semantic segmentation (-0.2 box/mask AP on COCO and -0.6 mIoU on ADE20K).

While the recent ViT/DeiT models abandon translation invariance in image classification even though it has long been shown to be crucial for visual modeling, we find that inductive bias that encourages certain translation invariance is still preferable for general-purpose visual modeling, particularly for the dense prediction tasks of object detection and semantic segmentation.

Different self-attention methods The real speed of different self-attention computation methods and implementations are compared in Table 5. Our cyclic implementation is more hardware efficient than naive padding, particularly for deeper stages. Overall, it brings a 13%, 18% and 18% speed-up on Swin-T, Swin-S and Swin-B, respectively.

The self-attention modules built on the proposed *shifted window* approach are 40.8×/2.5×, 20.2×/2.5×, 9.3×/2.1×, and 7.6×/1.8× more efficient than those of *sliding windows* in naive/kernel implementations on four network stages, respectively. Overall, the Swin Transformer architectures built on *shifted windows* are 4.1/1.5, 4.0/1.5, 3.6/1.5 times faster than variants built on *sliding windows* for Swin-T, Swin-S, and Swin-B, respectively. Table 6 compares their accuracy on the three tasks, showing that they are similarly accurate in visual modeling.

Compared to Performer [14], which is one of the fastest Transformer architectures (see [60]), the proposed *shifted window* based self-attention computation and the overall Swin Transformer architectures are slightly faster (see Table 5), while achieving +2.3% top-1 accuracy compared to Performer on ImageNet-1K using Swin-T (see Table 6).

5. Conclusion

This paper presents Swin Transformer, a new vision Transformer which produces a hierarchical feature repre-

	Backbone	ImageNet top-1	ImageNet top-5	COCO AP^{box}	COCO AP^{mask}	ADE20k mIoU
sliding window	Swin-T	81.4	95.6	50.2	43.5	45.8
Performer [14]	Swin-T	79.0	94.2	-	-	-
shifted window	Swin-T	81.3	95.6	50.5	43.7	46.1

Table 6. Accuracy of Swin Transformer using different methods for self-attention computation on three benchmarks.

sentation and has linear computational complexity with respect to input image size. Swin Transformer achieves the state-of-the-art performance on COCO object detection and ADE20K semantic segmentation, significantly surpassing previous best methods. We hope that Swin Transformer's strong performance on various vision problems will encourage unified modeling of vision and language signals.

As a key element of Swin Transformer, the *shifted window* based self-attention is shown to be effective and efficient on vision problems, and we look forward to investigating its use in natural language processing as well.

Acknowledgement

We thank many colleagues at Microsoft for their help, in particular, Li Dong and Furu Wei for useful discussions; Bin Xiao, Lu Yuan and Lei Zhang for help on datasets.

A1. Detailed Architectures

The detailed architecture specifications are shown in Table 7, where an input image size of 224×224 is assumed for all architectures. "Concat $n \times n$" indicates a concatenation of $n \times n$ neighboring features in a patch. This operation results in a downsampling of the feature map by a rate of n. "96-d" denotes a linear layer with an output dimension of 96. "win. sz. 7×7" indicates a multi-head self-attention module with window size of 7×7.

A2. Detailed Experimental Settings

A2.1. Image classification on ImageNet-1K

The image classification is performed by applying a global average pooling layer on the output feature map of the last stage, followed by a linear classifier. We find this strategy to be as accurate as using an additional `class` token as in ViT [20] and DeiT [63]. In evaluation, the top-1 accuracy using a single crop is reported.

Regular ImageNet-1K training The training settings mostly follow [63]. For all model variants, we adopt a default input image resolution of 224^2. For other resolutions such as 384^2, we fine-tune the models trained at 224^2 resolution, instead of training from scratch, to reduce GPU consumption.

