CVPR 2022 Submission #7254. CONFIDENTIAL REVIEW COPY. DO NOT DISTRIBUTE.

Beyond Fixation: Dynamic Window Visual Transformer

Anonymous CVPR submission

Paper ID 7254

Abstract

Recently, a surge of interest in visual transformers is to reduce the computational cost by limiting the calculation of self-attention to a local window. Most current work uses a fixed single-scale window for modeling by default, ignoring the impact of window size on model performance. However, this may limit the modeling potential of these window-based models for multi-scale information. In this paper, we propose a novel method, named Dynamic Window Vision Transformer (DW-ViT). To the best of our knowledge, we are the first to use dynamic multi-scale windows to explore the upper limit of the effect of window settings on model performance. In DW-ViT, multi-scale information is obtained by assigning windows of different sizes to different head groups of window multi-head self-attention. Then, the information is dynamically fused by assigning different weights to the multi-scale window branches. We conducted a detailed performance evaluation on three datasets, ImageNet-1K, ADE20K, and COCO. Compared with related state-of-the-art (SoTA) methods, DW-ViT obtains the best performance. Specifically, compared with the current SoTA Swin Transformers [30], DW-ViT has achieved consistent and substantial improvements on all three datasets with similar parameters and computational costs. In addition, DW-ViT exhibits good scalability and can be easily inserted into any window-based visual transformers.

1. Introduction

In computer vision (CV) tasks, the visual transformer represented by Vision Transformer (ViT) [12] has shown great potential. These methods have achieved impressive performance on tasks such as image classification [36, 48], semantic segmentation [29, 50] and object detection [30, 54, 56].

In ViT, the complexity of the self-attention operation is proportional to the square of the number of image patches. This is unfriendly to most tasks in the CV field. Swin [30] thus proposed to limit the calculation of self-attention to a local window to reduce the computational complexity and achieve some promising results. This local window self-attention quickly attracted a significant amount of attention [7, 27, 49]. However, most of these methods [7, 27, 49] use a fixed single-scale window (*e.g.*, $win = 7$) by default. The following questions accordingly arise: *Is this window size optimal? Does a bigger window entail better performance? Is a multi-scale window more advantageous than a single-scale window? Furthermore, will dynamic multi-scale windows yield better results?* To answer these questions, we evaluate the impact of window sizes on the model performance. In Fig. 1, we report the change curve (l) of top-1 accuracy and FLOPs (G) of Swin-T [30] under four single-scale windows ($win \in [7, 14, 21, 23]$) on ImageNet-1K [10]. In Swin [30], the window size has a very small effect on the amount of model parameters.

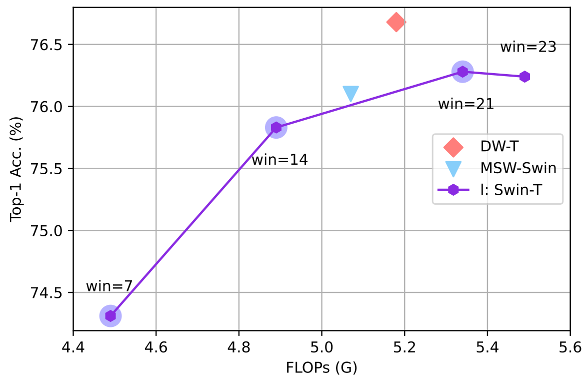

Figure 1. Performance comparison of DW-ViT, Swin [30] and Swin with multi-scale window (MSW-Swin) on ImageNet-1K [10] as the window size increases. We use a purple broken line (l) to indicate the performance and FLOPs changes of Swin-T [30] with a single-scale window ($win \in [7, 14, 21, 23]$). The multi-scale windows used by MSW-Swin and DW-T are all set to $[7, 14, 21]$.

As shown in Fig. 1, as the window size increases, the performance of the model is found to be significantly improved, but this is not absolutely monotonous. For example, when the window size is increased from 21 to 23, the performance of the model hardly improves or even drops. Therefore, it is not feasible to simply increase the window to improve the performance of the model. In addition, it is difficult to choose the best window size from multiple alter-

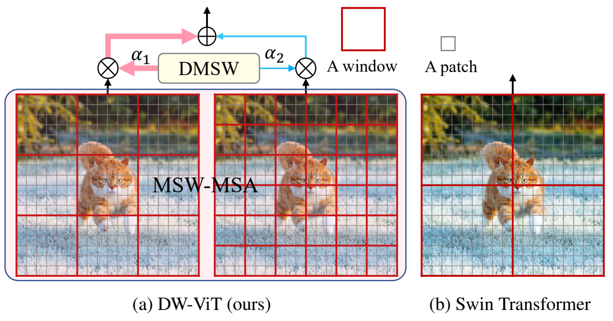

Figure 2. Comparison of DW-ViT's multi-scale window (*e.g.*, $win_1 = 6$ and $win_2 = 3$) and Swin-based single-scale window (*e.g.*, $win = 9$). The number of patches in the local window is $win \times win$. A dynamic multi-scale window (DMSW) is a dynamic adaptive window module we designed for multi-scale window multi-head self-attention (MSW-MSA). α is a learnable parameter of the DMSW module. α_1 and α_2 are a possible weight distribution scheme of DMSW.

native window sizes. And the optimal window settings of different layers may also be different. A natural idea is to mix information from windows of different scales for prediction tasks. Based on this idea, we design a multi-scale window multi-head self-attention (MSW-MSA) mechanism for the window-based ViT. In Fig. 1, as shown in the results of Swin-T with MSW (MSW-Swin) and Swin-T with a single-scale window, simply introducing the MSW mechanism for the W-MSA of the transformer cannot further effectively improve the performance of the model. For example, the performance of MSW-Swin ($win = [7, 14, 21]$) is lower than that of Swin-T with single-scale windows when $win = 21$. It may be caused by suboptimal window settings that impaired the performance of the model. This shows that it may require more effort to protect ViT with MSW from suboptimal window settings while retaining the advantages of multi-scale windows. On the other hand, the dynamic neural network [16] has been favored by a large number of researchers because of its ability to adjust the structure and parameters of the model adaptively according to the input. Moreover, the dynamic network has been successfully applied in CNN [26, 39, 42, 43, 52, 61] and ViT [4, 49, 54].

