BiFormer: Vision Transformer with Bi-Level Routing Attention

Lei Zhu[1] Xinjiang Wang[2] Zhanghan Ke[1] Wayne Zhang[2] Rynson Lau[1†]
[1] City University of Hong Kong [2] SenseTime Research
{lzhu68-c,zhanghake2-c}@my.cityu.edu.hk, {wangxinjiang,wayne.zhang}@sensetime.com
Rynson.Lau@cityu.edu.hk

Abstract

*As the core building block of vision transformers, attention is a powerful tool to capture long-range dependency. However, such power comes at a cost: it incurs a huge computation burden and heavy memory footprint as pairwise token interaction across all spatial locations is computed. A series of works attempt to alleviate this problem by introducing handcrafted and content-agnostic sparsity into attention, such as restricting the attention operation to be inside local windows, axial stripes, or dilated windows. In contrast to these approaches, we propose a novel dynamic sparse attention via bi-level routing to enable a more flexible allocation of computations with content awareness. Specifically, for a query, irrelevant key-value pairs are first filtered out at a coarse region level, and then fine-grained token-to-token attention is applied in the union of remaining candidate regions (*i.e.*, routed regions). We provide a simple yet effective implementation of the proposed bi-level routing attention, which utilizes the sparsity to save both computation and memory while involving only GPU-friendly dense matrix multiplications. Built with the proposed bi-level routing attention, a new general vision transformer, named BiFormer, is then presented. As BiFormer attends to a small subset of relevant tokens in a **query adaptive** manner without distraction from other irrelevant ones, it enjoys both good performance and high computational efficiency, especially in dense prediction tasks. Empirical results across several computer vision tasks such as image classification, object detection, and semantic segmentation verify the effectiveness of our design. Code is available at* https://github.com/rayleizhu/BiFormer.

1. Introduction

Transformer has many properties that are suitable for building powerful data-driven models. First, it is able to capture long-range dependency in the data [29, 42]. Second, it is almost inductive-bias-free and thus makes the model more flexible to fit tons of data [15]. Last but not least, it enjoys high parallelism, which benefits training and inference of large models [13, 33, 36, 42]. Hence, transformer has not only revolutionized natural language processing but also shown very promising progress in computer vision.

The computer vision community has witnessed an explosion of vision transformers in the past two years [1, 14, 15, 29, 44, 46]. Among these works, a popular topic is to improve the core building block, *i.e.*, attention. In contrast to convolution, which is intrinsically a local operator, a crucial property of attention is the global receptive field, which empowers vision transformers to capture long-range dependency [42]. However, such a property comes at a cost: as attention computes pairwise token affinity across all spatial locations, it has a high computational complexity and incurs heavy memory footprints.

To alleviate the problem, a promising direction is to introduce sparse attention [6] to vision transformers, so that each query attends to a small portion of key-value pairs instead of all. In this fashion, several *handcrafted* sparse patterns have been explored, such as restricting attention in local windows [29], dilated windows [41, 46], or axial stripes [46]. On the other hand, there are also works trying to make the sparsity adaptive to data [5, 48]. However, while they use different strategies to merge or select key/value tokens, these tokens are query-agnostic, *i.e.*, they are shared by all queries. Nonetheless, according to the visualization of pretrained ViT [1] [15] and DETR [2] [1], queries in different semantic regions actually attend to quite different key-value pairs. Hence, forcing all queries to attend to the same set of tokens may be suboptimal.

In this paper, we seek an attention mechanism with dynamic, query-aware sparsity. Basically, we aim for each query to attend to a small portion of the *most semantically relevant* key-value pairs. The first problem comes as how

[†] Corresponding author.

[1] https://epfml.github.io/attention-cnn/

[2] https://colab.research.google.com/github/facebookresearch/detr/blob/colab/notebooks/detr_attention.ipynb

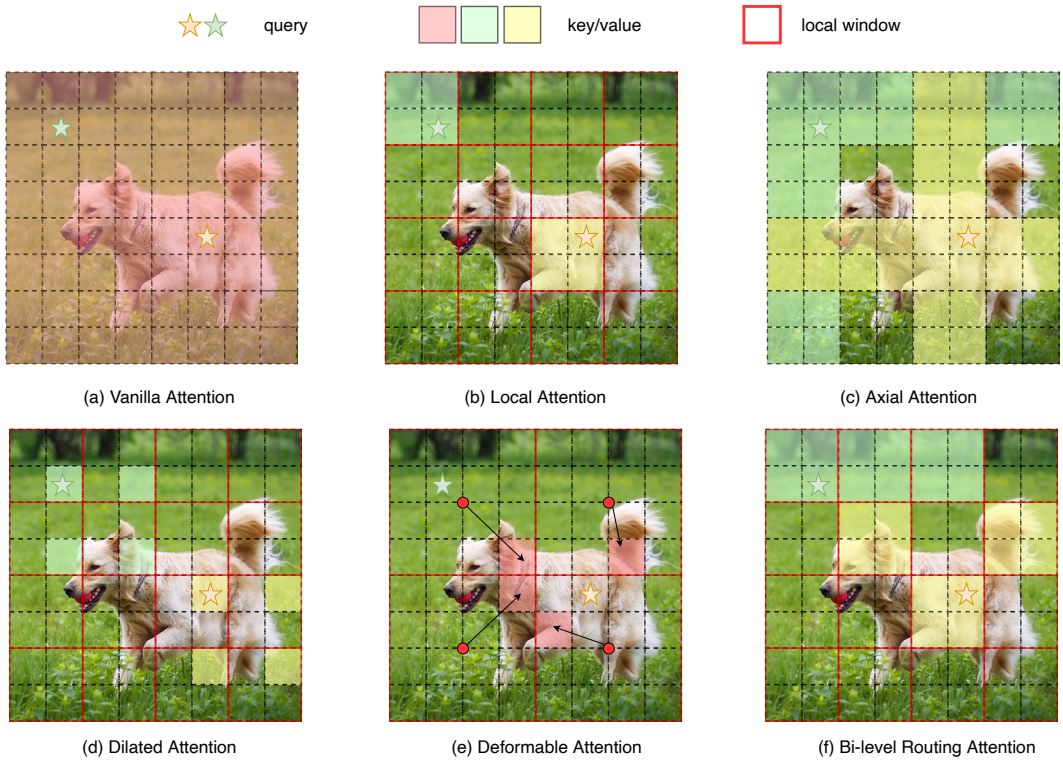

Figure 1. Vanilla attention and its sparse variants. **(a)** Vanilla attention operates gloabally and incurs high computational complexity and heavy memory footprint. **(b)-(d)** Several works attempt to alleviate the complexity by introducing sparse attention with different *handcrafted* patterns, such as local window [29, 46], axial stripe [14], dilated window [41, 46]. **(e)** Deformable attention [48] enables image-adaptive sparsity via deforming a regular grid. **(f)** We achieve dynamic, query-aware sparsity with bi-level routing attention, which first searches top-k ($k = 3$ in this case) relevant regions, and then attends to the union of them.

to locate these key-value pairs to attend. For example, if we select key-value pairs in a per-query manner as done in [17], it still requires evaluation of pairwise affinity between all queries and keys, and hence has the same complexity of vanilla attention. Another possibility is to predict attention offsets based on *local context* for each query [10, 48], and hence pairwise affinity computation is avoided. However, in this way, it is problematic to model long-range dependency [48].