When training from scratch with a 224^2 input, we employ an AdamW [37] optimizer for 300 epochs using a cosine decay learning rate scheduler with 20 epochs of linear warm-up. A batch size of 1024, an initial learning rate of 0.001, a weight decay of 0.05, and gradient clipping with a max norm of 1 are used. We include most of the augmentation and regularization strategies of [63] in training, including RandAugment [17], Mixup [77], Cutmix [75], random erasing [82] and stochastic depth [35], but not repeated augmentation [31] and Exponential Moving Average (EMA) [45] which do not enhance performance. Note that this is contrary to [63] where repeated augmentation is crucial to stabilize the training of ViT. An increasing degree of stochastic depth augmentation is employed for larger models, i.e. $0.2, 0.3, 0.5$ for Swin-T, Swin-S, and Swin-B, respectively.

For fine-tuning on input with larger resolution, we employ an adamW [37] optimizer for 30 epochs with a constant learning rate of 10^{-5}, weight decay of 10^{-8}, and the same data augmentation and regularizations as the first stage except for setting the stochastic depth ratio to 0.1.

ImageNet-22K pre-training We also pre-train on the larger ImageNet-22K dataset, which contains 14.2 million images and 22K classes. The training is done in two stages. For the first stage with 224^2 input, we employ an AdamW optimizer for 90 epochs using a linear decay learning rate scheduler with a 5-epoch linear warm-up. A batch size of 4096, an initial learning rate of 0.001, and a weight decay of 0.01 are used. In the second stage of ImageNet-1K fine-tuning with 224^2/384^2 input, we train the models for 30 epochs with a batch size of 1024, a constant learning rate of 10^{-5}, and a weight decay of 10^{-8}.

A2.2. Object detection on COCO

For an ablation study, we consider four typical object detection frameworks: Cascade Mask R-CNN [29, 6], ATSS [79], RepPoints v2 [12], and Sparse RCNN [56] in mmdetection [10]. For these four frameworks, we utilize the same settings: multi-scale training [8, 56] (resizing the input such that the shorter side is between 480 and 800 while the longer side is at most 1333), AdamW [44] optimizer (initial learning rate of 0.0001, weight decay of 0.05, and batch size of 16), and 3x schedule (36 epochs with the learning rate decayed by 10× at epochs 27 and 33).

For system-level comparison, we adopt an improved HTC [9] (denoted as HTC++) with instaboost [22], stronger multi-scale training [7] (resizing the input such that the shorter side is between 400 and 1400 while the longer side is at most 1600), 6x schedule (72 epochs with the learning rate decayed at epochs 63 and 69 by a factor of 0.1), soft-NMS [5], and an extra global self-attention layer appended at the output of last stage and ImageNet-22K pre-trained

	downsp. rate (output size)	Swin-T	Swin-S	Swin-B	Swin-L
stage 1	4× (56×56)	concat 4×4, 96-d, LN	concat 4×4, 96-d, LN	concat 4×4, 128-d, LN	concat 4×4, 192-d, LN
		$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 96\text{, head } 3\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 96\text{, head } 3\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 128\text{, head } 4\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$
stage 2	8× (28×28)	concat 2×2, 192-d , LN	concat 2×2, 192-d , LN	concat 2×2, 256-d , LN	concat 2×2, 384-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 192\text{, head } 6\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 256\text{, head } 8\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 2$
stage 3	16× (14×14)	concat 2×2, 384-d , LN	concat 2×2, 384-d , LN	concat 2×2, 512-d , LN	concat 2×2, 768-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 6$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 384\text{, head } 12\end{bmatrix} \times 18$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 512\text{, head } 16\end{bmatrix} \times 18$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 18$
stage 4	32× (7×7)	concat 2×2, 768-d , LN	concat 2×2, 768-d , LN	concat 2×2, 1024-d , LN	concat 2×2, 1536-d , LN
		$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 768\text{, head } 24\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 1024\text{, head } 32\end{bmatrix} \times 2$	$\begin{bmatrix}\text{win. sz. } 7\times 7, \\ \text{dim } 1536\text{, head } 48\end{bmatrix} \times 2$

Table 7. Detailed architecture specifications.

model as initialization. We adopt stochastic depth with ratio of 0.2 for all Swin Transformer models.