Based on the above observations, in this paper, we propose a novel method, named Dynamic Window Vision Transformer (DW-ViT). As far as we know, it is the first method to use dynamic multi-scale windows to explore the upper limit of the impact of window settings on model performance. In DW-ViT, we first obtain multi-scale information by assigning different scale windows to different head groups of multi-head self-attention in transformer. Then, we realize the dynamic fusion of information by assigning weights to the multi-scale window branches. In Fig. 2, we present a comparison of DW-ViT's multi-scale window and single-scale window approaches based on Swin [30] class methods. More specifically, in DW-ViT, MSW-MSA is re-

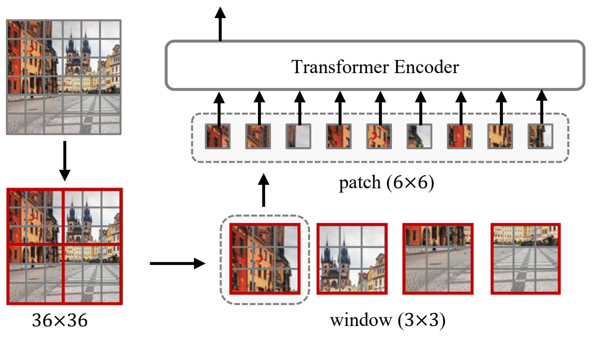

Figure 3. In the visual transformer, a schematic diagram of the window self-attention calculation process. Assume that the number of pixels in the input image is $H \times W$ (*e.g.* 36×36). The image is first split into $\lceil \frac{H}{p} \rceil \times \lceil \frac{W}{p} \rceil$ fixed-size patches (*e.g.* $p = 6$), and then the self-attention calculation is limited to a fixed-size window (*i.e.* each window has $M \times M$ patches, *e.g.* $M = win = 3$). For simplicity, patch and position embeddings are omitted here.

sponsible for the extraction of multi-scale window information, while DMSW is responsible for the dynamic enhancement of these multi-scale information. Through the above two parts, DW-ViT can improve the model's multi-scale information modeling capabilities dynamically while ensuring relatively low computational complexity. Our main contributions can be summarized as follows:

- The recently popular window-based ViT mostly ignores the influence of window size on model performance. This severely limits the upper limit of the model's performance. As far as we know, we are the first to challenge this problem.
- We propose a novel plug-and-play module with a dynamic multi-scale window for multi-head self-attention in transformer. DW-ViT is superior to all other ViTs that use the same single-scale window and can be easily embedded into any window-based ViT.
- Compared with the state-of-the-art methods, DW-ViT achieves the best performance on multiple CV tasks with similar parameters and FLOPs.

2. Related Works

Window self-attention. In the ViT context, standard self-attention splits each image into fixed-size patches [4, 12, 45, 48, 50]. These patches are expanded as a sequence of tokens, which are then fed to the transformer encoder after being encoded. The calculation amount of this standard self-attention is still huge. Subsequent work [21, 48, 50] has continued to try to reduce the computational complexity of standard self-attention. In particular, Swin [30] proposes to limit the calculation of self-attention to a local window. This window self-attention strategy reduces the computational complexity of MSA from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$ (here N

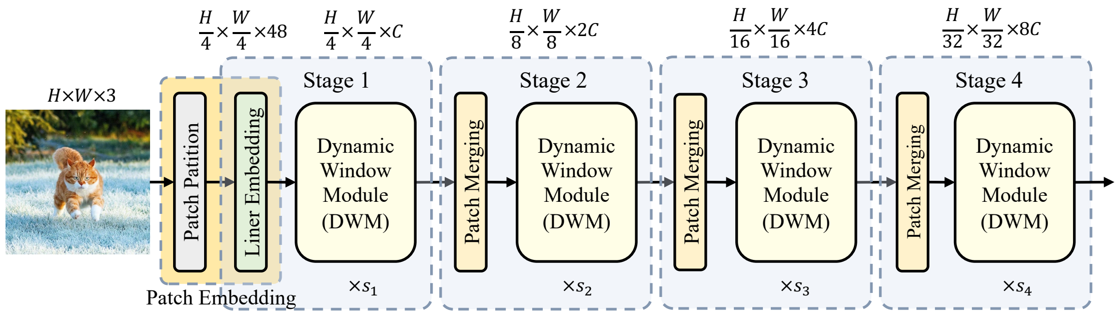
$H \times W \times 3$
$\frac{H}{4} \times \frac{W}{4} \times 48$
$\frac{H}{4} \times \frac{W}{4} \times C$
$\frac{H}{8} \times \frac{W}{8} \times 2C$
$\frac{H}{16} \times \frac{W}{16} \times 4C$
$\frac{H}{32} \times \frac{W}{32} \times 8C$
Stage 1
Stage 2
Stage 3
Stage 4
Patch Partition
Liner Embedding
Patch Embedding
Dynamic Window Module (DWM)
Patch Merging
$\times s_1$
$\times s_2$
$\times s_3$
$\times s_4$

Figure 4. The architecture of the Dynamic Window Vision Transformer (DW-ViT).

is the number of image patches). The schematic diagram of the window-based self-attention calculation process in ViT is shown in Fig. 3. This window self-attention mechanism quickly attracted the attention of a large number of researchers [7, 49, 54]. However, these works all use a fixed single-scale window. They ignored the impact of window size on model performance. This may limit the upper limit of the impact of window configuration on model performance. In Fig. 1, the performance comparison of Swin [30] under different single-scale windows just verifies this idea. Based on the above observations, we filled this gap and explored in detail the effect of window size on model performance, which is a supplement to the above work.

