https://github.com/mahaoyuHKU/pytorch-boat

BOAT: Bilateral Local Attention Vision Transformer

Tan Yu[1], **Gangming Zhao**[2], **Ping Li**[1] and **Yizhou Yu**[2]
[1]Cognitive Computing Lab, Baidu Research
[2] Department of Computer Science, The University of Hong Kong

Abstract

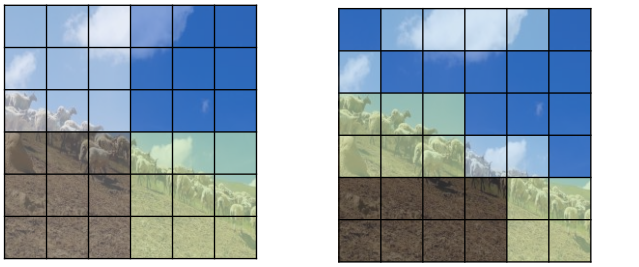
Vision Transformers achieved outstanding performance in many computer vision tasks. Early Vision Transformers such as ViT and DeiT adopt global self-attention, which is computationally expensive when the number of patches is large. To improve the efficiency, recent Vision Transformers adopt local self-attention mechanisms, where self-attention is computed within local windows. Despite the fact that window-based local self-attention significantly boosts efficiency, it fails to capture the relationships between distant but similar patches in the image plane. To overcome this limitation of image-space local attention, in this paper, we further exploit the locality of patches in the feature space. We group the patches into multiple clusters using their features, and self-attention is computed within every cluster. Such feature-space local attention effectively captures the connections between patches across different local windows but still relevant. We propose a Bilateral lOcal Attention vision Transformer (BOAT), which integrates feature-space local attention with image-space local attention. We further integrate BOAT with both Swin and CSWin models, and extensive experiments on several benchmark datasets demonstrate that our BOAT-CSWin model clearly and consistently outperforms existing state-of-the-art CNN models and vision Transformers.

1 Introduction

Following the great success of Transformers [Vaswani *et al.*, 2017] in natural language processing tasks, researchers have recently proposed vision Transformers [Dosovitskiy *et al.*, 2020; Touvron *et al.*, 2021; Liu *et al.*, 2021; Chu *et al.*, 2021], which have achieved outstanding performance in many computer vision tasks, including image recognition, detection, and segmentation. As early versions of vision transformers, ViT [Dosovitskiy *et al.*, 2020] and DeiT [Touvron *et al.*, 2021] uniformly divide an image into 16×16 patches (tokens) and apply a stack of standard Transformer layers to a sequence of tokens formed using these patches. The original self-attention mechanism is global, i.e., the receptive field of a patch in ViT and DeiT covers all patches of the image, which is vital for modeling long-range interactions among patches. On the other hand, the global nature of self-attention imposes a great challenge in efficiency. Specifically, the computational complexity of self-attention is quadratic in terms of the number of patches. As the number of patches is inversely proportional to the patch size when the size of the input image is fixed, the computational cost forces ViT and DeiT to adopt medium-size patches, which might not be as effective as smaller patches generating higher-resolution feature maps, especially for dense prediction tasks such as segmentation.

To maintain higher resolution feature maps while achieving high efficiency, some methods [Liu *et al.*, 2021; Chu *et al.*, 2021] exploit image-space local attention. They divide an image into multiple local windows, each of which includes a number of patches. Self-attention operations are only performed on patches within the same local window. This is a reasonable design since a patch is likely to be affiliated with other patches in the same local window but not highly relevant to patches in other windows. Thus, pruning attention between patches from different windows might not significantly deteriorate the performance. Meanwhile, the computational cost of window-based self-attention is much lower than that of the original self-attention over the entire image. Swin Transformer [Liu *et al.*, 2021] and Twins [Chu *et al.*, 2021] are such examples. Swin Transformer [Liu *et al.*, 2021] performs self-attention within local windows. To facilitate communication between patches from different windows, Swin Transformer has two complementary window partitioning schemes, and a window in one scheme overlaps with multiple windows in the second scheme. Twins [Chu *et al.*, 2021] performs self-attention within local windows and builds connections among different windows by performing (global) self-attention over feature vectors sparsely sampled from the entire image using a regular subsampling pattern.

In this work, we rethink local attention and explore locality from a broader perspective. Specifically, we investigate feature-space local attention apart from its image-space counterpart. Instead of computing local self-attention in the image space, feature-space local attention exploits locality in the feature space. It is based on the fact that patch feature vectors close to each other in the feature space tend to have more influence on each other in the computed self-attention results. This is because the actual contribution of a feature vector to



(a) Image-space Local Attention (b) Feature-space Local Attention

Figure 1: Comparisons between the image-space local attention and the feature-space local attention.

the self-attention result at another feature vector is controlled by the similarity between these two feature vectors. Feature-space local attention computes the self-attention result at a feature vector using its feature-space nearest neighbors only while setting the contribution from feature vectors farther away to zero. This essentially defines a piecewise similarity function, which clamps the similarity between feature vectors far apart to zero. In comparison to the aforementioned image-space local attention, feature-space local attention has been rarely exploited in vision transformers. As shown in Figure 1, feature-space local attention computes attention among relevant patches which might not be close to each other in the image plane. Thus, it is a natural compensation to image-space local attention, which might miss meaningful connections between patches residing in different local windows.