To locate valuable key-value pairs to attend globally with high efficiency, we propose a region-to-region routing approach. Our core idea is to filter out the most irrelevant key-value pairs at a coarse-grained region level, instead of directly at the fine-grained token level. This is done by first constructing a region-level affinity graph and then pruning it to keep only top-k connections for each node. Hence, each region only needs to attend to the top-k routed regions. With the attending regions determined, the next step is to apply token-to-token attention, which is non-trivial as key-value pairs are now assumed to be spatially scattered. For this case, while the sparse matrix multiplication is applicable, it is inefficient in modern GPUs, which rely on coalesced memory operations, *i.e.*, accessing blocks of dozens of contiguous bytes at once [31]. Instead, we propose a simple solution via gathering key/value tokens, where only hardware-friendly dense matrix multiplications are involved. We refer to this approach as **Bi**-level **R**outing **A**ttention (BRA), as it contains a region-level routing step and a token-level attention step.

By using BRA as the core building block, we propose BiFormer, a general vision transformer backbone that can be used for many applications such as classification, object detection, and semantic segmentation. As BRA enables BiFormer to attend to a small subset of the most relevant key/-value tokens for each query in a content-aware manner, our model achieves a better computation-performance trade-off. For example, with 4.6G FLOPs computation, BiFormer-T achieves 83.8% top-1 accuracy on ImageNet-1K classification, which is the best as far as we know under similar computation budgets without training with external data or distillation [23, 40]. The improvements are also consistently shown in downstream tasks such as instance segmentation and semantic segmentation.

To summarize, our contributions are as follows. We introduce a novel bi-level routing mechanism to vanilla attention, which enables content-aware sparse patterns in a query-adaptive manner. Using the bi-level routing attention as the basic building block, we propose a general vi-

sion transformer named BiFormer. Experimental results on various computer vision tasks including image classification, object detection, and semantic segmentation show that the proposed BiFormer achieves significantly better performances over the baselines under similar model sizes.

2. Related Works

Vision transformers. Transformers are a family of neural networks that adopt channel-wise MLP blocks for per-location embedding (channel mixing) and attention [42] blocks for cross-location relation modeling (spatial mixing). Transformers were originally proposed for natural language processing [13, 42] and then introduced to computer vision by pioneering works such as DETR [1] and ViT [15]. In comparison with CNNs, the biggest difference is that transformers use attention as an alternative to convolution to enable global context modeling. However, as vanilla attention computes pairwise feature affinity across all spatial locations, it incurs a high computation burden and heavy memory footprints, especially for high-resolution inputs. Hence, an important research direction is to seek more efficient attention mechanisms.

Efficient attention mechanisms. A large volume of works have been proposed to reduce the computation and memory complexity bottlenecks of vanilla attention by utilizing sparse connection patterns [6], low-rank approximations [43] or recurrent operations [11]. A thorough survey of these attention variants can be found at [39]. In the scope of vision transformers, sparse attention gains its popularity recently due to the tremendous success of Swin transformer [29]. In Swin transformer, attention is restricted to non-overlapping local windows, and the shift window operation is introduced to enable inter-window communication between *adjacent* windows. To enable larger and even quasi-global receptive fields under a reasonable computation budget, several follow-up works introduce different handcrafted sparse patterns, such as dilated windows [41, 46] or cross-shaped windows [14]. There are also works that try to make the sparse pattern adaptive to data, such as DAT [48], TCFormer [53] and DPT [5]. While these works reduce the number of key/value tokens via different merging or selection strategies, these key/value tokens are shared by all queries on an image. Instead, we explore query-aware key/value token selection. The key observation which motivates our work is that the attentive region for different queries may differ significantly according to the visualization of pretrained ViT [15] and DETR [1]. As we achieve the goal of query-adaptive sparsity in a coarse-to-fine manner, it shares some similarities with quad-tree attention [38]. Different from quad-tree attention, the goal of our bi-level routing attention is to locate a few most relevant key-value pairs, while quad-tree attention builds a token pyramid and assembles messages from all levels of different granularities. In addition, the quad-tree requires deep recursion to cover the whole feature map, which hurts parallelism, while our bi-level routing attention can be more efficiently implemented by key/value token gathering, followed by dense matrix multiplications. As a result, quad-tree transformer is much slower than our BiFormer.

3. Our Approach: BiFormer

This section elaborates the proposed approach. We start by briefly summarizing the attention mechanism in Section 3.1. We then introduce our novel bi-level routing attention (BRA) mechanism, which enables dynamic and query-adaptive sparsity, in Section 3.2. We further show that BRA can achieve $O((HW)^{\frac{4}{3}})$ complexity with a proper region partition size in Section 3.3. Finally, using BRA as the core building block, we present a new hierarchical vision transformer, named BiFormer, in Section 3.4.

3.1. Preliminaries: Attention

Taking queries $\mathbf{Q} \in \mathbb{R}^{N_q \times C}$, keys $\mathbf{K} \in \mathbb{R}^{N_{kv} \times C}$, and values $\mathbf{V} \in \mathbb{R}^{N_{kv} \times C}$ as input, an attention function transforms each query as a weighted sum of values, where the weights are computed as normalized dot products between the query and corresponding keys. It can be formally defined in a compact matrix form, as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{C}}\right)\mathbf{V}. \quad (1)$$

Here, the scalar factor $\sqrt{C}$ is introduced to avoid concentrated weights and gradient vanishing [42].

In transformers, the de facto building block used is multi-head self-attention (MHSA). By "self-attention", it means that queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ are derived as linear projections of the same input $\mathbf{X} \in \mathbb{R}^{N \times C}$. (For vision transformers, $\mathbf{X}$ is a spatially flattened feature map, *i.e.*, $N = H \times W$, where H and W are the height and width, respectively, of the feature map.) As for "multi-head", it implies splitting the output into h chunks (*i.e.*, heads) along the channel dimension with each chunk using an independent group of projection weights. Formally,

$$\begin{aligned} \text{MHSA}(\mathbf{X}) &= \text{Concat}(\mathbf{head}_0, \mathbf{head}_1, ..., \mathbf{head}_h)\mathbf{W^o}, \\ \mathbf{head}_i &= \text{Attention}(\mathbf{X}\mathbf{W}_i^q, \mathbf{X}\mathbf{W}_i^k, \mathbf{X}\mathbf{W}_i^v), \end{aligned} \quad (2)$$

where $\mathbf{head}_i \in \mathbb{R}^{N \times \frac{C}{h}}$ is the output of the i^{th} attention head. $\mathbf{W}_i^q, \mathbf{W}_i^k, \mathbf{W}_i^v \in \mathbb{R}^{C \times \frac{C}{h}}$ are corresponding input projection weights. An extra linear transformation with weight matrix $\mathbf{W}^o \in \mathbb{R}^{C \times C}$ is used to compose all heads.

MHSA has a complexity of $O(N^2)$, as there are N queries and each query will attend to N key-value pairs. Such a high complexity causes severe scalability issues w.r.t. the spatial resolution of the inputs.