Multi-scale information in ViT. Multi-scale information has been successfully applied in the field of convolution. To obtain more comprehensive information, the model not only needs small-scale information but also large-scale information. For example, Inception [40, 41], Timeception [22], MixConv [44] and SKNet [26], among others, obtain multi-scale information by using different sizes of convolution kernels. Recently, due to the popularity of ViT in the CV field, many researchers have attempted to introduce multi-scale information into ViT. For example, P2T [50] introduces pyramid pooling into the self-attention of the transformer. Similarly, Focal self-attention [54] also incorporates multi-scale information into the calculation of each self-attention. More directly, CrossVit [4] has designed a two-branch transformer encoder with image tokens of different sizes. All of these improve the model's ability to model multi-scale information to varying degrees. However, the above-mentioned method either has a large amount of calculation due to the global self-attention, or it is difficult to expand due to the complex design. In our work, we design a multi-scale window mechanism for MSA to enhance its modeling capabilities in the context of multi-scale information. This MSW-MSA strategy applies to most types of W-MSA computing and exhibits good expansion.

Dynamic multi-branch network. Recently, dynamic networks [16] are popular because they can flexibly adjust the structure and parameters of the network according to the input and have better adaptive capabilities. In a dynamic multi-branch network, a common strategy is to assign corresponding weights to different branches according to their importance to achieve a large-capacity, more versatile, and flexible network structure. For example, early works on this topic [13, 23] used real-valued weights to dynamically rescale the representations obtained from different experts. In addition, SKNet [26] and ResNeSt [58] propose a simple split-attention mechanism that dynamically adjusts the weight of the information obtained by different convolution kernels or branches. This strategy can obtain dynamic feature representations for different samples with a small computational cost, thereby improving the model's expressive ability. In our work, the proposed multi-scale window self-attention module has a natural affinity with the above-mentioned dynamic multi-branch network. Accordingly, we propose a dynamic multi-scale window (DMSW) module for MSW-MSA. This DMSW strategy enables DW-ViT to integrate information from windows of different scales in a dynamic manner so that the model can obtain better expressive capabilities.

3. Method

3.1. Overall Architecture

To facilitate proper comparison while maintaining its high-resolution task processing capabilities, DW-ViT follows the architectural design outlined in [30, 48, 59]. Fig. 4 presents the overall architecture of DW-ViT. The model comprises four stages. To generate hierarchical feature representation, the i-th stage consists of a feature compression layer and s_i Dynamic Window Module (DWM) transformer layers. More specifically, in Stage 1, similar to the ViT [12, 30], the RGB image is split into non-overlapping patches (the patch size is set to 4×4; that is, the compression ratio in the spatial dimension is 4). The original RGB pixel value of each patch is concatenated (*i.e.* after patch concatenation, the dimension is $4 \times 4 \times 3 = 48$) and projected to an arbitrary dimension (denoted as C) through a linear embedding layer. The feature dimension of the corresponding patch embedding layer output is $\frac{H}{4} \times \frac{W}{4} \times C$. These generated patch tokens are then used as the input of the DWM transformer layers, and the number (*i.e.* $\frac{H}{4} \times \frac{W}{4}$) of tokens remains unchanged during this process. Similarly, Stages 2–4 uses a similar structure. The difference is that the feature compression ratio of the patch merging layer in each stage is 2, while the number of channels is doubled. That is, the resolutions of the output features for Stages 2–4 are $\frac{H}{8} \times \frac{W}{8}$, $\frac{H}{16} \times \frac{W}{16}$, and $\frac{H}{32} \times \frac{W}{32}$, and the corresponding channel dimensions are $2C$, $4C$, and $8C$, respectively. The combination of output features at different stages can be used as the input of task networks such as classification, segmentation, and detection.

3.2. Dynamic Window Module

As shown in Fig. 5, the DWM we designed comprises two main parts: a multi-scale window multi-head self-attention module (MSW-MSA) and a dynamic multi-scale window module (DMSW). The former is responsible for the capture of multi-scale window information, while the latter is responsible for the dynamic adaptive weighting of this information.

3.2.1 Multi-Scale Window Multi-head Self-Attention

Fig. 5 (left) presents an architecture diagram of MSW-MSA with h heads and n_{win} scale windows. Here we take $h = 6$ and $n_{win} = 3$ as an example. The multi-head h of MSA is evenly divided into n_{win} groups, which perform multi-head self-attention at different scales window to capture multi-scale window information. A group of windows here can be set to Win $= \{win_i, i = 1, ..., n_{win}\}$. Specifically, assume the input feature map $\boldsymbol{x} \in \mathbb{R}^{H\times W\times C}$; we thus have the following output of MSW-MSA:

$$\begin{aligned} \boldsymbol{y}_{\text{MSW-MSA}} &= \text{MSW-MSA}(\boldsymbol{x}) \\ &= \text{Concat}(\{\text{W-MSA}_{win_i}(\hat{\boldsymbol{y}}_i)\}), \\ \hat{\boldsymbol{y}}_i &= \text{Split}_i(\hat{\boldsymbol{x}}) \in \mathbb{R}^{\frac{h}{n_{win}}\times H\times W\times \frac{C}{h}}, i = 1, ..., n_{win}, \\ \hat{\boldsymbol{x}} &= \text{Reshape}(\boldsymbol{x}) \in \mathbb{R}^{h\times H\times W\times \frac{C}{h}}, \end{aligned} \tag{1}$$

where the i-th branch $\hat{\boldsymbol{y}}_i$ is divided into $\lceil \frac{H}{win_i} \rceil \times \lceil \frac{W}{win_i} \rceil$ windows in the spatial dimension. Each window is expanded into a token sequence of length $win_i \times win_i$ and used as the input of the i-th branch W-MSA$_{win_i}$ of MSW-MSA. The structure of W-MSA is illustrated in Fig. 3. The output of W-MSA$_{win_i}$ is reconstructed as $H \times W$ in the spatial dimension, and the final output dimension is $H \times W \times \frac{C}{n_{win}}$. The outputs of these branches are concatenated in the channel dimension and used as the output of the entire MSW-MSA module.