In this paper, we propose a novel vision Transformer architecture, Bilateral lOcal Attention vision Transformer (BOAT), to exploit the complementarity between feature-space and image-space local attention. The essential component in our network architecture is the bilateral local attention block, consisting of a feature-space local attention module and an image-space local attention module. The image-space local attention module divides an image into multiple local windows as Swin [Liu *et al.*, 2021] and CSWin [Dong *et al.*, 2021], and self-attention is computed within each local window. In contrast, feature-space local attention groups all the patches into multiple clusters and self-attention is computed within each cluster. Feature-space local attention could be implemented in a straightforward way using K-means clustering. Nevertheless, K-means clustering cannot ensure the generated clusters are evenly sized, making it challenging to have efficient parallel implementation. In addition, unevenly sized clusters may also negatively impact the overall effectiveness of self-attention. To overcome this obstacle, we propose a hierarchical balanced clustering approach, which groups patches into clusters of equal size.

We conduct experiments on multiple computer vision tasks, including image classification, semantic segmentation, and object detection. Systematic experiments on several public benchmarks demonstrate that the proposed BOAT clearly and consistently improves existing image-space local attention vision Transformers, including Swin [Liu *et al.*, 2021] and CSWin [Dong *et al.*, 2021], on these tasks.

2 Related Work

2.1 Vision Transformers

In the past decade, CNN, as the *de facto* vision backbone, has achieved tremendous successes in numerous computer vision tasks [Krizhevsky *et al.*, 2012; He *et al.*, 2016]. The natural language processing (NLP) backbone, Transformer, has recently attracted the attention of researchers in the computer vision community. After dividing an image into non-overlapping patches (tokens), Vision Transformer (ViT) [Dosovitskiy *et al.*, 2020] applies Transformer to the tokens for communications among the tokens. Without delicately devised convolution kernels, ViT achieved excellent performance in image recognition [Dosovitskiy *et al.*, 2020], saliency prediction [Zhang *et al.*, 2021] and 3D object recognition [Chen *et al.*, 2021] in comparison to its CNN counterparts. Nevertheless, ViT requires a huge training corpus for a good performance. DeiT [Touvron *et al.*, 2021] improves data efficiency by exploring more advanced training and data augmentation strategies. Recently, many efforts have been devoted to further improving the recognition accuracy and efficiency of Transformer-based vision backbones.

To boost the recognition accuracy, T2T-ViT [Yuan *et al.*, 2021] proposes a Tokens-to-Token transformation, recursively aggregating neighboring tokens into one token for modeling local structures. TNT [Han *et al.*, 2021] also investigates local structure modeling. It additionally builds an inner-level Transformer to model the visual content within each local patch. PVT [Wang *et al.*, 2021a] uses small-scale patches, yielding higher resolution feature maps for dense prediction. Meanwhile, PVT progressively shrinks the feature map size to reduce the computational cost. PiT [Heo *et al.*, 2021] also decreases spatial dimensions through pooling and increases channel dimensions in deeper layers, achieving improved recognition performance in comparison to ViT.

More recently, computing self-attention within local windows [Liu *et al.*, 2021; Chu *et al.*, 2021; Huang *et al.*, 2021; Dong *et al.*, 2021], has achieved a good trade-off between effectiveness and efficiency. For example, Swin [Liu *et al.*, 2021] divides an image into multiple local windows and computes self-attention among patches from the same window. To achieve communication across local windows, Swin shifts window configurations in different layers. Twins [Chu *et al.*, 2021] also exploits local windows for enhancing efficiency. To achieve cross-window communication, it computes additional self-attention over features sampled from the entire image. Similarly, Shuffle Transformer [Huang *et al.*, 2021] exploits local windows and performs cross-window communication by shuffling patches. CSWin [Dong *et al.*, 2021] adopts cross-shaped windows, computing self-attention in horizontal and vertical stripes in parallel. The aforementioned local attention models [Liu *et al.*, 2021; Chu *et al.*, 2021; Huang *et al.*, 2021; Dong *et al.*, 2021] only exploit image-space locality. In contrast, our BOAT exploits not only image-space locality but also feature-space locality.

2.2 Efficient Transformers

High computational costs limit Transformer's usefulness in practice. Thus, much research [Tay *et al.*, 2020b] has re-

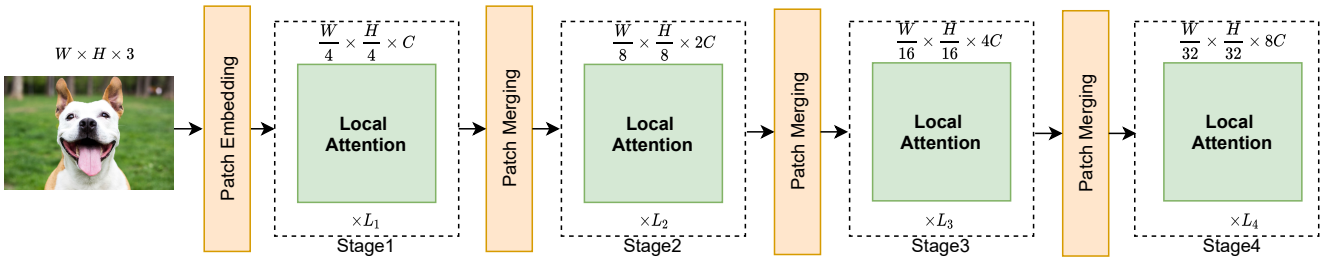


Figure 2: Architecture of the proposed Bilateral lOcal Attention Vision Transformer (BOAT).

