Dynamic Group Transformer: A General Vision Transformer Backbone with Dynamic Group Attention

Kai Liu[1*†], **Tianyi Wu**[2,3 *], **Cong Liu**[1 ‡], **Guodong Guo**[2,3 ‡]
[1]Sun Yat-sen University, Guangzhou, China
[2]Institute of Deep Learning, Baidu Research, Beijing, China
[3]National Engineering Laboratory for Deep Learning Technology and Application, Beijing, China
liuk95@mail2.sysu.edu.cn, liucong3@mail.sysu.edu.cn, {wutianyi01, guoguodong01}@baidu.com

Abstract

Recently, Transformers have shown promising performance in various vision tasks. To reduce the quadratic computation complexity caused by each query attending to all keys/values, various methods have constrained the range of attention within local regions, where each query only attends to keys/values within a hand-crafted window. However, these hand-crafted window partition mechanisms are data-agnostic and ignore their input content, so it is likely that one query maybe attends to irrelevant keys/values. To address this issue, we propose a Dynamic Group Attention (DG-Attention), which dynamically divides all queries into multiple groups and selects the most relevant keys/values for each group. Our DG-Attention can flexibly model more relevant dependencies without any spatial constraint that is used in hand-crafted window based attention. Built on the DG-Attention, we develop a general vision transformer backbone named Dynamic Group Transformer (DGT). Extensive experiments show that our models can outperform the state-of-the-art methods on multiple common vision tasks, including image classification, semantic segmentation, object detection, and instance segmentation.

1 Introduction

Recently, Transformer has shown a great potential for various vision tasks [Dosovitskiy *et al.*, 2020; Touvron *et al.*, 2020; Liu *et al.*, 2021b; Dong *et al.*, 2021]. The pioneer Vision Transformer [Dosovitskiy *et al.*, 2020] (ViT) stacked multiple Transformer blocks to process non-overlapping image patch (i.e., visual token) sequences for image classification. However, the global self-attention in Transformer makes each query attend to all keys, which has the quadratic complexity to sequence length, and results in expensive computation costs and memory usage, especially for high-resolution images.

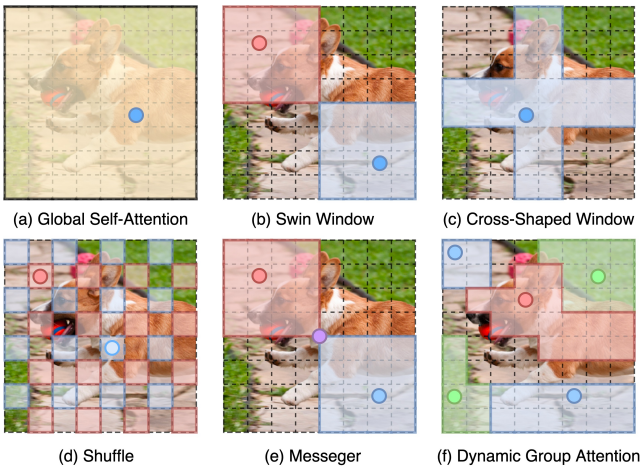


Figure 1: Illustrate our Dynamic Group Attention in comparison with other attention mechanisms in Transformer backbones. (a) Global self-attention, each query attends to all keys/values. (b) $\sim$ (e) Window-based attentions, each query attends to keys/values within a fixed window. (d) Dynamic group attention, all queries are dynamically divided into several groups, and each query attends to relevant keys/values only.

To improve the efficiency of the global self-attention, the state-of-the-art methods [Liu *et al.*, 2021b; Huang *et al.*, 2021; Dong *et al.*, 2021; Fang *et al.*, 2021] focused on how to divide the global image into multiple local regions (or windows). Each query only attends to a few keys within the manually designed local regions (windows). For example, Swin Transformer [Liu *et al.*, 2021b] computed the self-attention within each local window and employed a shifted mechanism to make cross-window connections (Figure 1 (b)). Different from Swin Transformer, CSwin [Dong *et al.*, 2021] proposed cross-shaped window self-attention for computing attention in the horizontal and vertical stripes in parallel that form a cross-shaped window (Figure 1 (c)). Shuffle Transformer [Huang *et al.*, 2021] presented a spatial shuffle operation to make information flow across windows (Figure 1 (d)). MSG-Transformer [Fang *et al.*, 2021] proposed to compute attention in local regular windows, and used additional MSG tokens to get connections between them (Figure 1 (e)). These window-based methods achieved excellent performances and

*Equal contribution
†Interns at the Institute of Deep Learning, Baidu Research
‡Corresponding author

were superior to the CNN counterparts, however, they rely on hand-crafted window partition mechanisms. Furthermore, these partition methods are data-agnostic and ignore the input content, as a result, it is likely that one query maybe attends to irrelevant keys/values.

