Beyond Fixation: Dynamic Window Visual Transformer

Pengzhen Ren1 Changlin Li2 Guangrun Wang3 Yun Xiao1*
Qing Du4* Xiaodan Liang5,6 Xiaojun Chang2,7

1Northwest University, China 2ReLER, AAIL, University of Technology Sydney 3University of Oxford
4South China University of Technology 5Sun Yat-sen University 6PengCheng Laboratory 7RMIT University

pzhren@foxmail.com, yxiao@nwu.edu.cn, duqing@scut.edu.cn, xiaojun.chang@uts.edu.au
{changlinli.ai, wanggrun, xdliang328}@gmail.com

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). The dynamic window strategy proposed by DW-ViT goes beyond the model that employs a fixed single window setting. 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 [31], 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 [37, 49], semantic segmentation [30, 51] and object detection [31, 55, 57].

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 [31] thus proposed to limit the calculation of self-attention to a local window to reduce the computational complexity and achieved some promising results. Local window self-attention quickly attracted a significant amount of attention [7, 28, 50]. However, most of these methods [7, 28, 50] 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 [31] under four single-scale windows (win ∈ [7, 14, 21, 23]) on ImageNet-
A window

Figure 2. Comparison of DW-ViT’s multi-scale window (e.g., \( \text{win}_1 = 6 \) and \( \text{win}_2 = 3 \)) and Swin-based single-scale window (e.g., \( \text{win} = 9 \)). The number of patches in the local window is \( \text{win} \times \text{win} \). A dynamic multi-scale window (DMSW) is a dynamic adaptive window module designed for multi-scale window multi-head self-attention (MSW-MSA). \( \alpha \) is a learnable parameter of the DMSW module. \( \alpha_1 \) and \( \alpha_2 \) are possible weight distribution scheme of DMSW.

1K [10]. In Swin [31], the window size has a very small effect on the amount of model parameters.

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 monotonic. 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 alternative 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 (\( \text{win} = 7, 14, 21 \)) is lower than that of Swin-T with single-scale windows when \( \text{win} = 21 \). It may be caused by suboptimal window settings that impairs 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 [17] 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 [27, 40, 43, 44, 53, 62] and ViT [4, 50, 55].

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 [31] class methods. More specifically, in DW-ViT, MSW-MSA is responsible 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. As shown in Fig. 1, the performance of DW-T with a dynamic window is significantly better than that of Swin-T with a single fixed-scale window, which we call "beyond fixed". 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 [12, 46,
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 [22, 49, 51] has continued to try to reduce the computational complexity of standard self-attention. In particular, Swin [31] proposes to limit the calculation of self-attention to a local window. This window self-attention strategy reduces the computational complexity of MSA from $O(N^2)$ to $O(N)$ (here $N$ 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, 50, 55]. 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 [31] 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 [41, 42], Timeception [23], MixConv [45] and SKNet [27], among others, obtain multi-scale information by using different sizes of convolution kernels. In addition, some works [15, 50] also try to use the output of CNN as the input of ViT to improve the ability of ViT to model local information. In particular, CrossFormer [50] uses multi-scale convolution to provide multi-scale information for the ViT input. Recently, due to the popularity of ViT in the CV field, many researchers have attempted to introduce multi-scale information into ViT. The pyramid structure in CNN is a widely borrowed idea. For example, T2T [57] reduces the length of the token sequence stage by stage by aggregating adjacent patches, while PVT [49] reduces the feature dimension by modifying self-attention. In this way, multi-scale feature information is constructed from the network framework. Further, P2T [51] introduces pyramid pooling into the self-attention of the transformer. Similarly, Focal self-attention [55] 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 [17] 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, 24] used real-valued weights to dynamically rescale the representations obtained from different experts. In addition, SKNet [27] and ResNeSt [59] 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 fol-
allows the architectural design outlined in [31, 49, 60]. 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, 31], the RGB image is split into non-overlapping patches (the patch size is set to $4 \times 4$; 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 $\frac{H}{8} \times \frac{W}{8} \times \frac{16}{16}$ 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 $x \in \mathbb{R}^{H \times W \times C}$; we thus have the following output of MSW-MSA:

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

where the $i$-th branch $\hat{y}_i$ is divided into $\left\lceil \frac{H}{\text{win}_i} \right\rceil \times \left\lceil \frac{W}{\text{win}_i} \right\rceil$ windows in the spatial dimension. Each window is expanded into a token sequence of length $\text{win}_i \times \text{win}_i$ and used as the input of the $i$-th branch W-MSA$_{\text{win}_i}$ of MSW-MSA. The structure of W-MSA is illustrated in Fig. 3. The output of W-MSA$_{\text{win}_i}$ 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 $y_{\text{MSW-MSA}} \in \mathbb{R}^{H \times W \times C}$ of the multi-branch structure MSW-MSA can naturally be used as the input of DMSW. $y_{\text{MSW-MSA}} = \text{Concat}(\{W\text{-MSA}_{\text{win}_i}(\hat{y}_i), i = 1, ..., n_{win}\})$ 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 fully connected layers $F_{fc}$ and activation layers $F_a$. The calculation process is as follows:

$$
\begin{align*}
    y_{\text{Fuse}} &= \delta_2(F_{fc}(F_{gp}(\delta_1(y_{\text{Fuse}})))), \\
    \hat{y}_{\text{Fuse}} &= F_{fc}(y_{\text{MSW-MSA}}),
\end{align*}
$$

where $F_a = \delta$ is the GELU [20] function. The specific dimension setting is presented in Fig. 5 (right), where $y_{\text{Fuse}} \in \mathbb{R}^{1 \times 1 \times C'}$ and $C'$ is set to $\frac{C}{n_{win}}$.

**Select:** It consists of two parts. The first part is composed of a set of fully connected layers $F_a = \{F_{\alpha_i, i=1,2,...,n_{win}}\}$ 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:

$$
\begin{align*}
    y_{\text{Select}} &= F_{fc}(F_{fc}(\sum_{i=1}^{n_{win}} \alpha_i \times W\text{-MSA}_{\text{win}_i}(\hat{y}_i))), \\
    \alpha_i &= \frac{e^{F_{\alpha_i}(y_{\text{Fuse}})}}{\sum_{i=1}^{n_{win}} e^{F_{\alpha_i}(y_{\text{Fuse}})}}, \quad i = 1, 2, ..., n_{win},
\end{align*}
$$

(3)
where $\alpha_i \in \mathbb{R}^{1 \times 1 \times \frac{C}{n_{\text{win}}}}$. The DMSW module output is as follows:

$$y_{\text{DMSW}} = y_{\text{Select}} + y_{\text{Fuse}}.$$  (4)

Moreover, $y_{\text{DMSW}} \in \mathbb{R}^{H \times W \times C}$ 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 [31], we retain the Swin’s [31] 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 non-linearity. DSW achieves cross-window information exchange by moving the feature $\frac{w_{\text{win}}}{2}$ patches to the upper left in the spatial dimension. When the feature is reconstructed, it moves $\frac{w_{\text{win}}}{2}$ 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:

$$\begin{align*}
\hat{z}^l &= \text{DWM}(\text{LN}(z^{l-1})) + z^{l-1}, \\
\hat{z}'^l &= \text{MLP}(\text{LN}(\hat{z}^l)) + \hat{z}'^l, \\
\hat{z}'^{l+1} &= \text{DSW}(\text{LN}(z^{l}')) + z^{l}’, \\
\hat{z}'^{l+1} &= \text{MLP}(\text{LN}(\hat{z}'^{l+1})) + \hat{z}'^{l+1},
\end{align*}$$  (5)

where $\hat{z}^l$ and $z^l$ 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, 31, 36], we added a set of relative position bias $B = \{B_i \in \mathbb{R}^{M^2 \times M^2}, i = 1, 2, ..., n_{\text{win}}\}$ to the similarity calculation of each head of DWM self-attention. For the W-MSA$_{\text{win}_i}$ of the $i$-th scale local window, we have the window self-attention calculation of $Q_i$ as follows:

$$\text{Attention}(Q_i, K_i, V_i) = \text{SoftMax}(\frac{Q_i K_i^T}{\sqrt{d}} + B_i) V_i,$$  (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 $B = \{B_i, i = 1, ..., n_{\text{win}}\}$. Specifically, for $B_i$, because the relative position on each axis lies in the range of $[-M_i+1, M_i-1]$, a small-sized bias matrix $B_i \in \mathbb{R}^{(2M_i-1) \times (2M_i-1)}$ is parameterized, and the values in $B_i$ are taken from $B_i$.