cently been dedicated to improving efficiency. One popularly used strategy for speeding up Transformers enforces sparse attention matrices by limiting the receptive field of each token. Image Transformer [Parmar *et al.*, 2018] and Block-wise Transformer [Qiu *et al.*, 2020] divide a long sequence into local buckets. In this case, the attention matrix has a block-diagonal structure. Only self-attention within each bucket is retained, and cross-bucket attention is pruned. Transformers based on image-space local attention, such as Swin Transformer [Liu *et al.*, 2021], Twins [Chu *et al.*, 2021], Shuffle Transformer [Huang *et al.*, 2021], and CSWin Transformer [Dong *et al.*, 2021] also adopt buckets (windows) for boosting efficiency. In parallel to bucket-based local attention, strided attention is another approach for achieving sparse attention matrices. Sparse Transformer [Child *et al.*, 2019] and LongFormer [Beltagy *et al.*, 2020] utilize strided attention, which computes self-attention over features sampled with a sparse grid with a stride larger than one, leading to a sparse attention matrix facilitating faster computation. The global sub-sampling layer in Twins [Chu *et al.*, 2021] and the shuffle module in Shuffle Transformer [Huang *et al.*, 2021] can be regarded as strided attention modules.

Unlike the above mentioned image-space local attention, several methods determine the scope of local attention in the feature space. Reformer [Kitaev *et al.*, 2020] distributes tokens to buckets by feature-space hashing functions. Routing Transformer [Roy *et al.*, 2021] applies online K-means to cluster tokens. Sinkhorn Sorting Network [Tay *et al.*, 2020a] learns to sort and divide an input sequence into chunks. Our feature-space local attention module also falls into this category. As far as we know, this paper is the first attempt to apply feature-space grouping to vision Transformers.

3 Method

As visualized in Figure 2, the proposed BOAT architecture consists of a patch embedding module, and a stack of L Bilateral Local Attention blocks. Meanwhile, we exploit a hierarchical pyramid structure. Below we only briefly introduce the patch embedding module and the hierarchical pyramid structure and leave the details of the proposed Bilateral Local Attention block in Section 3.1 and 3.2.

Patch embedding. For an input image with size $H \times W$, we follow Swin [Liu *et al.*, 2021] and CSWin Transformer [Dong *et al.*, 2021], and leverage convolutional token embedding (7×7 convolution layer with stride 4) to obtain $\frac{H}{4} \times \frac{W}{4}$ patch tokens, and the dimension of each token is C.

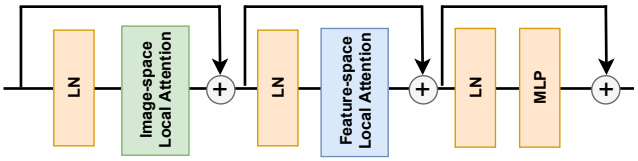


Figure 3: Architecture of Bilateral Local Attention (BLA) Block.

Hierarchical pyramid structure. Similar to Swin [Liu *et al.*, 2021] and CSWin Transformer [Dong *et al.*, 2021], we also build a hierarchical pyramid structure. The whole architecture consists of four stages. A convolution layer (3×3, stride 2) is used between two adjacent stages to merge patches. It reduces the number of tokens and doubles the number of channels. Therefore, in the i-th stage, the feature map contains $\frac{H}{2^{(i+1)}} \times \frac{W}{2^{(i+1)}}$ tokens and $2^{i-1}C$ channels.

3.1 Bilateral Local Attention Block

As shown in Figure 3, a Bilateral Local Attention (BLA) Block consists of an image-space local attention (ISLA) module, a feature-space (content-based) local attention (FSLA) module, an MLP module, and several layer normalization (LN) modules. Let us denote the set of input tokens by $\mathcal{T}_{\text{in}} = \{\mathbf{t}_i\}_{i=1}^{N}$ where $\mathbf{t}_i \in \mathbb{R}^C$, C is the number of channels and N is the number of tokens. The input tokens go through a normalization layer followed by an image-space local attention (ISLA) module, which has a shortcut connection:

$$\mathcal{T}_{\text{ISLA}} = \mathcal{T}_{\text{in}} + \text{ISLA}(\text{LN}(\mathcal{T}_{\text{in}})). \quad (1)$$

Image-space local attention only computes self-attention among tokens within the same local window. We adopt existing window-based local attention modules, such as those in Swin Transformer [Liu *et al.*, 2021] and CSWin Transformer [Dong *et al.*, 2021] as our ISLA module due to their excellent performance. Intuitively, patches within the same local window are likely to be closely related to each other. However, some distant patches in the image space might also reveal important connections, such as similar contents, that could be helpful for visual understanding. Simply throwing away such connections between distant patches in the image space might deteriorate image recognition performance.

To bring back the useful information dropped out by image-space local attention, we develop a feature-space local attention (FSLA) module. The output of the ISLA module, $\mathcal{T}_{\text{ISLA}}$, is fed into another normalization layer followed by a

feature-space (content-based) local attention (FSLA) module, which also has a shortcut connection:

$$\mathcal{T}_{\text{FSLA}} = \mathcal{T}_{\text{ISLA}} + \text{FSLA}(\text{LN}(\mathcal{T}_{\text{ISLA}})). \quad (2)$$

The FSLA module computes self-attention among tokens that are close in the feature space, which is complementary to the ISLA module. Meanwhile, by only considering local attention in the feature space, FSLA is more efficient than the original (global) self-attention. We will present the details of FSLA in Section 3.2. Following CSWin [Dong *et al.*, 2021], we also add locally-enhanced positional encoding to each feature-space local attention layer to model position.