To address the issues mentioned above, a good idea is to dynamically select relevant keys/values for each query. However, it leads to unreasonably high memory usage and computation complexity. We propose dynamic group attention (DG-Attention), which dynamically divides all queries into multiple groups and selects the most relevant keys/values for each group. Specifically, the input visual tokens (or feature vectors) are divided adaptively into multiple groups according to their similarity to all cluster centroids. Therefore, such a partition mechanism is adaptive to input images. Then, we use the cluster centroid of each group to select the most relevant keys/values subset from the whole keys/values set, and the self-attention is conducted within each group. It enables our model to focus on relevant keys/values without any spatial constraint. And also, through the dynamic grouping, our DG-Attention does not cause a high memory usage or large computation cost. For example, as shown in Figure 1 (f), the red point (query) can attend to its relevant region denoted with red solid line boundaries, and the blue point can attend to the regions with blue solid line boundaries. Benefiting from the data-dependent and flexible group mechanism, our DG-Attention shows superiority to the other window-based self-attention illustrated in Figure 1.

Based on the proposed DG-Attention, we design a general vision transformer backbone for image classification, named Dynamic Group Transformer (DGT). We scale our approach up to get a family of models, including DGT-T (24M), DGT-S (52M), and DGT-B (90M). They achieve significantly a better performance than previous methods. Our DGT-T can achieve Top-1 classification accuracy of 83.8% on ImageNet-1k, 50.2% mIoU on ADE20K for semantic segmentation, 47.7% box mAP for object detection, and 43.0% mask mAP on COCO for instance segmentation, outperforming the state-of-the-art methods. Furthermore, our largest variant DGT-B is also superior to the previous methods, achieving 85.0% Top-1 accuracy on ImageNet-1K, 51.2% mIoU on ADE20K, 49.1% box mAP, and 44.1% mask mAP on COCO dataset.

2 Related Work

This section briefly reviews related works, including improving efficiency and enhancing inductive bias for Vision Transformer.

Improving Efficiency for Vision Transformer. There are two main categories of methods to reduce the computation demand for Vision Transformer. (1) Pruning Token. It aims to remove redundant tokens and reduce the number of tokens entering into attention modules to save the computation demand. For example, DynamicViT [Rao *et al.*, 2021] pruned tokens in each layer with Gumbel softmax. IA-Red [Pan *et al.*, 2021] used reinforcement Learning to achieve a similar effect. Such methods achieved good performances on image classification, but they are not friendly enough for downstream dense prediction tasks. (2) Designing efficient attention mechanisms. Such methods mainly explored how to make each query attend to partial keys/values for reducing the computational cost. PVT [Wang *et al.*, 2021a] directly downsampled the keys and values in each block. Swin transformer [Liu *et al.*, 2021b] divided all queries/keys/values into multiple windows and computed the self-attention within each local window. Similarly, CSwin transformer [Dong *et al.*, 2021] expanded the window into a cross-shaped window. MSG-Transformer [Fang *et al.*, 2021] used additional MSG tokens to make connections between windows. Different from these methods that employed pre-designed, hand-crafted window partition mechanisms, our method dynamically divides all queries into multiple groups and selects the most relevant keys/values for each group.

Enhancing Inductive Bias for Vision Transformer. Vision transformers have shown successes in various computer vision tasks, due to their ability to model long-range dependencies within an image. However, recent works also showed that inductive bias could be incorporated for vision transformers. CPE [Chu *et al.*, 2021b] used convolution layers to generate the conditional position encoding. CVT [Wu *et al.*, 2021] employed convolution layers to generate the queries, keys and values. CMT [Guo *et al.*, 2021] also incorporated the convolution layers into the FFN.

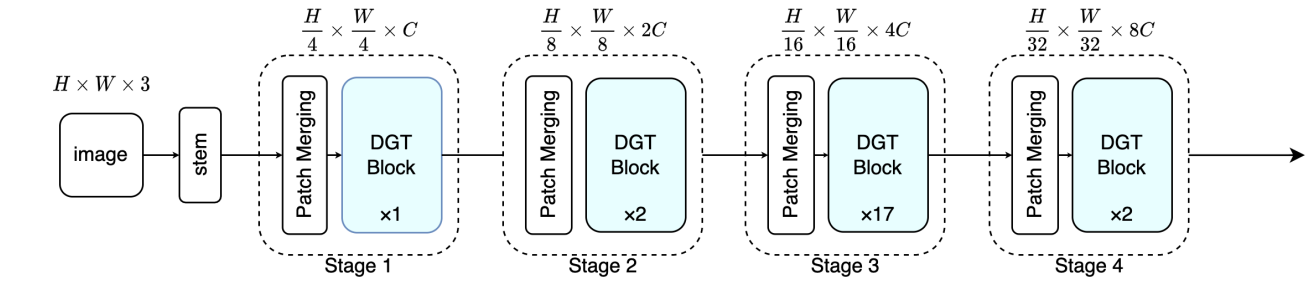
3 Method

In this section, we first introduce our Dynamic Group attention (DG-Attention). Then, we present the composition of the Dynamic Group Transformer Block. Finally, we describe the overall architecture and variant configurations of our Dynamic Group Transformer (DGT) backbone.