Algorithm 1 Pseudocode of BRA in a PyTorch-like style.

```
# input: features (H, W, C). Assume H==W.
# output: features (H, W, C).
# S: square root of number of regions.
# k: number of regions to attend.

# patchify input (H, W, C) -> (S^2, HW/S^2, C)
x = patchify(input, patch_size=H//S)

# linear projection of query, key, value
query, key, value = linear_qkv(x).chunk(3, dim=-1)

# regional query and key (S^2, C)
query_r, key_r = query.mean(dim=1), key.mean(dim=1)

# adjacency matrix for regional graph (S^2, S^2)
A_r = mm(query_r, key_r.transpose(-1, -2))

# compute index matrix of routed regions (S^2, K)
I_r = topk(A_r, k).index

# gather key-value pairs
key_g = gather(key, I_r) # (S^2, kHW/S^2, C)
value_g = gather(value, I_r) # (S^2, kHW/S^2, C)

# token-to-token attention
A = bmm(query, key_g.transpose(-2, -1))
A = softmax(A, dim=-1)
output = bmm(A, value_g) + dwconv(value)

# recover to (H, W, C) shape
output = unpatchify(output, patch_size=H//S)
```

`bmm`: batch matrix multiplication; `mm`: matrix multiplication. `dwconv`: depthwise convolution.

3.2. Bi-Level Routing Attention (BRA)

To mitigate the scalability issue of MHSA, several works [14, 29, 41, 46, 48] propose different sparse attention mechanisms, in which each query attends to only a small number of key-value pairs instead of all. However, these existing works either use handcrafted static patterns or share the sampled subset of key-value pairs among all queries, as shown in Figure 1. In this work, we explore a dynamic, query-aware sparse attention mechanism. Our key idea is to filter out most irrelevant key-value pairs in a coarse region level so that only a small portion of routed regions remain. We then apply fine-grained token-to-token attention in the union of these routed regions. To simplify the notations, we discuss the case of single-head self-attention with a single input, although we use multi-head self-attention [42] with batched input in practice. The whole algorithm is summarized with Pytorch-like [32] pseudo code in Algorithm 1. We give a detailed explanation as follows.

Region partition and input projection. Given a 2D input feature map $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$, we start by dividing it into $S\times S$ non-overlapped regions such that each region contains $\frac{HW}{S^2}$ feature vectors. This step is done by reshaping $\mathbf{X}$ as $\mathbf{X}^r \in \mathbb{R}^{S^2\times \frac{HW}{S^2}\times C}$. We then derive the query, key, value tensor, $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{S^2\times \frac{HW}{S^2}\times C}$, with linear projections:

$$\mathbf{Q} = \mathbf{X}^r\mathbf{W}^q, \quad \mathbf{K} = \mathbf{X}^r\mathbf{W}^k, \quad \mathbf{V} = \mathbf{X}^r\mathbf{W}^v, \tag{3}$$

where $\mathbf{W}^q, \mathbf{W}^k, \mathbf{W}^v \in \mathbb{R}^{C\times C}$ are projection weights for the query, key, value, respectively.

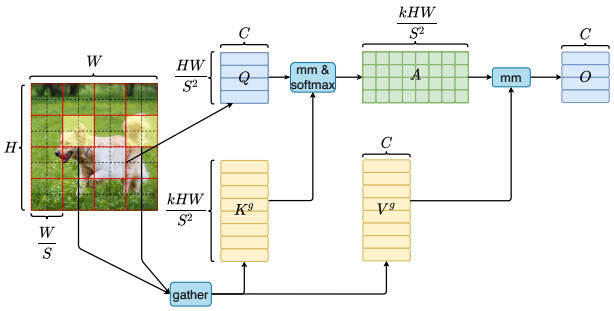

Figure 2. By gathering key-value pairs in top k related windows, we utilize the sparsity to skip computations in the most irrelevant regions, while only GPU-friendly dense matrix multiplications are involved.

Region-to-region routing with directed graph. We then find the attending relationship (*i.e.*, the regions that should be attended for each given region) by constructing a directed graph. Specifically, we first derive region-level queries and keys, $\mathbf{Q}^r, \mathbf{K}^r \in \mathbb{R}^{S^2\times C}$, via applying per-region average on $\mathbf{Q}$ and $\mathbf{K}$, respectively. We then derive the adjacency matrix, $\mathbf{A}^r \in \mathbb{R}^{S^2\times S^2}$, of region-to-region affinity graph via matrix multiplication between $\mathbf{Q}^r$ and transposed $\mathbf{K}^r$:

$$\mathbf{A}^r = \mathbf{Q}^r(\mathbf{K}^r)^T. \tag{4}$$

Entries in the adjacency matrix, $\mathbf{A}_r$, measure how much two regions are semantically related. The core step that we perform next is to prune the affinity graph by keeping only top-k connections for each region. Specifically, we derive a routing index matrix, $\mathbf{I}_r \in \mathbb{N}^{S^2\times k}$, with the row-wise topk operator:

$$\mathbf{I}^r = \text{topkIndex}(\mathbf{A}^r). \tag{5}$$

Hence, the i^{th} row of $\mathbf{I}^r$ contains k indices of most relevant regions for the i^{th} region.

Token-to-token attention. With the region-to-region routing index matrix $\mathbf{I}^r$, we can then apply fine-grained token-to-token attention. For each query token in region i, it will attend to all key-value pairs residing in the union of k routed regions indexed with $\mathbf{I}^r_{(i,1)}, \mathbf{I}^r_{(i,2)}, ..., \mathbf{I}^r_{(i,k)}$. However, it is non-trivial to implement this step efficiently, as these routed regions are expected to be scattered over the whole feature map, while modern GPUs rely on coalesced memory operations that load blocks of dozens of contiguous bytes at once. We thus gather key and value tensor first, *i.e.*,

$$\mathbf{K}^g = \text{gather}(\mathbf{K}, \mathbf{I}^r), \quad \mathbf{V}^g = \text{gather}(\mathbf{V}, \mathbf{I}^r), \tag{6}$$

where $\mathbf{K}^g, \mathbf{V}^g \in \mathbb{R}^{S^2\times \frac{kHW}{S^2}\times C}$ are gathered key and value tensor. We can then apply attention on the gathered key-value pairs as:

$$\mathbf{O} = \text{Attention}(\mathbf{Q}, \mathbf{K}^g, \mathbf{V}^g) + \text{LCE}(\mathbf{V}). \tag{7}$$

Here, we introduce a local context enhancement term $\mathrm{LCE}(\mathbf{V})$ as in [37]. Function $\mathrm{LCE}(\cdot)$ is parametrized with a depth-wise convolution, and we set the kernel size to 5.

3.3. Complexity Analysis of BRA

The proposed bi-level routing attention enables direct long-range dependency modeling similar to vanilla attention. However, we show here that BRA has a much lower complexity of $O((HW)^{\frac{4}{3}})$ with a proper region partition factor S compared to vanilla attention, which has a complexity of $O((HW)^2)$, and to quasi-global axial attention [14, 22], which has a complexity of $O((HW)^{\frac{3}{2}})$.