At last, the output of the FSLA module, $\mathcal{T}_{\text{FSLA}}$, is processed by another normalization layer and an MLP module to generate the output of a Bilateral Local Attention Block:

$$\mathcal{T}_{\text{out}} = \mathcal{T}_{\text{FSLA}} + \text{MLP}(\text{LN}(\mathcal{T}_{\text{FSLA}})). \quad (3)$$

Following existing vision Transformers [Dosovitskiy *et al.*, 2020; Touvron *et al.*, 2021], the MLP module consists of two fully-connected layers. The first one increases the feature dimension from C to rC and the second one decreases the feature dimension from rC back to C. By default, we set $r = 4$.

3.2 Feature-Space Local Attention

Different from image-space local attention which groups tokens according to their spatial locations in the image plane, feature-space local attention seeks to group tokens according to their content, *i.e.*, features. We could simply perform K-means clustering on token features to achieve this goal. Nevertheless, K-means clustering cannot ensure that the generated clusters are equally sized, which makes it difficult to have efficient parallel implementation on GPU platforms, and may also negatively impact the overall effectiveness of self-attention, which will be demonstrated in our ablation studies.

Balanced hierarchical clustering. To overcome the imbalance problem of K-means clustering, we propose a balanced hierarchical clustering, which performs K levels of clustering. At each level, it conducts a balanced binary clustering, which equally splits a set of tokens into two clusters. Let us denote the set of input tokens by $\mathcal{T} = \{\mathbf{t}_i\}_{i=1}^{N}$. In the first level, it splits N tokens in $\mathcal{T}$ into two subsets with $N/2$ tokens each. At the k-th level, it splits $N/2^{k-1}$ tokens assigned to the same subset in the upper level into two smaller subsets of $N/2^k$ size. At the end, we obtain 2^K evenly sized subsets in the final level, $\{\mathcal{T}_i\}_{i=1}^{2^K}$, and the size of each subset $|\mathcal{T}_i|$ is equal to $N/2^K$. Here, we require the condition that N is divisible by 2^K, which can be easily satisfied in existing vision Transformers. We visualize the process of balanced hierarchical clustering in Figure 4. The core operation in balanced hierarchical clustering is our devised balanced binary clustering, which we elaborate below.

Balanced binary clustering. Given a set of $2m$ tokens $\{\mathbf{t}_i\}_{i=1}^{2m}$, balanced binary clustering divides them into two groups and the size of each group is m. Similar to K-means clustering, our balanced binary clustering relies on cluster centroids. To determine the cluster assignment of each sample, K-means clustering only considers the distance between the sample and all centroids. In contrast, our balanced binary clustering further requires that the two resulting clusters have equal size. Let us denote the two cluster centroids as $\mathbf{c}_1$ and $\mathbf{c}_2$. For each token $\mathbf{t}_i$, we compute distance ratio, r_i, as a metric to determine its cluster assignment:

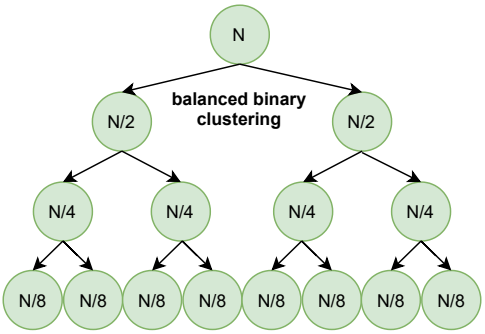


Figure 4: Example of balanced hierarchical clustering. In this example, the number of hierarchical levels is 3. There are $2^3 = 8$ clusters in the bottom level and the size of each cluster is $\frac{N}{8}$.

$$r_i = \frac{s(\mathbf{t}_i, \mathbf{c}_1)}{s(\mathbf{t}_i, \mathbf{c}_2)}, \ \forall i \in [1, 2m], \quad (4)$$

where $s(\mathbf{x}, \mathbf{y})$ denotes the cosine similarity between $\mathbf{x}$ and $\mathbf{y}$. The $2m$ tokens $\{\mathbf{t}_i\}_{i=1}^{2m}$ are sorted in the decreasing order of their distance ratios $\{r_i\}_{i=1}^{2m}$. We assign the tokens in the first half of the sorted list to the first cluster $\mathcal{C}_1$ and those in the second half of the sorted list to the second cluster $\mathcal{C}_2$, where the size of both $\mathcal{C}_1$ and $\mathcal{C}_2$ is m. The mean of the tokens from each cluster is used to update the cluster centroid. Similar to K-means, our balanced binary clustering updates cluster centroids and the cluster membership of every sample in an iterative manner. The detailed steps of the proposed balanced binary clustering are given in Algorithm 1.

Algorithm 1: Balanced Binary Clustering.

Input: Tokens $\{\mathbf{t}_i\}_{i=1}^{2m}$ and the iteration number, T.
Output: Two clusters, $\mathcal{C}_1$ and $\mathcal{C}_2$.