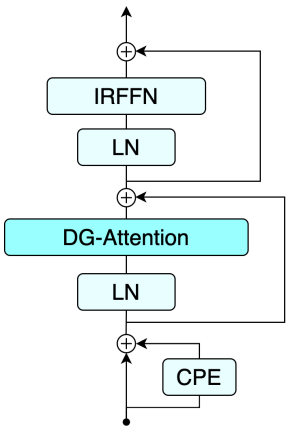
3.1 Dynamic Group Attention

To make each query attend to relevant keys/values, we propose a Dynamic Group Attention (DG-Attention). It dynamically divides all queries into multiple groups and selects the most relevant keys/values for each group to compute the self-attention. As shown in Figure 2(c), given an input feature map $X \in \mathcal{R}^{H\times W\times C}$ (C is the channel number, H and W denotes the height and width, respectively), we first get query embeddings $\{X_Q^i\}_{i=1}^L$, key embeddings $\{X_K^i\}_{i=1}^L$, and value embeddings $\{X_V^i\}_{i=1}^L$, where $L = H \times W$. For simplicity, we assume there is only one head in the DG-Attention. It's easy to expand to the multi-head situation where each head has its queries, keys, values, and cluster centroids. Then, we use k-means clustering algorithm to dynamically divide all queries into G different query groups (clusters) $X_Q = \{X_{Q_j}|X_{Q_j} \in \mathcal{R}^{N_j\times C}\}_{j=1}^G$, where j is the group index and N_j is the number of queries in the j^{th} group. Meanwhile, we use top-k operations to find the k most relevant keys and values for each query group, which are denoted as $X_K = \{X_{K_j}|X_{K_j} \in \mathcal{R}^{k\times C}\}_{j=1}^G$ and $X_V = \{X_{V_j}|X_{V_j} \in \mathcal{R}^{k\times C}\}_{j=1}^G$, respectively.

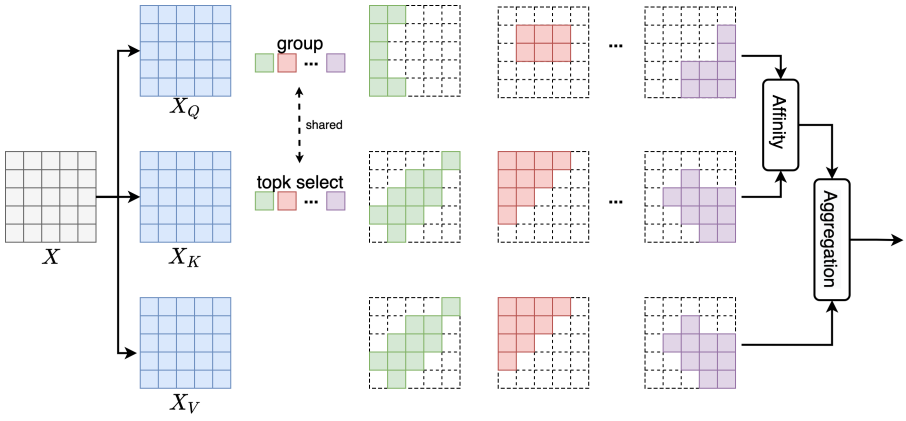
Specifically, for the j^{th} query group, we compute the dot product between its cluster centroid e_j and all keys $\{X_K^i\}_{i=1}^L$, and then select the k most relevant elements according to the dot product sorting, which can be formulated as follow:



(a) Overall Architecture of Dynamic Group Transformer (DGT)



(b) DGT Block



(c) Dynamic Group Attention (DG-Attention)

Figure 2: (a) The overall architecture of our Dynamic Group Transformer. (b) The composition of each block. (c) Illustration of our DG-Attention. It can dynamically divides all queries into multiple groups and selects the most relevant keys/values for each group.

$$\begin{aligned} id_j &= Top\text{-}k(e_j, \{X_K^i\}_{i=1}^L) \in \{1, .., L\}^k, \\ X_{K_j} &= \{X_K^i | i \in id_j\} \in \mathcal{R}^{k \times C}, \\ X_{V_j} &= \{X_V^i | i \in id_j\} \in \mathcal{R}^{k \times C}, \end{aligned} \tag{1}$$

where $Top\text{-}k$ is the function that returns the indices of top k values, and id_j is an index vector. Then, the self-attention is conducted within each group:

$$Y_j = SA(X_{Q_j}, X_{K_j}, X_{V_j}) \in \mathcal{R}^{N_j \times C}, \tag{2}$$

where SA denotes the Self-Attention, Y_j is the updated output of the j^{th} query group. Finally, $\{Y_j\}_{j=1}^G$ are scattered into the output $Y \in \mathcal{R}^{L \times C}$ according to their original spatial position indexes.