The computation of BRA consists of three parts: linear projection, region-to-region routing, and token-to-token attention. The total amount of computations is therefore:

$$\begin{aligned}\mathrm{FLOPs} &= \mathrm{FLOPs}_{proj} + \mathrm{FLOPs}_{routing} + \mathrm{FLOPs}_{attn} \\ &= 3HWC^2 + 2(S^2)^2C + 2HWk\frac{HW}{S^2}C \\ &= 3HWC^2 + C(2S^4 + \frac{k(HW)^2}{S^2} + \frac{k(HW)^2}{S^2}) \\ &\geq 3HWC^2 + 3C(2S^4 \cdot \frac{k(HW)^2}{S^2} \cdot \frac{k(HW)^2}{S^2})^{\frac{1}{3}} \\ &= 3HWC^2 + 3Ck^{\frac{2}{3}}(2HW)^{\frac{4}{3}},\end{aligned} \tag{8}$$

where C is the token embedding dimension (*i.e.*, number of channels of the feature map), and k is the number of regions to attend ("k" in "top-k"). Here, the inequality of arithmetic and geometric means has been applied. The equality in Eq. 8 holds if and only if $2S^4 = \frac{k(HW)^2}{S^2}$. Therefore:

$$S = (\frac{k}{2}(HW)^2)^{\frac{1}{6}}. \tag{9}$$

In other words, BRA achieves $O((HW)^{\frac{4}{3}})$ complexity if we scale the region partition factor S w.r.t. the input resolution according to Eq. 9.

3.4. Architecture Design of BiFormer

Using BRA as a basic building block, we propose a new general vision transformer, BiFormer. As shown in Figure 3, we follow the recent state-of-the-art vision transformers [14, 29, 41] to use a four-stage pyramid structure. Specifically, in stage i, we use an overlapped patch embedding in the first stage and a patch merging module [25, 37] in the second to fourth stages to reduce the input spatial resolution while increasing the number of channels, followed by N_i consecutive BiFormer blocks to transform the features. In each BiFormer block, we follow recent works [7, 25, 41] to use a 3×3 depthwise convolution at the beginning to encode relative position information implicitly. We then apply a BRA module and 2-layer MLP module with expansion ratio e sequentially for cross-location relation modeling and per-location embedding, respectively.

We instantiate BiFormer with 3 different model sizes by scaling the network width (*i.e.*, the number of base channels C) and depth (*i.e.*, the number of BiFormer blocks used in each stage, $N_i, i = 1, 2, 3, 4$), as listed in Table 1. They share other configurations. We set each attention head to 32 channels, and MLP expansion ratio e=3. For BRA, we use $topk = 1, 4, 16, S^2$[3] for the 4 stages, and region partition factor $S = 7/8/16$ for classification/semantic segmentation/object detection task, due to different input resolutions.

Models	#Channels.	#Blocks	Params	FLOPs
BiFormer-T	64	[2, 2, 8, 2]	13M	2.2G
BiFormer-S	64	[4, 4, 18, 4]	26M	4.5G
BiFomrer-B	96	[4, 4, 18, 4]	57M	9.8G

Table 1. Network width and depth of different model variants. The FLOPs are calculated with 224×224 input.

Model	FLOPs (G)	Params (M)	Top-1 Acc. (%)
ResNet-18 [19]	1.8	11.7	69.8
RegNetY-1.6G [34]	1.6	11.2	78.0
PVTv2-b1 [45]	2.1	13.1	78.7
Shunted-T [37]	2.1	11.5	79.8
QuadTree-B-b1 [38]	2.3	13.6	80.0
BiFormer-T	2.2	13.1	**81.4**
Swin-T [29]	4.5	29	81.3
CSWin-T [14]	4.5	23	82.7
DAT-T [48]	4.6	29	82.0
CrossFormer-S [46]	5.3	31	82.5
RegionViT-S [2]	5.3	31	82.6
QuadTree-B-b2 [38]	4.5	24	82.7
MaxViT-T [41]	5.6	31	83.6
ScalableViT-S [50]	4.2	32	83.1
Uniformer-S*	4.2	24	83.4
Wave-ViT-S* [51]	4.7	23	83.9
BiFormer-S	4.5	26	**83.8**
BiFormer-S*	4.5	26	**84.3**
Swin-B [29]	15.4	88	83.5
CSWin-B [14]	15.0	78	84.2
CrossFormer-L [46]	16.1	92	84.0
ScalableViT-B [50]	8.6	81	84.1
Uniformer-B* [25]	8.3	50	85.1
Wave-ViT-B* [51]	7.2	34	84.8
BiFormer-B	9.8	57	**84.3**
BiFormer-B*	9.8	58	**85.4**

Table 2. Comparison of different backbones on ImageNet-1K. All models are trained and evaluated on images of resolution 224×224. "*" indicates that the model is trained with token labeling [23]. Methods are grouped by the amount of computations.

[3] In the final stage, $topk = S^2$ means that we use full self-attention.

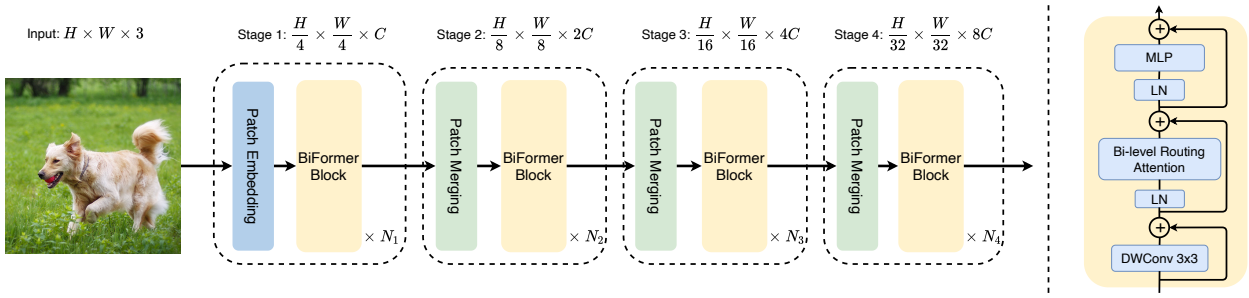

Figure 3. **Left**: The overall architecture of our BiFormer. Refer to Table 1 for configurations. **Right**: Details of a BiFormer Block.

4. Experiments

We evaluate the effectiveness of our proposed BiFormer experimentally on a series of mainstream computer vision tasks including image classification (Sec. 4.1), object detection and instance segmentation (Sec. 4.2), and semantic segmentation (Sec. 4.3). Specifically, we train from scratch on ImageNet1K [12] for image classification. We then finetune the pretrained backbones on COCO [28] for object detection and instance segmentation, and on ADE20K [55] for semantic segmentation. Additionally, we conduct ablation study to verify the effectiveness of the proposed bi-level routing attention and other architecture design choices of BiFormer in Sec. 4.4. Finally, to verify query-adaptive, sparse patterns are achieved by bi-level routing attention, we visualize the attention map in Sec. 4.5.