Initialize centroids $\mathbf{c}_1 = \frac{\sum_{i=1}^{m} \mathbf{t}_i}{m}$, $\mathbf{c}_2 = \frac{\sum_{i=m+1}^{2m} \mathbf{t}_i}{m}$
while $n\_iter \in [1, T]$ **do**
  for $i \in [1, 2m]$ **do**
    $r_i = \frac{s(\mathbf{t}_i, \mathbf{c}_1)}{s(\mathbf{t}_i, \mathbf{c}_2)}$
  $[i_1, \cdots, i_{2m}] = \text{argsort}([r_1, \cdots, r_{2m}])$
  $\mathcal{C}_1 = \{\mathbf{t}_{i_j}\}_{j=1}^{m}$, $\mathcal{C}_2 = \{\mathbf{t}_{i_j}\}_{j=m+1}^{2m}$
  $\mathbf{c}_1 = \frac{\sum_{\mathbf{c} \in \mathcal{C}_1} \mathbf{c}}{m}$, $\mathbf{c}_2 = \frac{\sum_{\mathbf{c} \in \mathcal{C}_2} \mathbf{c}}{m}$

In the aforementioned balanced binary clustering, two resulting clusters have no shared tokens, *i.e.*, $\mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset$. One main drawback of the non-overlapping setting is that, a token in the middle portion of the sorted list has some of its feature-space neighbors in one cluster while the other neighbors in the other cluster. No matter which cluster this token is finally assigned to, the connection between the token and part of its feature-space neighbors will be cut off. For example, the token at the m-th location of the sorted list cannot communicate

with the token at the $m+1$-st location during attention calculation because they are assigned to different clusters. Overlapping balanced binary clustering overcomes this drawback by assigning the first $m+n$ tokens in the sorted list to the first cluster, *i.e.*, $\hat{\mathcal{C}}_1 = \{\mathbf{t}_{j_i}\}_{i=1}^{m+n}$, and the last $m+n$ tokens in the sorted list to the second cluster, *i.e.*, $\hat{\mathcal{C}}_2 = \{\mathbf{t}_{j_i}\}_{i=m-n+1}^{2m}$. Thus, the two resulting clusters have $2n$ tokens in common, *i.e.*, $\hat{\mathcal{C}}_1 \cap \hat{\mathcal{C}}_2 = \{\mathbf{t}_{j_i}\}_{i=m-n+1}^{m+n}$. By default, we only adopt overlapping binary clustering at the last level of the proposed balanced hierarchical clustering and use the non-overlapping version at the other levels. We set $n = 20$ in all experiments for overlapping binary clustering.

Local attention within cluster. Through the above introduced balanced hierarchical clustering, the set of tokens, $\mathcal{T}$, are grouped into 2^K subsets $\{\mathcal{T}_i\}_{i=1}^{2^K}$, where $|\mathcal{T}_i| = \frac{N}{2^K}$. The standard self-attention (SA) is performed within each subset:

$$\hat{\mathcal{T}}_k = \mathrm{SA}(\mathcal{T}_k),\ \forall k \in [1, 2^K]. \tag{5}$$

The output, $\hat{\mathcal{T}}$, is the union of all attended subsets:

$$\hat{\mathcal{T}} = \bigcup_{k\in[1,K]} \hat{\mathcal{T}}_k. \tag{6}$$

Following the multi-head configuration in Transformer, we also devise multi-head feature-space local attention. Note that, in our multi-head feature-space local attention, we implement multiple heads not only for computing self-attention in Eq. (5) as a standard Transformer, but also for performing balanced hierarchical clustering. That is, balanced hierarchical clustering is performed independently in each head. Thus, for a specific token, in different heads, it might pay feature-based local attention to different tokens. This configuration is more flexible than Swin [Liu *et al.*, 2021], where multiple heads share the same local window.

4 Experiments

To demonstrate the effectiveness of our BOAT as a general vision backbone, we conduct experiments on image classification, semantic segmentation and object detection. It is developed in PaddlePaddle platform developed by Baidu. We build BOAT upon two recent local attention Vision Transformers, Swin [Liu *et al.*, 2021] and CSWin [Dong *et al.*, 2021]. We term the BOAT built upon Swin as BOAT-Swin. In BOAT-Swin, the image-space local attention (ISLA) module adopts shifted window attention in Swin. In contrast, the ISLA module in BOAT-CSWin uses cross-shape window attention in CSWin. We provide specifications of BOAT-Swin and BOAT-CSWin in Section B of the Appendix. We present main experimental results in the following sections. More ablation studies are presented in Section A of the Appendix.

4.1 Image Classification

We follow the same training strategies as other vision Transformer counterparts. We train our models using the training split of ImageNet-1K [Deng *et al.*, 2009] with 224×224 input resolution and without external data. Specifically, we train BOAT-Swin for 300 epochs and train BOAT-CSWin for 310 epochs. We use the AdamW optimizer with the cosine learning rate scheduler and a linear warm-up process.