3.2.2 Dynamic Multi-Scale Window

The output $\boldsymbol{y}_{\text{MSW-MSA}} \in \mathbb{R}^{H\times W\times C}$ of the multi-branch structure MSW-MSA can naturally be used as the input of DMSW. $\boldsymbol{y}_{\text{MSW-MSA}} = \text{Concat}(\{\text{W-MSA}_{win_i}(*), i = 1, ..., n_{win}\})$ retains the multi-scale information of window groups of different scales in the channel dimension. To this end, we designed an dynamic multi-scale window information weighting module DMSW for MSW-MSA.

In more detail, DMSW uses the integrated information of all branches to generate corresponding weights for each branch, then integrates the information of different branches via weighting. The DMSW structure diagram is presented on the right of Fig. 5. This process is divided into two main steps: *Fuse* and *Select*. The former is responsible for integrating the information of all branches, while the latter generates corresponding weights for each branch based on the global information and completes the fusion of branch information. Specifically, the details of these two parts are as follows:

***Fuse*:** It mainly consists of a pooling layer F_{gp} and two pairs of linear mapping layers F_{fc} and activation layers F_a. The calculation process is as follows:

$$\begin{aligned} \boldsymbol{y}_{\text{Fuse}} &= \delta_2(F_{fc_2}(F_{gp}(\delta_1(\hat{\boldsymbol{y}}_{\text{Fuse}})))), \\ \hat{\boldsymbol{y}}_{\text{Fuse}} &= F_{fc_1}(\boldsymbol{y}_{\text{MSW-MSA}}), \end{aligned} \tag{2}$$

where $F_a = \delta$ is the GELU [19] function. The specific dimension setting is presented in Fig. 5 (right), where $\boldsymbol{y}_{\text{Fuse}} \in \mathbb{R}^{1\times 1\times C'}$ and C' is set to $\frac{C}{2n_{win}}$.

***Select*:** It consists of two parts. The first part is composed of a set of linear mapping layers $F_{fc} = \{F_{fc_i}, i = 1, 2, ..., n_{win}\}$ and a softmax layer to generate corresponding weights for each branch, while the second contains two linear mapping layers to restore the channel dimension of the fused features. The specific calculation process is as follows:

$$\begin{aligned} \boldsymbol{y}_{\text{Select}} &= F_{fc_2}(F_{fc_1}(\sum_{i}^{n_{win}} \alpha_i \times \text{W-MSA}_{win_i}(\hat{\boldsymbol{y}}_i))), \\ \alpha_i &= \frac{e^{F_{fc_i}(\boldsymbol{y}_{\text{Fuse}})}}{\sum_i^{n_{win}} e^{F_{fc_i}(\boldsymbol{y}_{\text{Fuse}})}}, i = 1, 2, ..., n_{win}, \end{aligned} \tag{3}$$

where $\alpha_i \in \mathbb{R}^{1\times 1\times \frac{C}{n_{win}}}$. The DMSW module output is as follows:

$$\boldsymbol{y}_{\text{DMSW}} = \boldsymbol{y}_{\text{Select}} + \hat{\boldsymbol{y}}_{\text{Fuse}}. \tag{4}$$

Moreover, $\boldsymbol{y}_{\text{DMSW}} \in \mathbb{R}^{H\times W\times C}$ is also the output of the entire DWM.

3.3. Dynamic Window Block

The DW block is constructed by replacing the standard MSA module in the Transformer block with DWM. In addition, because DWM is designed for multi-scale information, it does not specifically design for cross-window information exchange. In the interests of simplicity, following the design presented in [30], we retain the Swin's [30]

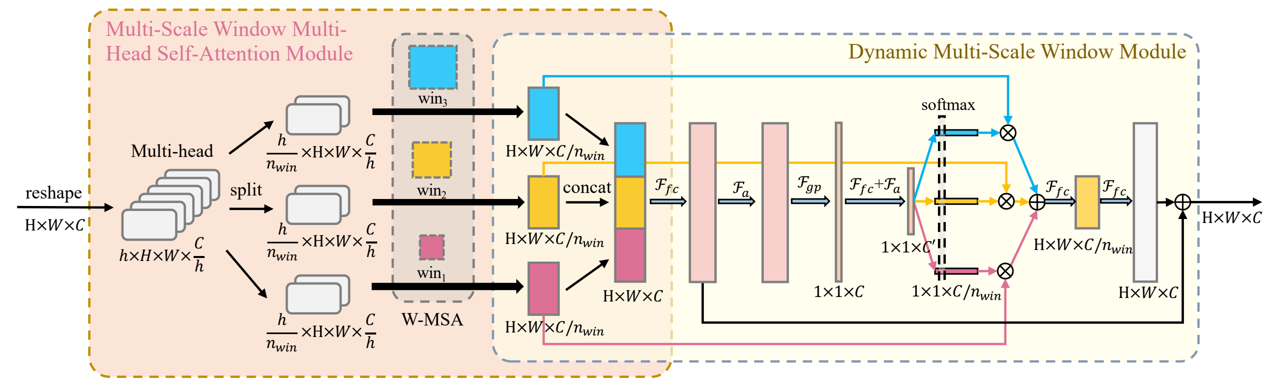

Figure 5. Dynamic Window Module (DWM). DWM has two main parts: Multi-Scale Window Multi-Head Self-Attention Module (MSW-MSA) and Dynamic Multi-Scale Window Module (DMSW).

shifted window strategy. DWM with shifted window strategy is defined as a dynamic shifted window (DSW) block. Each DWM (or DSW) block consists of two LayerNorm (LN) layers and a two-layer MLP with GELU nonlinearity. DSW achieves cross-window information exchange by moving the feature $\lfloor \frac{win}{2} \rfloor$ patches to the upper left in the spatial dimension. When the feature is reconstructed, it moves $\lfloor \frac{win}{2} \rfloor$ patches to the lower right to restore the spatial position of the feature. Alternate stacking of DWM and DSW is used to avoid a decline in information exchange. Specifically, two successive DWM blocks are calculated as follows:

$$\begin{aligned}
\hat{\boldsymbol{z}}^{l} &= \text{DWM}(\text{LN}(\boldsymbol{z}^{l-1})) + \boldsymbol{z}^{l-1}, \\
\boldsymbol{z}^{l} &= \text{MLP}(\text{LN}(\hat{\boldsymbol{z}}^{l})) + \hat{\boldsymbol{z}}^{l}, \\
\hat{\boldsymbol{z}}^{l+1} &= \text{DSW}(\text{LN}(\boldsymbol{z}^{l})) + \boldsymbol{z}^{l}, \\
\boldsymbol{z}^{l+1} &= \text{MLP}(\text{LN}(\hat{\boldsymbol{z}}^{l+1})) + \hat{\boldsymbol{z}}^{l+1},
\end{aligned} \tag{5}$$

where $\hat{\boldsymbol{z}}^{l}$ and $\boldsymbol{z}^{l}$ respectively define the output of the DWM (DSW) module and MLP module in the l-th block.