4.1. Image Classification on ImageNet-1K

Settings. We conduct image classification experiments on the ImageNet-1K [12] dataset, following the experimental settings of DeiT [40] for fair comparison. Specifically, each model is trained 300 epochs with input size of 224×224. We take AdamW as the optimizer with weight decay of 0.05, and apply cosine decay learning rate schedule with an initial learning rate of 0.001, while the first 5 epochs are utilized for linear warm-up [16]. The batch size is set to 1024. To avoid overfitting, we apply regularization techniques including RandAugment [9] (rand-m9-mstd0.5-inc1), MixUp [54] ($prob = 0.8$), CutMix [52] ($prob = 1.0$), Random Erasing ($prob = 0.25$), and increasing stochastic depth [21] ($prob = 0.1/0.15/0.4$ for BiFormer-T/S/B, respectively). To fairly compare the models trained with token labeling [23], including Uniformer [25] and WaveViT [51], we also provide a version trained with the same recipe provided by WaveViT.

Results. We compare our method with several closely related methods and/or recent state-of-the-arts. Quantitative results are listed in Table 2, where models are grouped by the amount of computations (FLOPs). In all 3 groups, our model consistently outperforms other compared ones. For example, for models in the smallest group (∼2G FLOPs), our BiFormer-T achieves 81.4% top-1 accuracy, 1.4% better than the most competitive QuadTree-b1 [38]. For models in the second group (∼4G FLOPs), BiFormer-S achieves 83.8% top-1 accuracy. To the best of our knowledge, this is the best result without extra training data or training tricks. In addition, using the distillation technique named token labeling [23], the accuracy of BiFormer-S can be further boosted to 84.3%, which implies that there is a huge potential for the proposed architecture. For models in the largest group (∼10G FLOPs), BiFormer-B achieves an even better performance than those of existing models with the amount of computations reaching up to ∼16G FLOPs, such as Swin-B [29], CSWin-B [14] and CrossFormer-L [46].

4.2. Object Detection and Instance Segmentation

Settings. We evaluate the models for object detection and instance segmentation on COCO 2017 [28]. For a fair comparison, all experiments are conducted with the MMDetection [3] toolbox. RetinaNet [27] and Mask R-CNN [18] frameworks are used for object detection and instance segmentation, respectively. Before training on COCO, we initialize the backbone with weights pretrained on ImageNet-1K, while leaving all other layers randomly initialized. The models are trained with the standard $1\times$ schedule (12 epochs) provided by MMDetection, except that we use the AdamW optimizer [30], instead of SGD. We use an initial learning rate of $1e-4$, and a batch size of 16, while the weight decay is set as $1e-4$ and $5e-2$ for RetinaNet and Mask R-CNN, respectively. During training, we resize the input images by fixing the shorter side to 800 pixels while keeping the longer side not exceeding 1,333 pixels.

Results. We list results in Table 3. For object detection with RetinaNet, we report mean Average Precision (mAP), Average Precision (AP) at different IoU thresholds (50%, 75%) and for three object sizes (*i.e.* small, medium, and large (S/M/L)). From the results, we can see that while the overall performance of BiFormer is only comparable to some most competitive existing methods, such as WaveViT and QuadTree-B, the performance on small objects (AP_M) outperforms these methods significantly. This may be because the BRA saves computations via sparse sampling instead of downsampling. Hence, it preserves fine-grained details, which are crucial for small objects. For instance seg-

Backbone	RetinaNet 1× schedule						Mask R-CNN 1× schedule					
	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	mAP^b	AP^b_{50}	AP^b_{75}	mAP^m	AP^m_{50}	AP^m_{75}
Swin-T [29]	41.5	62.1	44.2	25.1	44.9	55.5	42.2	64.6	46.2	39.1	61.6	42.0
DAT-T [48]	42.8	64.4	45.2	28.0	45.8	57.8	44.4	67.6	48.5	40.4	64.2	43.1
CSWin-T [14]	-	-	-	-	-	-	46.7	68.6	51.3	42.2	65.6	45.4
CrossFormer-S [46]	44.4	55.3	38.6	19.3	40.0	48.8	45.4	68.0	49.7	41.4	64.8	44.6
QuadTree-B2 [38]	46.2	67.2	49.5	29.0	50.1	61.8	-	-	-	-	-	-
WaveViT-S* [51]	45.8	67.0	49.4	29.2	50.0	60.8	46.6	68.7	51.2	42.4	65.5	45.8
BiFormer-S	45.9	66.9	49.4	**30.2**	49.6	61.7	**47.8**	**69.8**	**52.3**	**43.2**	**66.8**	**46.5**
Swin-S [29]	44.5	65.7	47.5	27.4	48.0	59.9	44.8	66.6	48.9	40.9	63.4	44.2
DAT-S [48]	45.7	67.7	48.5	30.5	49.3	61.3	47.1	69.9	51.5	42.5	66.7	45.4
CSWin-S [14]	-	-	-	-	-	-	47.9	70.1	52.6	43.2	67.1	46.2
CrossFormer-B [46]	46.2	67.8	49.5	30.1	49.9	61.8	47.2	69.9	51.8	42.7	66.6	46.2
QuadTree-B3 [38]	47.3	68.2	50.6	30.4	51.3	62.9	-	-	-	-	-	-
Wave-ViT-B* [51]	47.2	68.2	50.9	29.7	51.4	62.3	47.6	69.1	52.4	43.0	66.4	46.0
BiFormer-B	47.1	**68.5**	50.4	**31.3**	50.8	62.6	**48.6**	**70.5**	**53.8**	**43.7**	**67.6**	**47.1**

Table 3. Comparison based on the object detection (left group) and instance segmentation (right group) tasks, on the COCO 2017 dataset.

Backbone	S-FPN	Upernet	
	mIoU(%)	mIoU(%)	MS mIOU(%)
Swin-T [29]	41.5	44.5	45.8
DAT-T [48]	42.6	45.5	46.4
CSWin-T [14]	48.2	49.3	50.7
CrossFormer-S [46]	46.0	47.6	48.4
Shunted-S [37]	48.2	48.9	49.9
WaveViT-S* [51]	-	-	49.6
BiFormer-S	**48.9**	**49.8**	**50.8**
Swin-S [29]	-	47.6	49.5
DAT-S [48]	46.1	48.3	49.8
CSWin-S [14]	49.2	50.4	51.5
CrossFormer-B [46]	47.7	49.7	50.6
Uniformer-B [25]	48.0	50.0	50.8
WaveViT-B* [51]	-	-	51.5
BiFormer-B	**49.9**	**51.0**	**51.7**

Table 4. Comparison based on semantic segmentation with two segmentation heads (Semantic FPN and UpperNet), on ADE20K.

mentation with Mask R-CNN, we report bounding box and mask Average Precision (AP^b and AP^m) at different IoU thresholds (50%, 75%). As shown in Table 3, our method shows a clear advantage in this task on all metrics.

4.3. Semantic Segmentation on ADE20K

Settings. Following existing works, we conduct our semantic segmentation experiments on the ADE20K [55] dataset based on MMSegmentation [8]. We do comparisons under both Semantic FPN [24] and UperNet [49] frameworks. In both cases, the backbone is initialized with ImageNet-1K pretrained weights, and other layers use random initialization. Models are optimized with the AdamW optimizer and the batch size is set as 32. For a fair comparison, our Semantic FPN experiments use the same setting as PVT [44] to train the model 80k steps. Our Upernet experiments use the same setting as Swin Transformer [29] to train the model 160k iterations.

