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, AAII, 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

1. Introduction

In computer vision (CV) tasks, the visual transformer represented by Vision Transformer (ViT) [12] has shown great potential. These methods have achieved impres-
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.

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., \( 36 \times 36 \)). 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, \( e.g. M = \text{win} = 3 \)). For simplicity, patch and position embeddings are omitted here.

![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.](image1)

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 monotinous. 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 encoded. The calculation amount of this standard self-attention. In this way, multi-scale feature information are then fed to the transformer encoder after being 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 introduced pyramid pooling into the 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 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 \(H/4 \times W/4 \times C\).

These generated patch tokens are then used as the input of the DWM transformer layers, and the number (i.e., \(H/4 \times W/4\)) 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 \(H/8 \times W/8\), \(H/16 \times W/16\) and \(H/32 \times W/32\), 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 \(n_{\text{win}}\) scale windows. Here we take \(h = 6\) and \(n_{\text{win}} = 3\) as an example. The multi-head \(h\) of MSA is evenly divided into \(n_{\text{win}}\) groups, which perform multi-head self-attention at different scales window to capture multi-scale window information. A group of windows here can be self-attention at different scales window to capture multi-scale window information, while the latter window module (DMSW). The former is responsible for the attention module (MSW-MSA) and a dynamic multi-scale information. Specifically, the details of these two parts are as follows:

**Fuse:** It mainly consists of a pooling layer \(F_{\text{gp}}\) and two pairs of fully connected layers \(F_{\text{fc}}\) and activation layers \(F_a\). The calculation process is as follows:

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

where the \(i\)-th branch \(y_i\) is divided into \([H/n_{\text{win}}] \times [W/n_{\text{win}}]\) windows in the spatial dimension. Each window is expanded into a token sequence of length \(w_{\text{win}} \times w_{\text{win}}\) and used as the input of the \(i\)-th branch \(\text{W-MSA}_{\text{win}}\), of MSW-MSA. The structure of W-MSA is illustrated in Fig. 3. The output of \(\text{W-MSA}_{\text{win}}\) is reconstructed as \(H \times W\) in the spatial dimension, and the final output dimension is \(H \times W \times C/n_{\text{win}}\). 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. \(y_{\text{MSW-MSA}} = \text{Concat}(\{\text{W-MSA}_{\text{win}}(\cdot), i = 1, \ldots, n_{\text{win}}\})\) 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:** The features of all branches are first concatenated, then passed through a pooling layer \(F_{\text{gp}}\) and two fully connected layers \(F_{\text{fc}}\) and activation layers \(F_a\). The calculation process is as follows:

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

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 \(C/n_{\text{win}}\).

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

\[
\begin{align*}
y_{\text{Select}} &= F_{\text{fc}}(F_{\text{gp}}(\sum_{i}^{n_{\text{win}}}{\alpha_i \times \text{W-MSA}_{\text{win}}(\hat{y}_i)})), \\
\alpha_i &= \frac{e^{F_{\alpha_i}(y_{\text{Fuse}})}}{\sum_i^{n_{\text{win}}} e^{F_{\alpha_i}(y_{\text{Fuse}})}}, i = 1, 2, \ldots, n_{\text{win}},
\end{align*}
\]
where $\alpha_i \in \mathbb{R}^{1 \times 1 \times \frac{C}{n_{\text{win}}}}$. The DMSW module output is as follows:

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

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 nonlinearity. DSW achieves cross-window information exchange by moving the feature $\left\lfloor \frac{\text{win}}{2} \right\rfloor$ patches to the upper left in the spatial dimension. When the feature is reconstructed, it moves $\left\lfloor \frac{\text{win}}{2} \right\rfloor$ 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:

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

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 $B = \{ B_i \in \mathbb{R}^{M^2 \times M^2}, i = 1, 2, \ldots, n_{\text{win}} \}$ 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:

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

where $Q_i, K_i, V_i \in \mathbb{R}^{M^2 \times d}$ are query, key, and value matrices, while $M^2_i$ 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 $B = \{ B_i, i = 1, \ldots, n_{\text{win}} \}$. 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 $\hat{B}_i$.

### 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: $\Omega(\text{SMW-MSA})$ and $\Omega(\text{DMSW})$. For an image with $h \times w$ patches, their computational complexity is as follows:

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

$^2$The calculation of SoftMax is ignored here.
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 \( W_{in} \) 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.