As each group has a different number of queries, this algorithm cannot be implemented using the general matrix multiplication. We implement this algorithm using CUDA, and the detail can be found in the supplementary material.

To make the training stable, we update the cluster centroids with exponential moving average after each iteration. Specifically, for the j^{th} cluster centroid, we compute the current cluster centroid as follow:

$$e'_j = \frac{1}{N_j} \sum_i Norm(X_{Q_j}^i). \tag{3}$$

Then, we update the cluster centroid as below:

$$e_j = Norm(\tau \times e_j + (1 - \tau) \times {e'}_j), \tag{4}$$

where τ is a hyper-parameter to control the update speed. We empirically set τ to $0.1 \times lr$, where lr is the learning rate.

Computation Complexity Analysis

We analyze the computation complexity of our DG-Attention and the global self-attention to further reveal the efficiency of our method. Here, we only consider the process of computing the attention maps and weighted sums of values for clarity. Given input features of size $L \times C$, the global self-attention has a computational complexity of

$$\Omega_{Global} = 2L^2C. \tag{5}$$

In DG-Attention, each query only attends to k keys, so the basic computation complexity of DG-Attention is $2kLC$. Besides, grouping queries and selecting the most significant k keys require an additional computation cost of $2LGC + kGlogL$. Therefore, the total computation complexity of our DG-Attention is

$$\Omega_{DG\text{-}Attention} = 2kLC + 2LGC + kGlogL \tag{6}$$

The ratio of the computation complexity of our DG-Attention and Global self-attention is:

$$\frac{\Omega_{DG\text{-}Attention}}{\Omega_{Global}} = \frac{2kLC + 2LGC + kGlogL}{2L^2C} = \frac{k}{L} + \frac{G}{L} + \frac{kGlogL}{2L^2C} < 1 \quad (7)$$

where L is larger than G and k. For high-resolution inputs, the ratio $\frac{\Omega_{DG\text{-}Attention}}{\Omega_{Global}} << 1$. Typically, for the ImageNet classification task, k is set to 98 for the first three stages, while the corresponding L is 3136, 784, and 196. Thus, the ratio is 0.05, 0.19, and 0.75 for DGT-T. Besides, k is independent of the shapes of the parameters in our models, so we can adjust k to balance the performance and computation efficiency.

3.2 Dynamic Group Transformer Block

The Dynamic Group Transformer block is shown in Figure 2(b). It first employs the widely-used conditional position embeddings (CPE) [Chu *et al.*, 2021b] to generate the positional information. Then, DG-Attention is applied to model spatial relevant dependencies flexibly and dynamically. Last, IRFFN (Inverted Residual Feed-forward Network) [Guo *et al.*, 2021] further is employed to capture local dependencies. The forward process of the l^{th} block can be formulated as follows:

$$\tilde{X}^l = X^{l-1} + CPE(X^{l-1}), \quad (8)$$

$$\hat{X}^l = \tilde{X}^l + DG\text{-}Attention(LN(\tilde{X}^l)), \quad (9)$$

$$X^l = \hat{X}^l + IRFFN(LN(\hat{X}^l)), \quad (10)$$

where $LN(\cdot)$ denotes Layer Normalization, X^l and X^{l-1} are the output of the l^{th} and l-1^{th} block, respectively.

3.3 Overall Architecture and Variants

Our Dynamic Group Transformer (DGT) consists of a convolutional stem, four hierarchical stages, and a classifier head, as shown in Figure 2 (a). The stem is designed to extract local dependency, similar to [Guo *et al.*, 2021], which consists of one 3×3 convolution layer with stride = 2 and two 3×3 convolution layers with stride = 1. After the stem, each stage contains a patch merging layer and multiple transformer blocks. The first three stages use the DGT block, and the last stage applies global self-attention (GSA) block, which is achieved by replacing the DG-Attention with global self-attention in DGT block. We decrease the number of tokens and double the channel dimension by using a 3×3 convolutional layer with stride = 2 before each stage to produce a hierarchical representation. The final classifier head consists of two linear layers.

Finally, we design three different variants, including DGT-Tiny (DGT-T), DGT-Small (DGT-S), and DGT-Base (DGT-B), whose detailed configurations are shown in Table 1. For all variants, the number of blocks in each stage is fixed with [1,2,17,2]. In each DGT block, the expand ratios of IRFFN are set to 4, the number of groups G is 48. The number of selected keys/values k is 98 for image classification on ImageNet [Deng *et al.*, 2009]. The main differences among all variants are the channel dimension and the number of heads in DGT blocks. Besides, to train the model stably, we apply post LayerNorm and cosine attention [Liu *et al.*, 2021a] in DGT-S and DGT-B.