A2.3. Semantic segmentation on ADE20K

ADE20K [83] is a widely-used semantic segmentation dataset, covering a broad range of 150 semantic categories. It has 25K images in total, with 20K for training, 2K for validation, and another 3K for testing. We utilize UperNet [69] in mmsegmentation [16] as our base framework for its high efficiency.

In training, we employ the AdamW [44] optimizer with an initial learning rate of 6×10^{-5}, a weight decay of 0.01, a scheduler that uses linear learning rate decay, and a linear warmup of 1,500 iterations. Models are trained on 8 GPUs with 2 images per GPU for 160K iterations. For augmentations, we adopt the default setting in mmsegmentation of random horizontal flipping, random re-scaling within ratio range [0.5, 2.0] and random photometric distortion. Stochastic depth with ratio of 0.2 is applied for all Swin Transformer models. Swin-T, Swin-S are trained on the standard setting as the previous approaches with an input of 512×512. Swin-B and Swin-L with ‡ indicate that these two models are pre-trained on ImageNet-22K, and trained with the input of 640×640.

In inference, a multi-scale test using resolutions that are [0.5, 0.75, 1.0, 1.25, 1.5, 1.75]× of that in training is employed. When reporting test scores, both the training images and validation images are used for training, following common practice [71].

A3. More Experiments

A3.1. Image classification with different input size

Table 8 lists the performance of Swin Transformers with different input image sizes from 224^2 to 384^2. In general, a larger input resolution leads to better top-1 accuracy but with slower inference speed.

	Swin-T		Swin-S		Swin-B	
input size	top-1 acc	throughput (image / s)	top-1 acc	throughput (image / s)	top-1 acc	throughput (image / s)
224^2	81.3	755.2	83.0	436.9	83.3	278.1
256^2	81.6	580.9	83.4	336.7	83.7	208.1
320^2	82.1	342.0	83.7	198.2	84.0	132.0
384^2	82.2	219.5	83.9	127.6	84.5	84.7

Table 8. Swin Transformers with different input image size on ImageNet-1K classification.

Backbone	Optimizer	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R50	SGD	45.0	62.9	48.8	38.5	59.9	41.4
	AdamW	46.3	64.3	50.5	40.1	61.7	43.4
X101-32x4d	SGD	47.8	65.9	51.9	40.4	62.9	43.5
	AdamW	48.1	66.5	52.4	41.6	63.9	45.2
X101-64x4d	SGD	48.8	66.9	53.0	41.4	63.9	44.7
	AdamW	48.3	66.4	52.3	41.7	64.0	45.1

Table 9. Comparison of the SGD and AdamW optimizers for ResNe(X)t backbones on COCO object detection using the Cascade Mask R-CNN framework.

A3.2. Different Optimizers for ResNe(X)t on COCO

Table 9 compares the AdamW and SGD optimizers of the ResNe(X)t backbones on COCO object detection. The Cascade Mask R-CNN framework is used in this comparison. While SGD is used as a default optimizer for Cascade Mask R-CNN framework, we generally observe improved accuracy by replacing it with an AdamW optimizer, particularly for smaller backbones. We thus use AdamW for ResNe(X)t backbones when compared to the proposed Swin Transformer architectures.

A3.3. Swin MLP-Mixer

We apply the proposed hierarchical design and the shifted window approach to the MLP-Mixer architectures [61], referred to as Swin-Mixer. Table 10 shows the performance of Swin-Mixer compared to the original MLP-Mixer architectures MLP-Mixer [61] and a follow-up ap-

method	image size	#param.	FLOPs	throughput (image / s)	ImageNet top-1 acc.
MLP-Mixer-B/16 [61]	224^2	59M	12.7G	-	76.4
ResMLP-S24 [62]	224^2	30M	6.0G	715	79.4
ResMLP-B24 [62]	224^2	116M	23.0G	231	81.0
Swin-T/D24 (Transformer)	256^2	28M	5.9G	563	81.6
Swin-Mixer-T/D24	256^2	20M	4.0G	807	79.4
Swin-Mixer-T/D12	256^2	21M	4.0G	792	79.6
Swin-Mixer-T/D6	256^2	23M	4.0G	766	79.7
Swin-Mixer-B/D24 (no shift)	224^2	61M	10.4G	409	80.3
Swin-Mixer-B/D24	224^2	61M	10.4G	409	81.3