Results. Table 4 shows the results of the two different frameworks. It shows that with the Semantic FPN framework, our BiFormer-S/B achieves 48.9/49.9 mIoU, respectively, improving CSWin-T/S by 0.7 mIoU. A similar performance gain for the UperNet framework is also observed.

Sparse Attention	IN1K	ADE20K
	Top1(%)	mIoU(%)
Sliding window [35]	81.4	-
Shifted window [29]	81.3	41.5
Spatially Sep [7]	81.5	42.9
Sequential Axial [20]	81.5	39.8
Criss-Cross [22]	81.7	43.0
Cross-shaped window [14]	82.2	43.4
Deformable [48]	82.0	42.6
Block-Grid [41]	81.8	42.8
Bi-level Routing	**82.7**	**44.8**

Table 5. Ablation study on different attention mechanisms. All models follow the architecture design of the Swin-T model.

4.4. Ablation Study

The effectiveness of BRA. We compare BRA with several existing sparse attention mechanisms. Following [14], we align macro architecture designs with Swin-T [29] for a fair comparison. Specifically, we use 2, 2, 6, 2 blocks for the four stages, non-overlapped patch embedding, set the initial patch embedding dimension $C = 96$ and MLP expansion ratio $e = 4$. The results are reported in Table 5. Our bi-level routing attention has significantly better performance than existing sparse attention mechanisms, in terms of both image classification and semantic segmentation.

Other architecture design choices. Using the Swin-T lay-

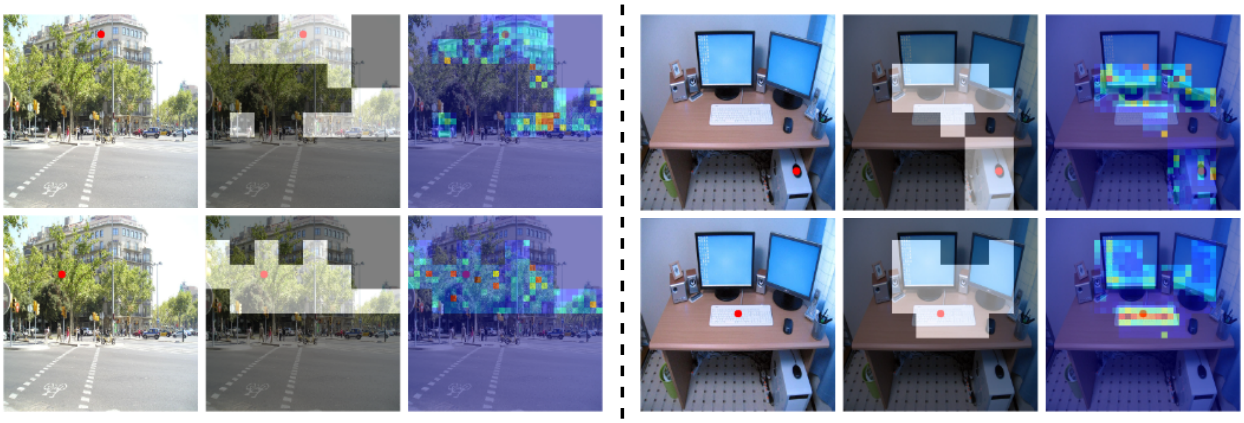

Figure 4. Visualization of the attention maps for two scenes. For each scene, we visualize two query positions on the input image (left), corresponding routed regions (middle), and a final attention heatmap (right).

out as the baseline, we present a summary of other modifications that we have applied, which further boost our BiFormer-S model to state-of-the-art performances on the ImageNet-1K dataset. These modifications include: (1) replacing non-overlapped patch embedding [29] with overlapped one [14,37,45], (2) using deeper layout (*i.e.* stacking more blocks in each stage, while reducing the base channels from 96 to 64 and MLP expansion ratio from 4 to 3 to keep similar FLOPs.), (3) adding convolution position encoding [7, 25] at the beginning of the BiFormer blocks, and (4) applying token labeling [23, 25, 51] training technique. As shown in Table 6, simply using a deeper layout can improve the performance significantly. However, this factor is usually not discussed in existing works.

Architecture design	Params (M)	FLOPs (G)	IN1K Top1 (%)
Baseline (Swin-T layout)	29	4.6	82.7
+Overlapped patch emb.	31	4.9	82.8 (+0.1)
+Deeper layout	25	4.5	83.5 (+0.7)
+Convolution pos. enc.	26	4.5	83.8 (+0.3)
+Token Labling	29	4.9	84.3 (+0.5)

Table 6. Ablation path from Swin-T [29] layout architecture to BiFormer-S. Note that the modifications are applied sequentially.

4.5. Visualization of Attention Map

To further understand how bi-level routing attention works, we visualize routed regions and attention response w.r.t. query positions. For this visualization, we use the routing indices and attention scores extracted from the final BiFormer block of the 3^{rd} stage, which is the major stage consuming most computations. We demonstrate two scenes in Figure 4. In both cases, we can clearly observe that semantically related regions are successfully located. For example, in the first scene, which is a street view, if the query position is on a building or a tree, the corresponding routed regions cover the same or similar entities. In the second indoor scene, when we place the query position on the mouse, the routed regions contain part of the host, keyboard, and display, even though these regions are not adjacent to each other. This implies that our bi-level routing attention can capture long-range inter-object relationships.

5. Limitation and Future Work

Compared to sparse attention with simple static patterns, we introduce an extra step to locate the regions to attend, where we build and prune a region-level graph and gather key-value pairs from the routed regions. While this step does not incur much computation as it operates at a coarse region level, it inevitably incurs extra GPU kernel launch and memory transactions. Hence, BiFormer has lower throughput than some existing models with similar FLOPs on GPU due to overheads of kernel launch and memory bottleneck. Nonetheless, this problem can be mitigated via engineering efforts, such as GPU kernel fusion. We will explore efficient sparse attention and vision transformer with hardware awareness in our future works.

6. Conclusion

We propose bi-level routing attention to enable efficient allocation of computations in a dynamic, query-aware manner. The core idea of BRA is to filter out the most irrelevant key-value pairs at a coarse region level. It is achieved by first building and pruning a region-level directed graph, and then applying fine-grained token-to-token attention in the union of routed regions. We have analyzed the computational complexity of BRA and demonstrated that it achieves $O((HW)^{\frac{4}{3}})$ with a proper region partition size. Using BRA as the core building block, we propose BiFormer, a new vision transformer that has shown superior performances on four popular vision tasks, image classification, object detection, instance segmentation, and semantic segmentation.

Appendix

A. Discussion on Regional Representations

In our proposed bi-level routing attention, we derive the regional representations ($\mathbf{Q}^r$ and $\mathbf{K}^r$) with average pooling for region-to-region routing. We justify the choice here.

In fact, as the goal of region-to-region routing is to find the most related tokens for token-to-token attention in the next step, it is reasonable to maximize *the average token-to-token affinity scores between the two regions*. However, this is equivalent to maximizing *the affinity score between the average tokens of the two regions*, because

$$\frac{1}{|\Omega| \cdot |\Omega'|} \sum_{i \in \Omega} \sum_{j \in \Omega'} \mathbf{Q}_i \mathbf{K}_j = \frac{\sum_{i \in \Omega} \mathbf{Q}_i}{|\Omega|} \cdot \frac{\sum_{j \in \Omega'} \mathbf{K}_j}{|\Omega'|}, \quad (10)$$

where we denote the set of token indices of the two regions with Ω and Ω'.