Method	size	#para.	FLOPs	Top-1
ReGNetY-4G	224	21M	4.0G	80.0
EffNet-B4	380	19M	4.2G	82.9
EffNet-B5	456	30M	9.9G	83.6
DeiT-S	224	22M	4.6G	79.8
PVT-S	224	25M	3.8G	79.8
PiT-S	224	24M	2.9G	80.9
T2T-14	224	22M	5.2G	81.5
TNT-S	224	24M	5.2G	81.3
Swin-T	224	29M	4.5G	81.3
BOAT-Swin-T (ours)	224	31M	5.2G	82.3
PVTv2-B2	224	25M	4.0G	82.0
Shuffle-T	224	29M	4.6G	82.5
NesT-T	224	17M	5.8G	81.5
Focal-T	224	29M	4.9G	82.2
CrossFormer-S	224	31M	4.9G	82.5
CSWin-T	224	23M	4.3G	82.7
BOAT-CSWin-T (ours)	224	27M	5.1G	**83.7**
ReGNetY-8G	224	39M	8.0	81.7
EffNet-B6	528	43M	19.0G	84.0
PVT-M	224	44M	6.7G	81.2
PVT-L	224	61M	9.8G	81.7
T2T-19	224	39M	8.9G	81.9
MViT-B	224	37M	7.8G	83.0
Swin-S	224	50M	8.7G	83.0
BOAT-Swin-S (ours)	224	56M	10.1G	83.6
PVTv2-B4	224	62M	10.1G	83.6
Twins-B	224	56M	8.3G	83.2
Shuffle-S	224	50M	8.9G	83.5
NesT-S	224	38M	10.4G	83.3
Focal-S	224	51M	9.1G	83.5
CrossFormer-B	224	52M	9.2G	83.4
CSWin-S	224	35M	6.9G	83.6
BOAT-CSWin-S (ours)	224	41M	8.0G	**84.1**
ReGNetY-16G	224	84M	16.0G	82.9
ViT-B/16T	384	86M	55.4G	77.9
DeiT-B	384	86M	55.4G	83.1
DeiT-B	224	86M	17.5G	81.8
T2T-24	224	64M	14.1G	82.3
TNT-B	224	66M	14.1G	82.8
PiT-B	224	74M	12.5G	82.0
Swin-B	224	88M	15.4G	83.5
BOAT-Swin-B (ours)	224	98M	17.8G	83.8
PVTv2-B5	224	82M	11.8G	83.8
Twins-L	224	99M	14.8G	83.7
Shuffle-B	224	88M	15.4G	84.0
NesT-B	224	68M	17.9G	83.8
Focal-B	224	90M	16.0G	83.8
CrossFormer-L	224	92M	16.1G	84.0
CSWin-B	224	78M	15.0G	84.2
BOAT-CSWin-B (ours)	224	90M	17.5G	**84.7**

Table 1: Our BOAT is compared with state-of-the-art CNNs and other vision Transformers on ImageNet-1K image classification.

Table 1 compares the performance of our BOAT models with CNN models including ReGNet [Radosavovic *et al.*, 2020] and EfficientNet [Tan and Le, 2019] as well as other vision Transformer models including ViT [Dosovitskiy *et*

al., 2020], DeiT [Touvron *et al.*, 2021], PVT [Wang *et al.*, 2021a], T2T [Yuan *et al.*, 2021], TNT [Han *et al.*, 2021], PiT [Heo *et al.*, 2021], Swin [Liu *et al.*, 2021], MViT [Fan *et al.*, 2021], PVTv2 [Wang *et al.*, 2021b], Twins [Chu *et al.*, 2021], Shuffle Transformer [Huang *et al.*, 2021], Nest Transformer [Zhang *et al.*, 2022], Focal Transformer [Yang *et al.*, 2021], Cross-Former [Wang *et al.*, 2021c] and CSWin [Dong *et al.*, 2021]. As shown in the table, with a slight increase in the number of parameters and FLOPs, our BOAT-Swin model consistently improves the vanilla Swin model under the tiny, small and base settings. Specifically, our model improves the top-1 accuracy of Swin-Tiny from 81.3 to 82.3, that of Swin-Small from 83.0 to 83.6 and that of Swin-Base from 83.5 to 83.8. Meanwhile, our BOAT-CSWin model also improves the CSWin model by a similar degree under the tiny, small and base settings. Such improvements over Swin and CSWin demonstrate the effectiveness of feature-space local attention. Note that our BOAT-CSWin model outperforms existing CNN models and vision Transformers.

4.2 Semantic Segmentation

We further investigate the effectiveness of our BOAT for semantic segmentation on the ADE20K dataset [Zhou *et al.*, 2017]. Here, we employ UperNet [Xiao *et al.*, 2018] as the basic framework. We load the BOAT model pre-trained on ImageNet-1K for fine-tuning on the downstream semantic segmentation task. For a fair comparison, we follow previous work and train UperNet 160K iterations with batch size 16 using 8 GPUs. In Table 2, we compare the semantic segmentation performance of our BOAT with other vision Transformer models including Swin [Liu *et al.*, 2021], Twins [Chu *et al.*, 2021], Shuffle Transformer [Huang *et al.*, 2021], Focal Transformer [Yang *et al.*, 2021], and CSWin [Dong *et al.*, 2021]. As shown in the table, with a slight increase in the number of parameters and FLOPs, our BOAT-Swin model consistently improves the semantic segmentation performance of the Swin model under the tiny, small and base settings. Specifically, Swin-Tiny achieves a 44.5% mIoU whereas our BOAT-Swin-Tiny achieves a 46.0% mIoU. In the meantime, our BOAT-CSWin model also constantly obtains higher segmentation mIoUs than the CSWin model under the tiny, small and base settings. In comparison to other vision Transformers in Table 2, our BOAT-CSWin model achieves state-of-the-art performance in semantic segmentation.