### 3.4. Model Configuration

To facilitate fair comparison, following [31], 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 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, 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: $\Omega$(SMW-MSA) and $\Omega$(DMSW). For an image with $h \times w$ patches, their computational complexity is as follows:

$$\begin{align*}
\Omega(\text{SMW-MSA}) &= 4hwC^2 + 2hw \frac{C}{n_{\text{win}}} \sum_{i} n_{\text{win}_i}^2, \\
\Omega(\text{DMSW}) &= (1 + \frac{1}{n_{\text{win}}}) \frac{C^2}{n_{\text{win}}}. 
\end{align*}$$  (7) (8)

$^2$The calculation of SoftMax is ignored here.


The total computational complexity of DWM is as follows:

\[ \Omega(DWM) = \Omega(SMW-MSA) + \Omega(DMSW) \\
= (1 + 4n_{win} + \frac{h_{win} + n_{win}}{h_{win}}) \frac{h_{win}}{n_{win}} C^2 + 2h_{win} \frac{C}{n_{win}} \sum_{i} w_{in_i}^2. \]  

(9)

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

### 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 [61], and object detection and instance segmentation on COCO 2017 [29]. 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, 31], we carefully avoid using any tricks that provide unfair advantage [25, 48]. Specifically, following the settings in [7, 31], DW-ViT was trained for 300 epochs with a batch size of 1024 using the AdamW optimizer [32]. 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, [47]'s augmentation and regularization strategies are used. Following the settings in [31], the repeated enhancement [21] and EMA [34] strategy are abandoned.

**Results** Tab. 2 reports the performance comparison of DW-ViT and state-of-the-art methods on ImageNet-1K.

<table>
<thead>
<tr>
<th>Method</th>
<th>#param. (M)</th>
<th>FLOPs (G)</th>
<th>Top-1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50 [19]</td>
<td>26</td>
<td>4.1</td>
<td>76.6</td>
</tr>
<tr>
<td>ResNet101 [19]</td>
<td>45</td>
<td>7.9</td>
<td>78.2</td>
</tr>
<tr>
<td>X3-2x4d [53]</td>
<td>41</td>
<td>4.0</td>
<td>77.9</td>
</tr>
<tr>
<td>X3-1024d [53]</td>
<td>44</td>
<td>8.0</td>
<td>78.7</td>
</tr>
<tr>
<td>RegNetY-4G [35]</td>
<td>21</td>
<td>4.0</td>
<td>80.0</td>
</tr>
<tr>
<td>RegNetY-8G [35]</td>
<td>39</td>
<td>8.0</td>
<td>81.7</td>
</tr>
<tr>
<td>RegNetY-16G [35]</td>
<td>84</td>
<td>16</td>
<td>82.9</td>
</tr>
<tr>
<td>DeiT-Small/16 [46]</td>
<td>22</td>
<td>4.6</td>
<td>79.9</td>
</tr>
<tr>
<td>DeiT-Small/32 [46]</td>
<td>27</td>
<td>5.6</td>
<td>81.0</td>
</tr>
<tr>
<td>DeiT-Base/16</td>
<td>22</td>
<td>5.2</td>
<td>81.5</td>
</tr>
<tr>
<td>DeiT-Base/32</td>
<td>24</td>
<td>5.2</td>
<td>81.3</td>
</tr>
<tr>
<td>CosT-Mini [54]</td>
<td>10</td>
<td>6.8</td>
<td>80.8</td>
</tr>
<tr>
<td>PVT-Small [49]</td>
<td>25</td>
<td>3.8</td>
<td>79.8</td>
</tr>
<tr>
<td>PVT-GAP [57]</td>
<td>23</td>
<td>4.6</td>
<td>81.5</td>
</tr>
<tr>
<td>CrossFormer S1 [30]</td>
<td>28</td>
<td>4.5</td>
<td>81.5</td>
</tr>
<tr>
<td>CrossFormer S1 [30]</td>
<td>28</td>
<td>4.5</td>
<td>81.3</td>
</tr>
<tr>
<td>DW-T</td>
<td>30</td>
<td>5.2</td>
<td>82.0</td>
</tr>
<tr>
<td>ViT-Base/16</td>
<td>87</td>
<td>17.6</td>
<td>77.9</td>
</tr>
<tr>
<td>DeiT-Base/16</td>
<td>87</td>
<td>17.6</td>
<td>81.8</td>
</tr>
<tr>
<td>DeiT-Base/32</td>
<td>64</td>
<td>14.1</td>
<td>82.3</td>
</tr>
<tr>
<td>DeiT-ViT/24</td>
<td>62</td>
<td>21.2</td>
<td>82.2</td>
</tr>
<tr>
<td>DeiT-ViT/48</td>
<td>66</td>
<td>14.1</td>
<td>82.8</td>
</tr>
<tr>
<td>PVT-V [48]</td>
<td>88</td>
<td>17.6</td>
<td>82.3</td>
</tr>
<tr>
<td>DeiT-Large [49]</td>
<td>81</td>
<td>9.8</td>
<td>81.7</td>
</tr>
<tr>
<td>Swin-B [31]</td>
<td>88</td>
<td>15.4</td>
<td>83.3</td>
</tr>
<tr>
<td>DW-B</td>
<td>91</td>
<td>17.0</td>
<td>83.8</td>
</tr>
</tbody>
</table>