Position encoding. For a local window with $M \times M$ patches, following [1, 30, 35], we added a set of relative position bias $B = \{B_i \in \mathbb{R}^{M_i^2 \times M_i^2}, i = 1, 2, ..., n_{win}\}$ to the similarity calculation of each head of DWM self-attention. For the W-MSA$_{win_i}$ of the i-th scale local window, we have the window self-attention calculation of Q_i as follows:

$$\text{Attention}(Q_i, K_i, V_i) = \text{SoftMax}(\frac{Q_i K_i^T}{\sqrt{d}} + B_i)V_i, \tag{6}$$

where $Q_i, K_i, V_i \in \mathbb{R}^{M_i^2 \times d}$ are *query*, *key*, and *value* matrices, while M_i^2 is the number of patches in the i-th scale window, and d is the Q_i/K_i dimension. In addition, we parameterized a bias matrix set $\hat{B} = \{\hat{B}_i, i = 1, ..., n_{win}\}$. Specifically, for $\hat{B}_i$, because the relative position on each axis lies in the range of $[-M_i + 1, M_i - 1]$, a small-sized bias matrix $\hat{B}_i \in \mathbb{R}^{(2M_i-1)\times(2M_i-1)}$ is parameterized, and the values in B_i are taken from $\hat{B}_i$.

3.4. Model Configuration

To facilitate fair comparison, following [30], we set the two configuration models as DW-T and DW-B. Their configuration details are summarized in Tab. 1. In particular, according to the results in Fig. 1 and the size of the output features in each stage on ImageNet [10], for the DW-T with three heads in the first stage, we set $\text{Win}_1 = [7, 14, 21]$. For Stages 2–4, we adjust the window according to the size of the output feature of each stage (when the size of the window and the output feature are equal, the standard self-attention is calculated at this time). Similarly, for DW-B, $\text{Win}_1 = [7, 12, 17, 22]$. For all experiments, the query dimension of each head is $d = 32$, while the expansion layer of each MLP is $\alpha = 4$.

3.5. Complexity Analysis

The computational complexity of the DWM block is composed of two main parts: Ω(SMW-MSA) and Ω(DMSW). For an image with $\hbar \times \omega$ patches, their computational complexity is as follows[1]:

$$\Omega(\text{SMW-MSA}) = 4\hbar\omega C^2 + 2\hbar\omega \frac{C}{n_{win}} \sum_{i}^{n_{win}} win_i^2. \tag{7}$$

$$\Omega(\text{DMSW}) = (1 + \hbar\omega(1 + \frac{1}{n_{win}}))\frac{C^2}{n_{win}}. \tag{8}$$

The total computational complexity of DWM is as follows:

$$\begin{aligned}
\Omega(\text{DWM}) =&\Omega(\text{SMW-MSA}) + \Omega(\text{DMSW}) \\
=&(1 + 4n_{win} + \frac{\hbar\omega + n_{win}}{\hbar\omega n_{win}})\frac{\hbar\omega}{n_{win}}C^2 + \\
&2\hbar\omega \frac{C}{n_{win}} \sum_{i}^{n_{win}} win_i^2.
\end{aligned} \tag{9}$$

Since both win_i and n_{win} are constants, the total computational complexity of DWM does not significantly increase. The computational complexity of DWM is still $\mathcal{O}(N)$.

[1]The calculation of SoftMax is ignored here.

	Output Size	Layer Name	DW-T	DW-B
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	Patch Embedding	$p_1 = 4; C_1 = 96$	$p_1 = 4; C_1 = 128$
		DWM	$\begin{bmatrix} \text{Win}_1 = [7, 14, 21] \\ h_1 = 3, C_1 = 96 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Win}_1 = [7, 12, 17, 22] \\ h_1 = 4, C_1 = 128 \end{bmatrix} \times 2$
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	Patch Merging	$p_2 = 2; C_2 = 192$	$p_2 = 2; C_2 = 256$
		DWM	$\begin{bmatrix} \text{Win}_2 = [7, 14, 21] \\ h_2 = 6, C_2 = 192 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Win}_2 = [7, 12, 17, 22] \\ h_2 = 8, C_2 = 256 \end{bmatrix} \times 2$
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	Patch Merging	$p_3 = 2; C_3 = 384$	$p_3 = 2; C_3 = 512$
		DWM	$\begin{bmatrix} \text{Win}_3 = [7, 14, 14] \\ h_3 = 12, C_3 = 384 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{Win}_3 = [7, 12, 14, 14] \\ h_3 = 16, C_3 = 512 \end{bmatrix} \times 18$
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	Patch Merging	$p_4 = 2; C_4 = 768$	$p_4 = 2; C_4 = 1024$
		DWM	$\begin{bmatrix} \text{Win}_4 = [7, 7, 7] \\ h_4 = 24, C_4 = 768 \end{bmatrix} \times 2$	$\begin{bmatrix} \text{Win}_4 = [7, 7, 7, 7] \\ h_4 = 32, C_4 = 1024 \end{bmatrix} \times 2$

Table 1. Configuration details of DW-ViT. Here, $p_i \times p_i$ is the size of the patch in the i-th stage, and is also the downsampling ratio of the feature in the spatial dimension. C_i is the number of feature channels, while Win_i and h_i are the window combination used by the MSW-MSA module and the number of heads used by the MSA in transformer respectively.