4.3 Object Detection

We also evaluate the proposed BOAT on object detection. Experiments are conducted on the MS-COCO dataset using the Mask R-CNN [He *et al.*, 2017] framework. We load the BOAT model pre-trained on ImageNet-1K for fine-tuning on the downstream object detection task. Since CSWin has not released codes for object detection, we only implement BOAT-Swin for this task. We adopt the 3× learning rate schedule, which is the same as Swin. We compare the performance of our BOAT-Swin and the original Swin in Table 3. The evaluation is on the MSCOCO val2017 split. Since Swin only reports the performance of Swin-Tiny and Swin-Small models when using the Mask R-CNN framework, we also report the performance of our BOAT-Swin-Tiny and BOAT-Swin-Small only. As shown in Table 3, with a slight increase in the number of parameters and FLOPs, our BOAT-Swin outperforms Swin under both tiny and small settings.

Method	#para.(M)	FLOPs(G)	mIoU(%)
TwinsP-S [Chu *et al.*, 2021]	55	919	46.2
Twins-S [Chu *et al.*, 2021]	54	901	46.2
Swin-T [Liu *et al.*, 2021]	60	945	44.5
BOAT-Swin-T (ours)	62	986	46.0
Shuffle-T [Huang *et al.*, 2021]	60	949	46.6
Focal-T [Yang *et al.*, 2021]	62	998	45.8
CSWin-T [Dong *et al.*, 2021]	60	959	49.3
BOAT-CSWin-T (ours)	64	1012	**50.5**
TwinsP-B [Chu *et al.*, 2021]	74	977	47.1
Twins-B [Chu *et al.*, 2021]	89	1020	47.7
Swin-S [Liu *et al.*, 2021]	81	1038	47.6
BOAT-Swin-S (ours)	87	1113	48.4
Shuffle-S [Huang *et al.*, 2021]	81	1044	48.4
Focal-S [Yang *et al.*, 2021]	85	1130	48.0
CSWin-S [Dong *et al.*, 2021]	65	1027	50.0
BOAT-CSWin-S (ours)	70	1101	**50.6**
TwinsP-L [Chu *et al.*, 2021]	92	1041	48.6
Twins-L [Chu *et al.*, 2021]	133	1164	48.8
Swin-B [Liu *et al.*, 2021]	121	1188	48.1
BOAT-Swin-B (ours)	131	1299	48.7
Shuffle-B [Huang *et al.*, 2021]	121	1196	49.0
Focal-B [Yang *et al.*, 2021]	126	1354	49.0
CSWin-B [Dong *et al.*, 2021]	109	1222	50.8
BOAT-CSWin-B (ours)	121	1349	**50.9**

Table 2: Comparison of semantic segmentation performance on ADE20K dataset with UperNet framework. FLOPs are calculated with the 512 × 2048 resolution. We report mIoU under the single-scale setting, and the size of a testing image is 512 × 512.

Method	#para.(M)	FLOPs(G)	mAP^{Box}	mAP^{Mask}
Swin-T	48	267	46.0	41.6
BOAT-Swin-T (ours)	50	306	**47.5**	**42.8**
Swin-S	69	359	48.5	43.3
BOAT-Swin-S (ours)	75	431	**49.0**	**43.8**

Table 3: Comparison of object detection performance. Experiments are conducted on the MS-COCO dataset using the Mask R-CNN framework. FLOPs are measured at 800 × 1280 resolution.

5 Conclusion

In this paper, we have presented a new Vision Transformer architecture named Bilateral lOcal Attention Transformer (BOAT), which performs multi-head local self-attention in both feature and image spaces. To compute feature-space local attention, we propose a hierarchical balanced clustering approach to group patches into multiple evenly sized clusters, and self-attention is computed within each cluster. We have applied BOAT to multiple computer vision tasks including image classification, semantic segmentation and object detection. Our systematic experiments on several benchmark datasets have demonstrated that BOAT can clearly and consistently improve the performance of existing image-space local attention vision Transformers, including Swin [Liu *et al.*, 2021] and CSWin [Dong *et al.*, 2021], on these tasks.

References

[Beltagy *et al.*, 2020] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[Chen *et al.*, 2021] Shuo Chen, Tan Yu, and Ping Li. Mvt: Multi-view vision transformer for 3d object recognition. *BMVC*, 2021.

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

[Chu *et al.*, 2021] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, H. Ren, X. Wei, H. Xia, and C. Shen. Twins: Revisiting the design of spatial attention in vision transformers. In *NeurIPS*, 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 *CVPR*, 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, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020.

[Fan *et al.*, 2021] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, J. Malik, and C. Feichtenhofer. Multiscale vision transformers. In *ICCV*, 2021.

[Han *et al.*, 2021] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *NeurIPS*, 2021.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[Heo *et al.*, 2021] Byeongho Heo, Sangdoo Yun, Dongyoon Han, S. Chun, J. Choe, and S. J. Oh. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021.

[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.

[Kitaev *et al.*, 2020] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *ICLR*, 2020.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

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

[Parmar *et al.*, 2018] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

[Qiu *et al.*, 2020] J. Qiu, H. Ma, O. Levy, S. Yih, S. Wang, and J. Tang. Blockwise self-attention for long document understanding. *EMNLP:Findings*, 2020.

[Radosavovic *et al.*, 2020] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *CVPR*, 2020.

[Roy *et al.*, 2021] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *TACL*, 2021.

[Tan and Le, 2019] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[Tay *et al.*, 2020a] Yi Tay, Dara Bahri, Liu Yang, D. Metzler, and D. Juan. Sparse sinkhorn attention. In *ICML*, 2020.