The total computational complexity of DWM is as follows:

\[
\Omega(\text{DWM}) = \Omega(\text{SMW-MSA}) + \Omega(\text{DMSW})
\]

\[
= (1 + 4n_{\text{win}} + \frac{h \cdot w + n_{\text{win}}}{w \cdot n_{\text{win}}}) \cdot \frac{n_{\text{win}}}{2} \cdot \frac{C^2}{n_{\text{win}}} + 2h \cdot w \cdot \frac{C}{n_{\text{win}}} \sum_{i} w_{i}n_{i}^2.
\]

Since both \( win_i \) and \( n_{\text{win}} \) 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. Methods of comparison include the classic and the latest ConvNet [19, 53] and Transformer-based [4, 31, 50] models. All models are trained and evaluated at 224 \( \times \) 224 resolution. CrossFormer-S1 shows the performance in the case of single-scale embedding.
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 × 1024.

| Backbone      | Method             | #Param. (M) | FLOPs (G) | mIoU  | +MS |
|---------------|--------------------|-------------|-----------|-------|-----|
| ResNet-101 [19] | DANet [31]        | 69          | 1119      | 45.3  | -   |
| ResNet-101    | OCRNet [56]        | 56          | 923       | 44.1  | -   |
| ResNet-101    | DLab.v3+ [5]       | 63          | 1021      | 44.1  | -   |
| ResNet-101    | ACNet [14]         | -           | -         | 45.9  | -   |
| ResNet-101    | DNL [50]           | 69          | 1249      | 46.0  | -   |
| ResNet-101    | UperNet [52]       | 86          | 1029      | 44.9  | -   |
| HKNet-w48 [58]| DLab.v3+ [5]       | 71          | 664       | 45.7  | -   |
| ResNet-101    | UperNet [52]       | 86          | 1029      | 44.9  | -   |
| ResNet-101    | UperNet [52]       | 86          | 1029      | 44.9  | -   |
| PVT-S [49]    | S-FPN              | 28          | -         | 39.8  | -   |
| PVT-M         | S-FPN              | 48          | 219       | 41.6  | -   |
| PVT-L         | S-FPN              | 65          | 283       | 42.1  | -   |
| CAT-S [28]    | S-FPN              | 41          | 214       | 42.8  | -   |
| CAT-B         | S-FPN              | 55          | 276       | 44.9  | -   |
| Swin-T [31]   | UperNet [52]       | 60          | 945       | 44.5  | 45.8|
| Swin-B [31]   | UperNet [52]       | 121         | 1188      | 48.1  | 49.7|
| DW-T          | UperNet [52]       | 61          | 953       | 45.7  | 46.9|
| DW-B          | UperNet [52]       | 125         | 1200      | 48.7  | 50.3|

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 × schedule (it has 36 epochs, and the learning rate decays by 10 × between epochs 27 and 33). It is implemented based on MM Detection [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 (‘1’, ‘·’, ‘✓’). MSW-MSA + ‘1’ refers to removing the dynamic weight generation and directly assigning the same weight (1) 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

3We 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.
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.

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 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 (DECREA) under 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.
[28] 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

[29] 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

[30] 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

[31] 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

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

[33] Hyunseob 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

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

[35] 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

[36] Colin Raffel, Noam M. Shazeer, Adam Roberts, Ilija Radosavovic, Raj Prateek Kosaraju, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Christian 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

[37] 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

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

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

[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] 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

[42] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In ICML, 2021. 6

[43] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. ArXiv, abs/2103.17239, 2021. 6

[44] 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, 4, 6, 7, 8

[45] 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

[46] Yu-Huan Wu, Yun Liu, Xin Zhan, and Ming-Ming Cheng. P2l: Pyramid pooling transformer for scene understanding. arXiv preprint arXiv:2106.12011, 2021. 1, 3

[47] 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

[48] 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

[49] Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. Coscale conv-attentional image transformers. arXiv preprint arXiv:2104.06599, 2021. 6

[50] 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

[56] 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

[57] 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, 3, 6

[58] 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

[59] 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

[60] Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. Multi-scale vision long-former: A new vision transformer for high-resolution image encoding. *ArXiv*, abs/2103.15358, 2021. 4, 8

[61] 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

[62] 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