4. Experiments

We conduct a performance comparison with the state-of-the-art (SoTA) methods on an upstream task, ImageNet-1K image classification [10], and two downstream tasks: semantic segmentation on ADE20K [60], and object detection and instance segmentation on COCO 2017 [28]. Finally, we ablate the important modules of DW-ViT.

4.1. Image Classification on ImageNet-1K

Experimental Settings We benchmark DW-ViT on ImageNet-1K [10]. ImageNet-1K contains 1.28M training images and 50K test images from 1000 categories. To test the effectiveness of DW-ViT and conduct a fair comparison with similar methods [4, 7, 30], we carefully avoid using any tricks that provide unfair advantage [24, 47]. Specifically, following the settings in [7, 30], DW-ViT was trained for 300 epochs with a batch size of 1024 using the AdamW optimizer [31]. The cosine decay learning rate scheduler and 20 epochs of a linear warm-up are used. The initial learning rate and weight decay are set to 0.001 and 0.05, respectively. In training, [46]'s augmentation and regularization strategies are used. Following the settings in [30], the repeated enhancement [20] and EMA [33] strategy are abandoned.

Results Tab. 2 reports the performance comparison of DW-ViT and state-of-the-art methods on ImageNet-1K. Methods of comparison include the classic and the latest ConvNet-based [18, 34, 52] and Transformer-based [4, 30, 49] models. All models are trained and evaluated at 224 × 224 resolution. As shown in Tab. 2, with similar parameters and FLOPs, DW-ViT still has obvious advantages compared with other current state-of-the-art methods. Specifically, compared with Transformer baseline DeiT [45], the performance of DW-T and DW-B are improved by 2.1% and 2.0%, respectively. At the same time, under the same settings, compared with Swin [30], DW-T and DW-B also achieved performance gains of 0.7 and 0.5 points, respectively, with the help of dynamic windows. This shows that DW-ViT as a general visual feature extractor can obtain better feature representation. In addition, it is worth mentioning that as an independent module, DWM can be flexibly embedded in any window-based ViT model [7, 27, 49] like Swin [30] to improve the model's dynamic modeling capabilities for multi-scale information. Compared with these

Method	#param. (M)	Flops (G)	Top-1 (%)
ConvNet			
ResNet50 [18]	26	4.1	76.6
ResNet101 [18]	45	7.9	78.2
X50-32x4d [52]	25	4.3	77.9
X101-32x4d [52]	44	8.0	78.7
RegNetY-4G [34]	21	4.0	80.0
RegNetY-8G [34]	39	8.0	81.7
RegNetY-16G [34]	84	16	82.9
Transformer			
DeiT-Small/16 [45]	22	4.6	79.9
CrossViT-S [4]	27	5.6	81.0
T2T-ViT-14 [56]	22	5.2	81.5
TNT-S [15]	24	5.2	81.3
CoaT Mini [53]	10	6.8	80.8
PVT-Small [48]	25	3.8	79.8
CPVT-GAP [56]	23	4.6	81.5
CrossFormer-S† [49]	28	4.5	81.5
Swin-T [30]	28	4.5	81.3
DW-T	30	5.2	**82.0**
ViT-Base/16 [11]	87	17.6	77.9
DeiT-Base/16 [45]	87	17.6	81.8
T2T-ViT-24 [56]	64	14.1	82.3
CrossViT-B [4]	105	21.2	82.2
TNT-B [15]	66	14.1	82.8
CPVT-B [8]	88	17.6	82.3
PVT-Large [48]	61	9.8	81.7
Swin-B [30]	88	15.4	83.3
DW-B	91	17.0	**83.8**

Table 2. Performance comparison on ImageNet-1K. All models are trained and evaluated at 224×224 resolution. CrossFormer-S† shows the performance in the case of single-scale embedding.

Backbone	Method	#param. (M)	FLOPs (G)	mIoU	+MS
ResNet-101 [18]	DANet [32]	69	1119	45.3	-
ResNet-101	OCRNet [57]	56	923	44.1	-
ResNet-101	DLab.v3+ [6]	63	1021	44.1	-
ResNet-101	ACNet [14]	-	-	45.9	-
ResNet-101	DNL [55]	69	1249	46.0	-
ResNet-101	UperNet [51]	86	1029	44.9	-
HRNet-w48 [37]	DLab.v3+ [6]	71	664	45.7	
ResNeSt-101 [58]	DLab.v3+ [6]	66	1051	46.9	-
ResNeSt-200 [58]	DLab.v3+ [6]	88	1381	48.4	-
PVT-S [48]	S-FPN [25]	28	-	39.8	
PVT-M	S-FPN	48	219	41.6	-
PVT-L	S-FPN	65	283	42.1	-
CAT-S [27]	S-FPN	41	214	42.8	-
CAT-B	S-FPN	55	276	44.9	-
Swin-T [30]	UperNet [51]	60	945	44.5	45.8
Swin-B [30]	UperNet [51]	121	1188	48.1	49.7
DW-T	UperNet [51]	61	953	45.7	46.9
DW-B	UperNet [51]	125	1200	**48.7**	**50.3**

Table 3. Performance comparison on the ADE20K [60] val. The single-scale and multi-scale evaluation results are presented in the last two columns. The FLOPs (G) are calculated at an input resolution of 1024×1024.

ViTs [7, 27, 49] that use a fixed single-scale window, DWM enables DW-ViT to have a larger model capacity and perform better in terms of adaptability and scalability.

4.2. Semantic Segmentation on ADE20K

ADE20K [60] is also a widely used semantic segmentation dataset. It contains 20K training images, 2K verification images, and 3K test images, covering a total of 150 semantic categories. DW-ViT and UperNet [51] in mmsegmentation [9] are used as the backbone and segmentation methods respectively. The pre-trained backbone used is DW-ViT trained on ImageNet-1K. Following the settings in [30], the input size of the image is 512×512, AdamW [31] is used as the optimizer (the initial learning rate is 6×10^{-5}, weight decay is 0.01, and a linear learning rate decay is used), and the model is trained with a batch size of 16 and 160K iterations. For multi-scale evaluation (+MS), the scaling ratio is between 0.5 and 1.75.