Table 2. Performance comparison on ImageNet-1K. All models are trained and evaluated at 224 × 224 resolution. CrossFormer-S1 shows the performance in the case of single-scale embedding.

Methods of comparison include the classic and the latest ConvNet-based [19, 35, 53] and Transformer-based [4, 31, 50] 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 [46], 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 [31], 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, 28, 50] like...
4.2. Semantic Segmentation on ADE20K

ADE20K [61] 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 [52] in mm-segmentation [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 [31], the input size of the image is $512 \times 512$, AdamW [32] is used as the optimizer (the initial learning rate is $6 \times 10^{-5}$, 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 [31], 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.

### Table 3. Performance comparison on the ADE20K [61] 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 $1024 \times 1024$.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Method</th>
<th>#param. (M)</th>
<th>FLOPs (G)</th>
<th>mIoU</th>
<th>+MS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-101</td>
<td>OCRNet [58]</td>
<td>56</td>
<td>923</td>
<td>44.1</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>DLab v3+ [6]</td>
<td>63</td>
<td>1021</td>
<td>44.1</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>ACNet [14]</td>
<td>-</td>
<td>-</td>
<td>45.9</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>DNL [50]</td>
<td>69</td>
<td>1249</td>
<td>46.0</td>
<td>-</td>
</tr>
<tr>
<td>ResNet-101</td>
<td>UperNet [52]</td>
<td>86</td>
<td>1029</td>
<td>44.9</td>
<td>-</td>
</tr>
<tr>
<td>HKNet-w48 [38]</td>
<td>DLab v3+ [6]</td>
<td>71</td>
<td>664</td>
<td>45.7</td>
<td>-</td>
</tr>
<tr>
<td>ResNetSt-101 [59]</td>
<td>DLab v3+ [6]</td>
<td>66</td>
<td>1051</td>
<td>46.9</td>
<td>-</td>
</tr>
<tr>
<td>ResNetSt-200 [59]</td>
<td>DLab v3+ [6]</td>
<td>88</td>
<td>1381</td>
<td>48.4</td>
<td>-</td>
</tr>
<tr>
<td>PV-TS [49]</td>
<td>S-FPN [26]</td>
<td>28</td>
<td>-</td>
<td>39.8</td>
<td>-</td>
</tr>
<tr>
<td>PV-T-M</td>
<td>S-FPN</td>
<td>48</td>
<td>219</td>
<td>41.6</td>
<td>-</td>
</tr>
<tr>
<td>PV-T-L</td>
<td>S-FPN</td>
<td>65</td>
<td>283</td>
<td>42.1</td>
<td>-</td>
</tr>
<tr>
<td>CAT-S [28]</td>
<td>S-FPN</td>
<td>41</td>
<td>214</td>
<td>42.8</td>
<td>-</td>
</tr>
<tr>
<td>CAT-B</td>
<td>S-FPN</td>
<td>55</td>
<td>276</td>
<td>44.9</td>
<td>-</td>
</tr>
<tr>
<td>Swin-T [31]</td>
<td>UperNet [52]</td>
<td>60</td>
<td>945</td>
<td>44.5</td>
<td>45.8</td>
</tr>
<tr>
<td>Swin-B [31]</td>
<td>UperNet [52]</td>
<td>121</td>
<td>1188</td>
<td>48.1</td>
<td>49.7</td>
</tr>
<tr>
<td>DW-T</td>
<td>UperNet [52]</td>
<td>61</td>
<td>953</td>
<td>45.7</td>
<td>46.9</td>
</tr>
<tr>
<td>DW-B</td>
<td>UperNet [52]</td>
<td>125</td>
<td>1200</td>
<td><strong>48.7</strong></td>
<td><strong>50.3</strong></td>
</tr>
</tbody>
</table>