Stage/Stride	Layer	DGT-T	DGT-S	DGT-B
Stride=2	Stem	$3\times3, 32, s=2$ $[3\times3, 32]\times2$	$3\times3, 48, s=2$ $[3\times3, 48]\times2$	$3\times3, 64, s=2$ $[3\times3, 64]\times2$
Stage 1 Stride=4	Patch Merge	3×3, 64, s=2	3×3, 96, s=2	3×3, 128, s=2
	DGT Block	$\begin{bmatrix} H_1{=}2 \\ G_1{=}48 \\ k_1{=}98 \\ R_1{=}4 \end{bmatrix}\times1$	$\begin{bmatrix} H_1{=}3 \\ G_1{=}48 \\ k_1{=}98 \\ R_1{=}4 \end{bmatrix}\times1$	$\begin{bmatrix} H_1{=}4 \\ G_1{=}48 \\ k_1{=}98 \\ R_1{=}4 \end{bmatrix}\times1$
Stage 2 Stride=8	Patch Merge	3×3, 128, s=2	3×3, 192, s=2	3×3, 256, s=2
	DGT Block	$\begin{bmatrix} H_2{=}4 \\ G_2{=}48 \\ k_2{=}98 \\ R_2{=}4 \end{bmatrix}\times2$	$\begin{bmatrix} H_2{=}6 \\ G_2{=}48 \\ k_2{=}98 \\ R_2{=}4 \end{bmatrix}\times2$	$\begin{bmatrix} H_2{=}8 \\ G_2{=}48 \\ k_2{=}98 \\ R_2{=}4 \end{bmatrix}\times2$
Stage 3 Stride=16	Patch Merge	3×3, 256, s=2	3×3, 384, s=2	3×3, 512, s=2
	DGT Block	$\begin{bmatrix} H_3{=}8 \\ G_3{=}48 \\ k_3{=}98 \\ R_3{=}4 \end{bmatrix}\times17$	$\begin{bmatrix} H_3{=}12 \\ G_3{=}48 \\ k_3{=}98 \\ R_3{=}4 \end{bmatrix}\times17$	$\begin{bmatrix} H_3{=}16 \\ G_3{=}48 \\ k_3{=}98 \\ R_3{=}4 \end{bmatrix}\times17$
Stage 4 Stride=32	Patch Merge	3×3, 512, s=2	3×3, 768, s=2	3×3, 1024, s=2
	GSA Block	$\begin{bmatrix} H_4{=}16 \\ R_4{=}4 \end{bmatrix}\times2$	$\begin{bmatrix} H_4{=}24 \\ R_4{=}4 \end{bmatrix}\times2$	$\begin{bmatrix} H_4{=}32 \\ R_4{=}4 \end{bmatrix}\times2$
	FC	1×1, 1280		
	Classifier	1×1, 1000		
Params		24.09 M	51.76 M	90.28 M
Flops		4.35 G	9.41 G	16.4 G

Table 1: Detailed configurations of different variants of our DGT. H_i, G_i and k_i represent the number of heads, group, and the selected key/value in DGT block, respectively. R_i is the expand ratio in IRFFN.

4 Experiments

We first compare our Dynamic Group Transformer (DGT) with the state-of-the-art backbones on ImageNet-1K [Deng *et al.*, 2009] for image classification. To further verify the effectiveness and generalization of our backbone, we perform experiments on ADE20K [Zhou *et al.*, 2017] for semantic segmentation, COCO [Lin *et al.*, 2014] for object detection and instance segmentation. Finally, we analyze the key design of our Dynamic Group Transformer.

4.1 Image Classification on ImageNet-1K

We conduct experiments on ImageNet-1K [Deng *et al.*, 2009] dataset, which has 1.28M images in the training set and 50K images in the validation set. Detailed settings are described in the supplementary material.

Results

Table 2 compares the performance of our DGT with the state-of-the-art CNN models and vision transformer backbones on ImageNet-1K validation set. We can see that our DGT variants outperform the state-of-the-art CNN models and vision transformer models when using similar FLOPs. DGT-T achieves 83.8% top-1 accuracy with 4.3G FLOPs and outperforms the CNN models Reg-4G and Efficient B4 by 3.8% and 0.9%, respectively. Meanwhile, our DGT outperforms the advanced Transformer-based backbones, and is $+1.6\%$ and $+0.8\%$ higher than Swin and CSwin Transformer, respectively, for all variants under the similar model size and FLOPs. For example, our DGT-T can surpass PVT-S, Swin-T, and CSwin-S by 4.0%, 2.5%, and 1.1%, respectively. Our DGT-B can outperform Swin-B and CSwin-B by 1.7% and 0.8%, respectively. These results demonstrate the effectiveness and efficiency of our approach.