The performance comparison between DW-ViT and other methods on ADE20K val is shown in Tab. 3. As shown in Tab. 3, DW-ViT achieves the best performance compared to many state-of-the-art methods. Specifically, under similar FLOPs and parameters, compared with Swin [30], DW-ViT improves the single-scale evaluation by 1.2 and 0.6 points, respectively. Compared with other methods, DW-ViT has also obtained competitive results. Compared with Swin, DW-ViT has a more obvious advantage (*e.g.* $0.7 \rightarrow 1.2$) in ADE20K than in ImageNet. This shows that the dynamic window mechanism of DW-ViT has more obvious advantages in downstream tasks such as more complex image datasets.

4.3. Object Detection and Instance Segmentation on COCO

Further, we benchmark DW-ViT on object detection and instance segmentation with COCO 2017 [28]. COCO contains 118K training, 5K validation, and 20K test images. The pre-trained model used is DW-ViT trained on ImageNet-1K. DW-ViT is used as the visual backbone and is then plugged into a representative object detection framework. We here consider two representative object detection frameworks: Mask R-CNN [17] and Cascade Mask R-CNN [2]. All models are trained on the training images and the results are reported on the validation set. The same settings were used for all frameworks. Specifically, we use multi-scale training [3, 38], the AdamW [31] optimizer (the initial learning rate, weight decay and batch size are 0.0001, 0.05, and 16), and a 3 $\times$ schedule (it has 36 epochs, and the learning rate decays by 10 $\times$ between epochs 27 and 33). It is implemented based on MMDetection [5].

The performance comparison of object detection and instance segmentation on the COCO2017 val dataset is shown in Tab. 4. Compared with other state-of-the-art methods, DW-ViT achieves the best performance in both object detection frameworks. Specifically, compared with the Transformer baseline DeiT-S [45], DW-T is improved by 3.5 points. Compared with Swin [30], DW-ViT has achieved an improvement of more than 0.7 points in object detection and instance segmentation under the two object detection frameworks. At the same time, compared with Swin, the parameters and FOLPs of DW-ViT have not increased significantly, which once again demonstrates the superiority of the dynamic window mechanism. In addition, the results of the two detection frameworks show that DW-ViT can be easily embedded into different frameworks like other backbones.

4.4. Ablation Study

To explore the effects of each component of DW-ViT, we compared the performance of Swin-T with single-scale window, MSW-Swin, and DW-ViT with and without DMSW mechanism. Specifically, we set $epoch = 50$; for all other settings, we adopt the default settings presented Swin [30]. Single-scale windows are taken from $[7, 11, 14, 17, 21, 23]$, and multi-scale windows are set to $[7, 14, 21]$[2]. Their performance on ImageNet-1K [10] are shown in Tab. 5.

In Tab. 5, DMSW shows three states ('1', '-', '✓'). MSW-MSA + '1' refers to removing the dynamic weight generation and directly assigning the same weight ($\frac{1}{3}$) to all branches. MSW-MSA + '-' (MSW-Swin) denotes removing the entire DMSW module, while, MSW-MSA + '✓' means normal DW-T. The performance of MSW-Swin is lower

[2]We adopted the original settings in Swin [30] and modified only the window size. When the window size is larger than the input feature, the global self-attention is performed at this time.

Method	#param. (M)	FLOPs (G)	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
			Mask R-CNN [17]					
ResNet50 [18]	44	260	41.0	61.7	44.9	37.1	58.4	40.1
PVT-Small [48]	44	245	43.0	65.3	46.9	39.9	62.5	42.8
ViL-Small [59]	45	174	43.4	64.9	47.0	39.6	62.1	42.4
Swin-T [30]	48	264	46.0	68.2	50.2	41.6	65.1	44.8
DW-T	49	275	**46.7**	**69.1**	**51.4**	**42.4**	**66.2**	**45.6**
ResNeXt101-64x4d [52]	102	493	44.4	64.9	48.8	39.7	61.9	42.6
PVT-Large [48]	81	364	44.5	66.0	48.3	40.7	63.4	43.7
ViL-Base [59]	76.1	365	45.7	67.2	49.9	41.3	64.4	44.5
Swin-Base [30]	107	496	48.5	69.8	53.2	43.4	66.8	46.9
DW-B	111	505	**49.2**	**70.6**	**54.0**	**44.0**	**68.0**	**47.7**
			Cascade Mask R-CNN [2, 17]					
DeiT-S† [45]	80	889	48.0	67.2	51.7	41.4	64.2	44.3
ResNet50 [18]	82	739	46.3	64.3	50.5	40.1	61.7	43.4
Swin-T [30]	86	745	50.5	69.3	54.9	43.7	66.6	47.1
DW-T	87	754	**51.5**	**70.5**	**55.9**	**44.7**	**67.8**	**48.5**
X101-64 [52]	140	972	48.3	66.4	52.3	41.7	64.0	45.1
Swin-B [30]	145	982	51.9	70.9	56.5	45.0	68.4	48.7
DW-B	149	992	**52.9**	**71.6**	**57.5**	**45.7**	**69.0**	**50.0**

Table 4. Performance comparison of object detection and instance segmentation on the COCO2017 val dataset. Two object detection frameworks are used: Mask R-CNN [17] and Cascade Mask R-CNN [2]. The FLOPs (G) are calculated at an input resolution of 1280×800. † indicates that additional deconvolution layers are used to generate hierarchical features.

Method	Window	#param. (M)	FLOPs (G)	Top-1 (%)
Swin-T	7	28.29	4.49	74.31
	11	28.31	4.69	75.18
	14	28.34	4.89	75.83
	17	28.35	5.06	76.31
	21	28.36	5.34	76.28
	23	28.36	5.49	76.24
DW-T	DMSW			
MSW-MSA ([7, 14, 21])	1	29.05	5.18	73.43
	-	28.33	5.07	76.10
	✓	29.77	5.18	**76.68**

Table 5. Performance comparison of Swin and DW-ViT on ImageNet-1K [10] under different window and module settings.

than that of Swin-T with $win = 21$. This may be due to the sub-optimal window setting that impairs the performance of the model to a certain extent. The performance comparison between DW-T and MSW-MSA + '1' further shows that this dynamic window mechanism achieves a very significant improvement (*i.e.* 3.3%). In addition, with the help of the dynamic window mechanism, the performance of DW-ViT is better than all ViTs that use the same single-scale window. This shows that this dynamic window weighting mechanism does play a very important role in DW-ViT.