Table 10. Performance of Swin MLP-Mixer on ImageNet-1K classification. D indictes the number of channels per head. Throughput is measured using the GitHub repository of [68] and a V100 GPU, following [63].

proach, ResMLP [61]. Swin-Mixer performs significantly better than MLP-Mixer (81.3% vs. 76.4%) using slightly smaller computation budget (10.4G vs. 12.7G). It also has better speed accuracy trade-off compared to ResMLP [62]. These results indicate the proposed hierarchical design and the shifted window approach are generalizable.

References

[1] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *International Conference on Machine Learning*, pages 642–652. PMLR, 2020. 5

[2] Josh Beal, Eric Kim, Eric Tzeng, Dong Huk Park, Andrew Zhai, and Dmitry Kislyuk. Toward transformer-based object detection. *arXiv preprint arXiv:2012.09958*, 2020. 3

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks, 2020. 3

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 7

[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 (ICCV)*, Oct 2017. 6, 9

[6] 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. 6, 9

[7] 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 (ICCV) Workshops*, Oct 2019. 3, 6, 7, 9

[8] 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. 3, 6, 9

[9] 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. 6, 9

[10] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6, 9

[11] 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. 7

[12] Yihong Chen, Zheng Zhang, Yue Cao, Liwei Wang, Stephen Lin, and Han Hu. Reppoints v2: Verification meets regression for object detection. In *NeurIPS*, 2020. 6, 7, 9

[13] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. In *NeurIPS*, 2020. 3, 7

[14] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. 8, 9

[15] Xiangxiang Chu, Bo Zhang, Zhi Tian, Xiaolin Wei, and Huaxia Xia. Do we really need explicit position encodings for vision transformers? *arXiv preprint arXiv:2102.10882*, 2021. 3

[16] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 8, 10

[17] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 9

[18] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017. 1, 3

[19] 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. 5

[20] 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. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4, 5, 6, 9

[21] 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. 7

[22] 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. 6, 9

[23] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019. 3, 7

[24] 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. 7

[25] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3):121–136, 1975. 3

[26] 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. 2, 7

[27] Jiayuan Gu, Han Hu, Liwei Wang, Yichen Wei, and Jifeng Dai. Learning region features for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 3

[28] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 3

[29] 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. 6, 9

[30] 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, 4

[31] 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. 6, 9

[32] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018. 3, 5

[33] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3464–3473, October 2019. 2, 3, 5

[34] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1, 2

[35] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 9

[36] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962. 3

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 9

[38] 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(2):8, 2019. 6

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2

[40] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 2

[41] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999. 3

[42] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[43] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[44] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 6, 9, 10

[45] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 6, 9

[46] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020. 2, 7

[47] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[48] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 6

[49] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. 5

[50] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 2, 3

[51] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2

[52] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, May 2015. 2, 4

[53] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018. 2

[54] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2

[55] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 3

[56] 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. 3, 6, 9

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[58] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019. 3, 6

[59] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020. 7

[60] 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. In *International Conference on Learning Representations*, 2021. 8

[61] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021. 2, 10, 11

[62] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou. Resmlp: Feedforward networks for image classification with data-efficient training, 2021. 11

[63] 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, 3, 5, 6, 9, 11

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017. 1, 2, 4

[65] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 3

[66] 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. 3

[67] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018*, 2018. 3

[68] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6, 11

[69] 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. 7, 8, 10

[70] 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. 1, 2, 3

[71] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, 2020. 3, 7, 10

[72] 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. 3

[73] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In

16th European Conference Computer Vision (ECCV 2020), August 2020. 7

[74] Yuhui Yuan and Jingdong Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018. 3

[75] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 9

[76] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 1

[77] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 9

[78] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020. 7, 8

[79] 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. 6, 9

[80] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10076–10085, 2020. 3

[81] 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. 2, 3, 7, 8

[82] 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. 9

[83] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal on Computer Vision*, 2018. 5, 7, 10

[84] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 1, 3

[85] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable {detr}: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2021. 3