Method	Param.	FLOPs	Top-1
DeiT-S [Touvron *et al.*, 2020]	22M	4.6G	79.8
PVT-S [Wang *et al.*, 2021a]	25M	3.8G	79.8
Reg-4G [Radosavovic *et al.*, 2020]	21M	4.0G	80.0
Swin-T [Liu *et al.*, 2021b]	29M	4.5G	81.3
CPVT-S [Chu *et al.*, 2021b]	23M	4.6G	81.5
CvT-13 [Wu *et al.*, 2021]	20M	4.5G	81.6
ViL-S [Zhang *et al.*, 2021]	25M	4.9G	82.0
CSWin-T [Dong *et al.*, 2021]	23M	4.3G	82.7
Eff-B4* [Tan and Le, 2019]	19M	4.2G	82.9
CMT-S [Guo *et al.*, 2021]	25M	4.0G	83.5
DGT-T (ours)	24M	4.3G	**83.8**
PVT-M [Wang *et al.*, 2021a]	44M	6.7G	81.2
PVT-L [Wang *et al.*, 2021a]	61M	9.8G	81.7
Reg-8G [Radosavovic *et al.*, 2020]	39M	8.0G	81.7
CvT-21 [Wu *et al.*, 2021]	32M	7.1G	82.5
Swin-S [Liu *et al.*, 2021b]	50M	8.7G	83.0
Twins-B [Chu *et al.*, 2021a]	56M	8.3G	83.2
ViL-M [Zhang *et al.*, 2021]	40M	8.7G	83.3
CSWin-S [Dong *et al.*, 2021]	35M	6.9G	83.6
Eff-B5* [Tan and Le, 2019]	30M	9.9G	83.6
DGT-S (ours)	52M	9.4G	**84.6**
DeiT-B [Touvron *et al.*, 2020]	87M	17.5G	81.8
CPVT-B [Chu *et al.*, 2021b]	88M	17.6G	82.3
Reg-16G [Radosavovic *et al.*, 2020]	84M	16.0G	82.9
ViL-B [Zhang *et al.*, 2021]	56M	13.4G	83.2
Swin-B [Liu *et al.*, 2021b]	88M	15.4G	83.3
Twins-L [Chu *et al.*, 2021a]	99M	14.8G	83.7
Eff-B6* [Tan and Le, 2019]	43M	19.0G	84.0
CSWin-B [Dong *et al.*, 2021]	78M	15.0G	84.2
DGT-B (ours)	90M	16.4G	**85.0**

Table 2: Comparison with the state-of-the-art models, trained with 224× 224 on ImageNet-1K Classification.

4.2 Semantic Segmentation on ADE20K

To demonstrate the superiority of our Dynamic Group Transformer for semantic segmentation. We conduct experiments on ADE20K with the widely-used UperNet [Xiao *et al.*, 2018] framework for fair comparisons to other backbones. Detailed implementation can be found in the supplementary material.

Results

Table 3 shows the comparisons of UperNet [Xiao *et al.*, 2018] with various advanced Transformer backbones on ADE20K validation set. We report both single-scale (SS) mIoU and multi-scale (MS) mIoU for a cleaner comparison. Our DGT variants outperforms the state-of-the-art methods consistently. Specifically, our DGT-T achieves 50.2% mIoU with single scale testing, outperforming the Swin-T and CrossFormer-S by 5.7% and 2.6%. Our DGT-S outperforms Swin-S and CrossFormer-B by 3.2% and 0.9% SS mIoU. Besides, our DGT-B achieves 51.2%/51.8% SS/MS mIoU, outperforming the Swin-B and CrossFormer-L by 3.1%/2.1% and 0.7%/0.4%. These results demonstrate the advantages of our Dynamic Group Transformer for semantic segmentation.

Backbone	Prams (M)	FLOPs (G)	mIoU SS/MS
TwinsP-S [Chu *et al.*, 2021a]	54.6	919	46.2/47.5
Twins-S [Chu *et al.*, 2021a]	54.4	901	46.2/47.1
Swin-T [Liu *et al.*, 2021b]	59.9	945	44.5/45.8
CrossFormer-S [Wang *et al.*, 2021b]	62.3	980	47.6/48.4
DGT-T (ours)	52.5	954	**50.2/50.8**
Res101 [He *et al.*, 2016]	86.0	1029	–/44.9
TwinsP-B [Chu *et al.*, 2021a]	74.3	977	47.1/48.4
Twins-B [Chu *et al.*, 2021a]	88.5	1020	47.7/48.9
Swin-S [Liu *et al.*, 2021b]	81.3	1038	47.6/49.5
CrossFormer-B [Wang *et al.*, 2021b]	83.6	1090	49.7/50.6
DGT-S (ours)	81.9	1074	**50.8/51.6**
TwinsP-L [Chu *et al.*, 2021a]	91.5	1041	48.6/49.8
Twins-L [Chu *et al.*, 2021a]	133.0	1164	48.8/50.2
Swin-B [Liu *et al.*, 2021b]	121.0	1188	48.1/49.7
CrossFormer-L [Wang *et al.*, 2021b]	125.5	1244	50.5/51.4
DGT-B (ours)	122.2	1234	**51.2/51.8**

Table 3: Comparison with different backbones on ADE20K. FLOPs are calculated with the resolution of 512×2048.