B. Throughput Comparison

To demonstrate the computation efficiency of the proposed bi-level routing attention, we compare the throughputs of models using different attention mechanisms. Specifically, we replace the shift window attention modules in Swin-T [29] with quad-tree attention [38] modules to form QuadTree-STL, and with our bi-level routing attention modules to form BiFormer-STL. We then use the widely used timm [47] script to benchmark the training and inference throughput on a 32 GB Tesla V100 GPU with a batch size of 128 and image resolution of 224×224.

As shown in Figure 5, Swin-T has the highest throughput due to its simplicity. Switching to our bi-level routing attention(BRA), the training and inference throughput of BiFormer-STL decrease by ∼30% and ∼40% respectively in comparison with Swin-T. This is caused by extra GPU kernel launch and memory transactions caused by the routing process (*i.e.* locating the regions to attend and gather key-value pairs). Nonetheless, BiFormer-STL is still $3\times \sim 6\times$ faster than QuadTree-STL. This is due to that on the one hand the recursive nature of quad-tree attention hurts the parallelism, on the other hand quad-tree attention relies on sparse matrix multiplications which are inefficient on GPUs, while our BRA can be efficiently implemented with key-value token gathering followed by GPU-friendly dense matrix multiplications.

It is worth noting that, the overheads of both memory transactions and kernel launch incurred by the routing process can be reduced via engineering efforts such as GPU kernel fusion. We leave this optimization to our future work.

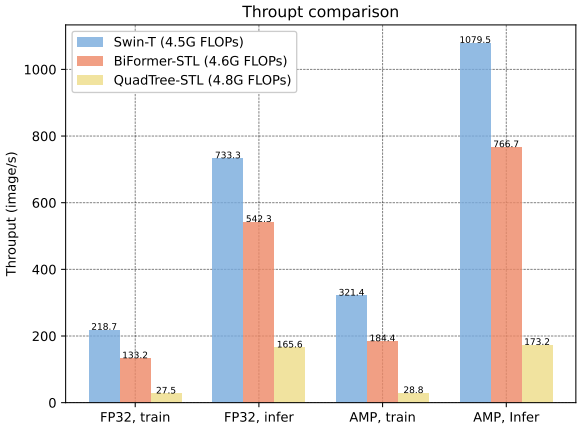

Figure 5. Throughput comparison on a 32GB Tesla V100 GPU. The suffix "STL" denotes **S**win-**T** **L**ayout, which means we use Swin-T [29] backbone with only attention module being replaced. We report results under both FP32 precision and automatic mixed precision (AMP) modes.

S	k	#tokens to attend	Acc	im/s (FP32)
7	1,4,16,49	64,64,64,49	**82.7**	522.3
7	1,2,8,32	64,32,32,32	82.4	563.2
7	2,8,32,49	128,128,128,49	82.6	419.9
8,4,2,1	2,2,2,1	98,98,98,49	82.3	**606.2**

Table 7. Ablation study on top-k and partition factor S.

C. Choices of top-k and partition factor S

In the paper, S and k were chosen more with consideration of engineering issues. **(1)** S is chosen as a divisor of the training size to avoid padding, which slows down the training and may also degrade the performance. For example, in image classification where the resolution is $224 = 7 \times 32$, we use $S = 7$ so that it is a divisor of the size of feature maps in every stage. This is similar to SWinTransformer [29], which uses a window size of 7. **(2)** In dense prediction tasks, we use larger S to balance the complexity of region-level routing and token-level attention to achieve overall lower complexity. One can find hints from Eq. 9 of the paper, though we do not strictly follow the scaling rule due to the size divisor constraint. **(3)** We gradually increase k to keep a reasonable number of tokens to attend as the region size becomes smaller in later stages.

It is possible to try different combinations of S and k. We show ablation results on IN-1K in Table 7, based on BiFormer-STL (as in the paper). A key observation from these experiments is that increasing the number of tokens to attend may even hurt the accuracy. This implies the explicit sparsity constraint may serve as a regularization to avoid distractions from the background.

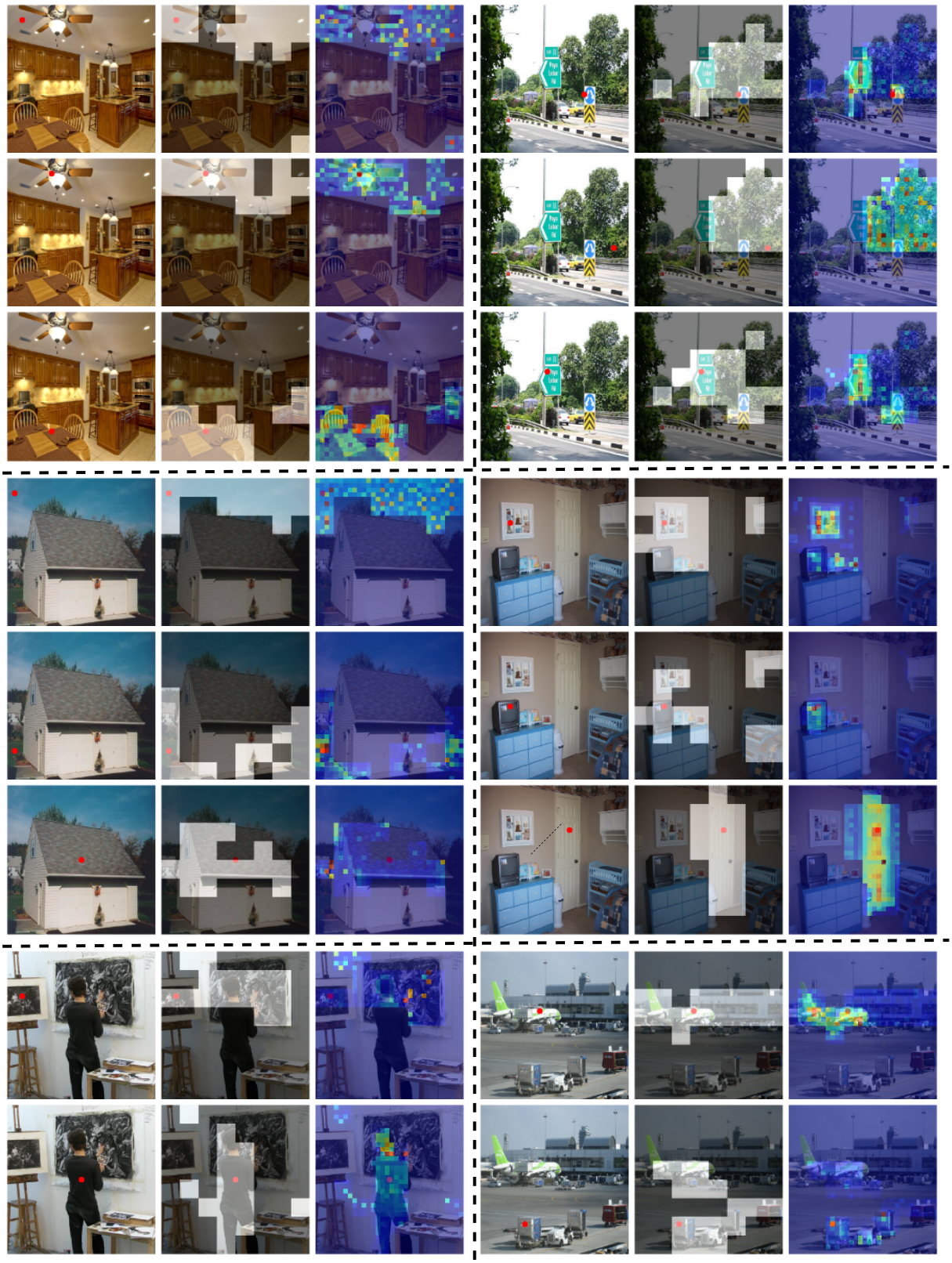

Figure 6. More attention map visualization results. For each scene, We demonstrate 2-3 query positions on the input image (left), corresponding routed regions (middle) and final attention heat map (right).