5. Conclusion

The current existing improved methods of window-based ViT usually use a fixed single-scale window by default, which most likely limits the model's ability to dynamically model multi-scale information. In this paper, we challenged this problem for the first time and discussed in detail the impact of window size on model performance. Based on our insightful observations on the above issues, we propose a novel dynamic multi-scale window mechanism for W-MSA to obtain the optimal window configuration, thereby enhancing the model's dynamic modeling capabilities for multi-scale information. With the help of the dynamic window mechanism, the performance of DW-ViT is found to be better than all ViTs that use the same single-scale window, with the proposed approach achieving good results on multiple CV tasks. At the same time, DWM has good scalability, and can thus be easily inserted into any window-based ViT as a module.

6. Discussion

Potential negative societal impact: As a general visual feature extractor, DW-ViT has shown good performance on multiple CV tasks. However, due to the domain gap between different tasks, when the model is transferred to other tasks, some fine adjustments may still be needed.

Limitation: These are a few issues that we need to improve in the future: (1) Although DW-ViT has shown good performance on multiple vision tasks. But compared with the single-scale window self-attention mechanism [30], DWM still introduces a small number of additional parameters and calculations. (2) In addition, as far as DWM's dynamic window mechanism is concerned, part of the computational budget is still allocated to suboptimal optional windows. An ideal strategy is to allocate the entire computational budget to the most potential windows at each layer of the network. However, this may require the construction of a huge supernet, which means huge memory and computational costs. But this is still an exciting direction for improvement.

References

[1] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unilmv2: Pseudo-masked language models for unified language model pre-training. In *ICML*, 2020. 5

[2] 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. 7, 8

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 7

[4] Chun-Fu Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *arXiv preprint arXiv:2103.14899*, 2021. 2, 3, 6

[5] 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. 7

[6] 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

[7] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv preprint arXiv:2104.13840*, 1(2):3, 2021. 1, 3, 6, 7

[8] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021. 6

[9] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 7

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 5, 6, 7, 8

[11] 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. 6

[12] 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

[13] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *CoRR*, abs/1312.4314, 2014. 3

[14] 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

[15] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *arXiv preprint arXiv:2103.00112*, 2021. 6

[16] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *arXiv preprint arXiv:2102.04906*, 2021. 2, 3

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

[18] 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. 6, 7, 8

[19] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *ArXiv*, abs/1606.08415, 2016. 4

[20] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8126–8135, 2020. 6

[21] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Ching-Feng Lin. Local relation networks for image recognition. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3463–3472, 2019. 2

[22] Noureldien Hussein, Efstratios Gavves, and Arnold W. M. Smeulders. Timeception for complex action recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 254–263, 2019. 3

[23] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991. 3

[24] Zihang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Xiaojie Jin, Anran Wang, and Jiashi Feng. Token labeling: Training a 85.4% top-1 accuracy vision transformer with 56m parameters on imagenet. *ArXiv*, abs/2104.10858, 2021. 6

[25] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 7

[26] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 510–519, 2019. 2, 3

[27] Hezheng Lin, Xing Cheng, Xiangyu Wu, Fan Yang, Dong Shen, Zhongyuan Wang, Qing Song, and Wei Yuan. Cat:

Cross attention in vision transformer. *arXiv preprint arXiv:2106.05786*, 2021. 1, 6, 7

[28] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 6, 7

[29] Huajun Liu, Fuqiang Liu, Xinyi Fan, and Dong Huang. Polarized self-attention: Towards high-quality pixel-wise regression. *arXiv preprint arXiv:2107.00782*, 2021. 1

[30] 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. 1, 2, 3, 4, 5, 6, 7, 8

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 6, 7

[32] 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. 7

[33] Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *Siam Journal on Control and Optimization*, 30:838–855, 1992. 6

[34] 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

[35] Colin Raffel, Noam M. 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. *ArXiv*, abs/1910.10683, 2020. 5

[36] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16519–16529, 2021. 1

[37] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019. 7

[38] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14454–14463, 2021. 7

[39] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017. 2

[40] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Amir Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 2017. 3

[41] 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. 3

[42] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 2

[43] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *arXiv preprint arXiv:1907.09595*, 2019. 2

[44] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. *ArXiv*, abs/1907.09595, 2019. 3

[45] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2, 6, 7, 8

[46] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 6

[47] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Herv'e J'egou. Going deeper with image transformers. *ArXiv*, abs/2103.17239, 2021. 6

[48] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *arXiv preprint arXiv:2102.12122*, 2021. 1, 2, 3, 6, 7, 8

[49] Wenxiao Wang, Lu Yao, Long Chen, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer based on cross-scale attention. *arXiv preprint arXiv:2108.00154*, 2021. 1, 2, 3, 6, 7

[50] Yu-Huan Wu, Yun Liu, Xin Zhan, and Ming-Ming Cheng. P2t: Pyramid pooling transformer for scene understanding. *arXiv preprint arXiv:2106.12011*, 2021. 1, 2, 3

[51] 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

[52] 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. 2, 6, 8

[53] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Co-scale conv-attentional image transformers. *arXiv preprint arXiv:2104.06399*, 2021. 6

[54] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021. 1, 2, 3

[55] 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. 7

[56] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zihang Jiang, 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. 1, 6

[57] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 173–190. Springer, 2020. 7

[58] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi-Li Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alex Smola. Resnest: Split-attention networks. *ArXiv*, abs/2004.08955, 2020. 3, 7

[59] 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*, abs/2103.15358, 2021. 3, 8

[60] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127:302–321, 2018. 6, 7

[61] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2