4.3 Object Detection and Instance Segmentation on COCO

We further evaluate our DGT backbone on COCO [Lin *et al.*, 2014] dataset for object detection and instance segmentation. Following previous works [Liu *et al.*, 2021b; Dong *et al.*, 2021], we utilize Mask R-CNN [He *et al.*, 2017] framework under 1x schedule. More details are provided in the supplementary material.

Results

The results on COCO dataset are shown in Table 4(a). All DGT variants outperform the state-of-the-art vision transformer backbones under similar FLOPs. Specifically, for object detection, our DGT-T, DGT-S, and DGT-B can achieve 47.7%, 48.4% and 49.1% box AP, which surpass Swin by 5.5%, 3.6%, and 2.2%, respectively. For instance segmentation, our DGT-T, DGT-S, and DGT-B are 3.9%, 2.6%, and 1.8% mask AP higher than the Swin. Besides, DGT-T outperforms CrossFormer-S by 2.3% bos AP on object detection and 1.6% mask AP on instance segmentation. DGT-S outperforms CrossFormer-B by 1.2% box AP on object detection and 0.8% mask AP on instance segmentation.

4.4 Ablation Studies

We conduct ablation studies for the key designs of our methods on the image classification task. All experiments are performed with the Tiny variant under the same training settings.

Effect of Hyper-parameters G and k

First, we validate the effect of the hyper-parameters G and k. G is the number of groups. A small G causes the queries in a group to be very different, so the selected keys cannot suit all queries. k determines how many keys each query attends to. There will be too much information loss if k is too small. The default setting of our model on ImageNet is $G = 48$ and $k = 98$. We compare the default setting with halving G from 48 to 24 and halving k from 98 to 49.

The results are shown in Table 4(b). Decreasing G or k leads to a poorer performance. Halving G and k decrease the top-1 accuracy by 0.2% and 0.1%, respectively. We can balance the performance and efficiency by adjusting G and k.

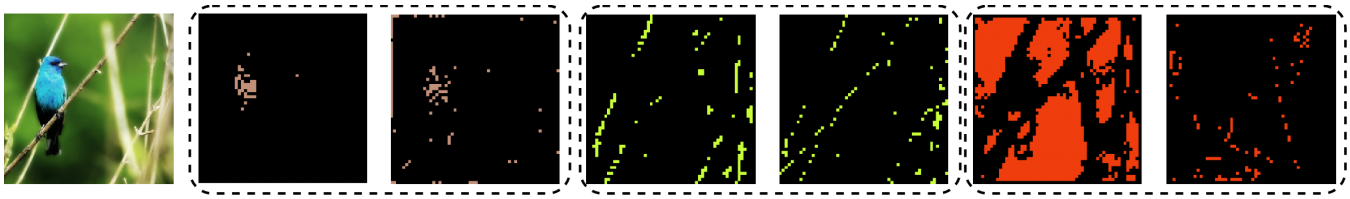


Figure 3: Visualizations of some query and keys groups. Each dashed box contains a query group and its corresponding keys group. It can be seen that our method can flexibly model relevant dependencies without any spatial constraint.

Backbone	Params (M)	FLOPS (G)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
Res50 [He *et al.*, 2016]	44	260	38.0	58.6	41.4	34.4	55.1	36.7
PVT-S [Wang *et al.*, 2021a]	44	245	40.4	62.9	43.8	37.8	60.1	40.3
ViL-S [Zhang *et al.*, 2021]	45	218	44.9	67.1	49.3	41.0	64.2	44.1
TwinsP-S [Chu *et al.*, 2021a]	44	245	42.9	65.8	47.1	40.0	62.7	42.9
Twins-S [Chu *et al.*, 2021a]	44	228	43.4	66.0	47.3	40.3	63.2	43.4
Swin-T [Liu *et al.*, 2021b]	48	264	42.2	64.6	46.2	39.1	61.6	42.0
CrossFormer-S [Wang *et al.*, 2021b]	50	301	45.4	68.0	49.7	41.4	64.8	44.6
DGT-T (ours)	42	272	**47.7**	**69.8**	**52.2**	**43.0**	**66.9**	**46.4**
Res101 [He *et al.*, 2016]	63	336	40.4	61.1	44.2	36.4	57.7	38.8
X101-32 [Xie *et al.*, 2017]	63	340	41.9	62.5	45.9	37.5	59.4	40.2
PVT-M [Wang *et al.*, 2021a]	64	302	42.0	64.4	45.6	39.0	61.6	42.1
ViL-M [Zhang *et al.*, 2021]	60	261	43.4	—	—	39.7	—	—
TwinsP-B [Chu *et al.*, 2021a]	64	302	44.6	66.7	48.9	40.9	63.8	44.2
Twins-B [Chu *et al.*, 2021a]	76	340	45.2	67.6	49.3	41.5	64.5	44.8
Swin-S [Liu *et al.*, 2021b]	69	354	44.8	66.6	48.9	40.9	63.4	44.2
CrossFormer-B [Wang *et al.*, 2021b]	72	408	47.2	69.9	51.8	42.7	66.6	46.2
DGT-S (ours)	70	386	**48.4**	**70.7**	**53.2**	**43.5**	**67.6**	**47.0**
X101-64 [Xie *et al.*, 2017]	101	493	42.8	63.8	47.3	38.4	60.6	41.3
PVT-L [Wang *et al.*, 2021a]	81	364	42.9	65.0	46.6	39.5	61.9	42.5
ViL-B [Zhang *et al.*, 2021]	76	365	45.1	—	—	41.0	—	—
TwinsP-L [Chu *et al.*, 2021a]	81	364	45.4	—	—	41.5	—	—
Twins-L [Chu *et al.*, 2021a]	111	474	45.9	—	—	41.6	—	—
Swin-B [Liu *et al.*, 2021b]	107	496	46.9	—	—	42.3	—	—
DGT-B (ours)	108	540	**49.1**	**70.9**	**54.1**	**44.1**	**68.1**	**47.6**