D. Adapting Pretrained Plain ViT with BRA

Recently, to take advantage of large-scale pretraining with masked image modeling, a new research direction emerges to adapt plain ViT [15] for dense prediction tasks [4, 26]. Here we explore adapting pre-trained plain ViT [15] for semantic segmentation with our proposed BRA.

Specifically, we replace all or part of full multi-head self-attention (MHSA) modules in DeiT-B [40] with our BRA and directly load the weights pre-trained on ImageNet before training on ADE20K dataset for semantic segmentation. In this way, the linear projection weights of BRA modules are initialized with those of the original MHSA. We compare such an adaptation with those proposed in [26], *i.e*. using local window attention (window size $w = 14$) together with several global attention or convolution propagation blocks. We set window size $w = 4$ (which is equivalent to region partition size $S = 8$ since the feature map has a resolution of 32×32) and the number of regions to attend $k = 12$, hence each query attends to $4^2 \times 12 = 192$ key-value pairs, which is comparable to the local window attention where each query attends to $14 \times 14 = 196$ key-value pairs.

Table 8 shows the results. Without propagation blocks, the architecture using BRA significantly surpasses the one with local window attention by 2.4 mAP. When further equipped with 4 global propagation blocks, the performance of both architectures is improved, while the one using BRA still has an advantage of 0.2 mAP.

attention function	mIoU(%)
local window attention ($w = 14$)	43.55
BRA($w = 4, k = 12$)	**45.92**
local window attention + 4 conv prop. blks.	44.68
local window attention + 4 global prop. blks.	46.64
BRA + 4 global prop. blks.	**46.84**

Table 8. Adapting pretrained ViT [15] with BRA for semantic segmentation on ADE20K. For decoder, we use the Simple Feature Pyramid [26] followed by with Upernet [49] head.

E. More Visualization Results

To further show how BRA works, we demonstrate more visualization results in Figure 6.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229, 2020. 1, 3

[2] Chun-Fu Chen, Rameswar Panda, and Quanfu Fan. Regionvit: Regional-to-local attention for vision transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. 5

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

[4] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022. 11

[5] Zhiyang Chen, Yousong Zhu, Chaoyang Zhao, Guosheng Hu, Wei Zeng, Jinqiao Wang, and Ming Tang. Dpt: Deformable patch-based transformer for visual recognition. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2899–2907, 2021. 1, 3

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv:1904.10509*, 2019. 1, 3

[7] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34:9355–9366, 2021. 5, 7, 8

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

[9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition workshops*, pages 702–703, 2020. 6

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 764–773, 2017. 2

[11] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pages 2978–2988, 2019. 3

[12] 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, 2009. 6

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, June 2019. 1, 3

[14] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12124–12134, 2022. 1, 2, 3, 4, 5, 6, 7, 8

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations, ICLR 2021, 2021*, 2021. 1, 3, 11

[16] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017. 6

[17] Ankit Gupta, Guy Dar, Shaya Goodman, David Ciprut, and Jonathan Berant. Memory-efficient transformers via top-k attention. In Nafise Sadat Moosavi, Iryna Gurevych, Angela Fan, Thomas Wolf, Yufang Hou, Ana Marasovic, and Sujith Ravi, editors, *Proceedings of the Workshop on Simple and Efficient Natural Language Processing, 2021*, pages 39–52, 2021. 2

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 6

[19] 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. 5

[20] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv:1912.12180*, 2019. 7

[21] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661, 2016. 6

[22] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 603–612, 2019. 5, 7

[23] Zi-Hang Jiang, Qibin Hou, Li Yuan, Daquan Zhou, Yujun Shi, Xiaojie Jin, Anran Wang, and Jiashi Feng. All tokens matter: Token labeling for training better vision transformers. *Advances in Neural Information Processing Systems*, 34:18590–18602, 2021. 2, 5, 6, 8

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

[25] Kunchang Li, Yali Wang, Peng Gao, Guanglu Song, Yu Liu, Hongsheng Li, and Yu Qiao. Uniformer: Unified transformer for efficient spatiotemporal representation learning. *arXiv:2201.04676*, 2022. 5, 6, 7, 8

[26] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. *arXiv preprint arXiv:2203.16527*, 2022. 11

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 6

[28] 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. 6

[29] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9

[30] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 6

[31] Nvidia. How to access global memory efficiently in cuda c/c++ kernels. https://developer.nvidia.com/blog/how-access-global-memory-efficiently-cuda-c-kernels/. Accessed: 2022-10-25. 2

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems*, volume 32, 2019. 4

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 1

[34] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 5

[35] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *Proceedings of Advances in Neural Information Processing Systems*, volume 32, 2019. 7

[36] Mr D Murahari Reddy, Mr Sk Masthan Basha, Mr M Chinnaiahgari Hari, and Mr N Penchalaiah. Dall-e: Creating images from text. 2021. 1

[37] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10853–10862, 2022. 5, 7, 8

[38] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. In *The International Conference on Learning Representations, ICLR 2022, 2022*, 2022. 3, 5, 6, 7, 9

[39] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys (CSUR)*, 2020. 3

[40] 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, 11

[41] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, 2022. 1, 2, 3, 4, 5, 7

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 3, 4

[43] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv:2006.04768*, 2020. 3

[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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021. 1, 7

[45] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022. 5, 8

[46] 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. In *International Conference on Learning Representations, ICLR*, 2022. 1, 2, 3, 4, 5, 6, 7

[47] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 9

[48] Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4794–4803, 2022. 1, 2, 3, 4, 5, 7

[49] 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, 11

[50] Rui Yang, Hailong Ma, Jie Wu, Yansong Tang, Xuefeng Xiao, Min Zheng, and Xiu Li. Scalablevit: Rethinking the context-oriented generalization of vision transformer. *arXiv:2203.10790*, 2022. 5

[51] Ting Yao, Yingwei Pan, Yehao Li, Chong-Wah Ngo, and Tao Mei. Wave-vit: Unifying wavelet and transformers for visual representation learning. In *European Conference on Computer Vision*, pages 328–345. Springer, 2022. 5, 6, 7, 8

[52] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 6

[53] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not all tokens are equal: Human-centric visual analysis via token clustering transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11101–11111, 2022. 3

[54] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations, ICLR 2018*, 2018. 6

[55] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 6, 7