4.3. Object Detection on COCO

Further, we benchmark DW-ViT on object detection and instance segmentation with COCO 2017 [29]. 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 [18] 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, 39], the AdamW [32] 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 [46], DW-T is improved by 3.5 points. Compared with Swin [31], 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 FLOPs 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 [31]. Single-scale windows are taken from [7, 11, 14, 17, 21, 23], and multi-scale windows are set to [7, 14, 21]$^3$. Their performance on ImageNet-1K [10] are shown in Tab. 5.

In Tab. 5, DMSW shows three states (‘✓’, ‘−’, ‘✓’). MSW-MSA + ‘1’ refers to removing the dynamic weight generation and directly assigning the same weight ($\frac{1}{2}$) 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 than that of Swin-T with win $= 21$. This may be due to the

---

$^3$We adopted the original settings in Swin [31] 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.
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 [31], 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. However, an ideal strategy is to allocate the entire computational budget to the most potential windows at each layer of the network.

Acknowledgment

This work was partially supported in part by National Key R&D Program of China under Grant No.2020AAA0109700, NSFC under Grant (No.61972315 and No.61976233), Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Guangdong Outstanding Youth Fund (Grant No. 2021B1515020061), Australian Research Council (ARC) Discovery Early Career Researcher Award (DE190100626, Shaanxi Province International Science and Technology Cooperation Program Project-Key Projects No.2022KWZ-14, Ministry of Science and Technology Foundation Project 2020AAA0106900 and Key Realm R&D Program of Guangzhou 202007030007 and Open Fund from Alibaba.

5. Conclusion

The size of the window has an important impact on the performance of the model. There is currently very little systematic study of window size in the window-based ViT works. In this paper, we challenged this problem for the first time. 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

Table 4. Performance comparison of object detection and instance segmentation on the COCO2017 val dataset. Two object detection frameworks are used: Mask R-CNN [18] 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.

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

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.

Acknowledgment

This work was partially supported in part by National Key R&D Program of China under Grant No.2020AAA0109700, NSFC under Grant (No.61972315 and No.61976233), Guangdong Province Basic and Applied Basic Research (Regional Joint Fund-Key) Grant No.2019B1515120039, Guangdong Outstanding Youth Fund (Grant No. 2021B1515020061), Australian Research Council (ARC) Discovery Early Career Researcher Award (DE190100626, Shaanxi Province International Science and Technology Cooperation Program Project-Key Projects No.2022KWZ-14, Ministry of Science and Technology Foundation Project 2020AAA0106900 and Key Realm R&D Program of Guangzhou 202007030007 and Open Fund from Alibaba.

11994
References


[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, 2, 5, 6, 7, 8