(a) Comparision with different backbones on COCO. Flops are calculated with the resolution of 800×1280.

G	k	Top-1
24	98	83.6
48	49	83.7
48	98	**83.8**

(b) Effect of Hyper-parameters G and k.

Block	Top-1
Swin block	82.8
CSwin block	83.0
CMT block	83.4
DGT block (ours)	**83.8**

(c) Comparison with different Vision Transformer Blocks.

Table 4: Experiments on COCO and ablation studies.

Comparison with Related Transformer Blocks

To validate the design of our DGT block, which uses convolution layers to extract local dependency and uses the DG-Attention to extract non-local dependency, we replace our DGT block with other blocks and compare their performance. We select three blocks: Swin block, CSwin block, and CMT block. Swin block and Cswin block use shifted window-based self-attention and cross-shaped window self-attention, respectively. The results are shown in Figure 4(c). Our DGT block obviously outperforms Swin block, CSwin block and CMT block by 1.0%, 0.8% and 0.4%.

5 Visualization

We visualize the query groups and their corresponding selected keys with an example shown in Figure 3. One can find : 1) different queries prefer to attends to different keys according to their content. These three groups mainly contain the queries of the bird, branches, and background, and they also attend to the keys of the bird, branches, and background, respectively. 2) A query may prefer to attend to a long-range area rather than a short-range local region. These findings show the advantages of our DG-Attention. More visualization examples can be found in the supplementary material.

6 Conclusion

We have presented an effective dynamic attention mechanism named Dynamic Group Attention (DG-Attention), which dynamically divides input queries into multiple groups and selects relevant keys/values for each group. DG-Attention can model more relevant context dependencies than the previous pre-designed window-based local attention mechanism. Based on the proposed DG-Attention, we have developed a general Vision Transformer backbone, Dynamic Group Transformer (DGT), which can outperform the state-of-the-art on ImageNet-1K for image classification. Furthermore, our DGT outperforms the existing Vision Transformer backbones on ADE20K for semantic segmentation, and COCO for object detection and instance segmentation.

References

[Chu *et al.*, 2021a] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting spatial attention design in vision transformers. *arXiv preprint arXiv:2104.13840*, 2021.

[Chu *et al.*, 2021b] 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.

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

[Dong *et al.*, 2021] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021.

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

[Fang *et al.*, 2021] Jiemin Fang, Lingxi Xie, Xinggang Wang, Xiaopeng Zhang, Wenyu Liu, and Qi Tian. Msg-transformer: Exchanging local spatial information by manipulating messenger tokens. *arXiv preprint arXiv:2105.15168*, 2021.

[Guo *et al.*, 2021] Jianyuan Guo, Kai Han, Han Wu, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Cmt: Convolutional neural networks meet vision transformers. *arXiv preprint arXiv:2107.06263*, 2021.

[He *et al.*, 2016] 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.

[He *et al.*, 2017] 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.

[Huang *et al.*, 2021] Zilong Huang, Youcheng Ben, Guozhong Luo, Pei Cheng, Gang Yu, and Bin Fu. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv preprint arXiv:2106.03650*, 2021.

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

[Liu *et al.*, 2021a] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021.

[Liu *et al.*, 2021b] 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.

[Pan *et al.*, 2021] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. Ia-red^2: Interpretability-aware redundancy reduction for vision transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[Radosavovic *et al.*, 2020] 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.

[Rao *et al.*, 2021] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *arXiv preprint arXiv:2106.02034*, 2021.

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

[Touvron *et al.*, 2020] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[Wang *et al.*, 2021a] 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.

[Wang *et al.*, 2021b] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. *arXiv preprint arXiv:2108.00154*, 2021.

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

[Xiao *et al.*, 2018] 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.

[Xie *et al.*, 2017] 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.

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

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