[Tay *et al.*, 2020b] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[Touvron *et al.*, 2021] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, J. Uszkoreit, L. Jones, A. N Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, 2017.

[Wang *et al.*, 2021a] Wenhai Wang, Enze Xie, Xiang Li, D. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.

[Wang *et al.*, 2021b] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, T. Lu, P. Luo, and L. Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv preprint arXiv:2106.13797*, 2021.

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

[Xiao *et al.*, 2018] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[Yang *et al.*, 2021] Jianwei Yang, C. Li, P. Zhang, X. Dai, B. Xiao, L. Yuan, and J. Gao. Focal attention for long-range interactions in vision transformers. *NeurIPS*, 2021.

[Yuan *et al.*, 2021] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.

[Zhang *et al.*, 2021] Jing Zhang, Jianwen Xie, Nick Barnes, and Ping Li. Learning generative vision transformer with energy-based latent space for saliency prediction. *NeurIPS*, 2021.

[Zhang *et al.*, 2022] Zizhao Zhang, Han Zhang, Long Zhao, Ting Chen, Sercan Arik, and Tomas Pfister. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *AAAI*, 2022.

[Zhou *et al.*, 2017] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017.

A Ablation Studies

Comparisons with Reformer and K-means clustering. Reformer [Kitaev *et al.*, 2020] also exploits feature-based local attention. It divides tokens into multiple groups using Locality Sensitivity Hashing (LSH). LSH is based on random projection, which is independent of specific input data, and might be sub-optimal for different input data. K-means clustering is another choice for dividing tokens into multiple groups for exploiting feature-based local attention. Nevertheless, K-means clustering cannot ensure that the generated clusters are equally sized, which makes it difficult to have efficient parallel implementation on GPU platforms. To enforce the clusters from K-means clustering to be equally sized, we can sort the tokens according to their cluster index and then equally divide the sorted tokens into multiple groups, as visualized in Figure 5. However, this sort-and-divide process might divide tokens from a large cluster into multiple groups and also merge tokens from small clusters into the same group. This would negatively impact the overall effectiveness of feature-based local attention.

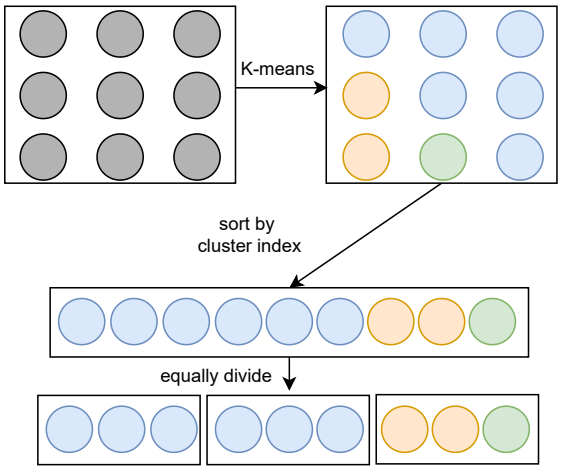


Figure 5: Enforcing the clusters from K-means to be equally sized.

We compare the image recognition performance of BOAT-Swin-Tiny with that of a model where our balanced hierarchical clustering is replaced with LSH in Reformer or K-means clustering. We keep the other layers unchanged. As shown in Table 4, our BOAT-Swin-Tiny clearly outperforms Reformer and K-means clustering.

Method	Reformer	K-means	Ours
Top-1 Accuracy	81.7	81.8	82.3

Table 4: Comparisons with Reformer and K-means.

Overlapping Balanced Hierarchical Clustering. In Table 5, we compare the performance of overlapping balanced hierarchical clustering with its non-overlapping counterpart on the ImageNet-1K dataset. As shown in the table, the overlapping setting achieves consistently higher classification accuracy in BOAT-CSwin-Tiny, Small and Base models. Higher accuracy is expected since the overlapping setting gives rise to larger receptive fields.

Overlap	BOAT-CSWin-T	BOAT-CSWin-S	BOAT-CSWin-B
No	83.3%	84.0%	84.5%
Yes	83.7%	84.1%	84.7%

Table 5: Comparison of classification accuracy between overlapping balanced hierarchical clustering and the non-overlapping version.

The influence of the number of iterations. As shown in Algorithm 1 of the main text, our balanced binary clustering is an iterative algorithm. We evaluate the influence of the number of iterations on the performance. As shown in Table 6, using a single iteration, the BOAT-Swin-Tiny model achieves 82.1% top-1 accuracy. When the number of iterations is increased to 2 and 3, the top-1 accuracy reaches 82.3. By default, we use two iterations in all experiments.

# of iter.	1	2	3
Top-1 Accuracy	82.1	82.3	82.3

Table 6: The influence of the number of iterations in balanced binary clustering on the classification accuracy of the BOAT-Swin-Tiny.

B Network Architecture Specifications

B.1 BOAT-Swin

BOAT-Swin is built by replacing a subset of the blocks in Swin with the proposed Bilateral Local Attention (BLA) Block. Similar to Swin, our BOAT-Swin consists of 4 stages. In each of the first three stages, we replace half of the Swin blocks with our BLA blocks. In all different versions, our BOAT-Swin has the same number of blocks as the corresponding version of Swin. To be specific, our BOAT-Swin-Tiny has $[2, 2, 6, 2]$ blocks in four stages, our BOAT-Swin-Small has $[2, 2, 18, 2]$ blocks in four stages, and our BOAT-Swin-Base has $[2, 2, 18, 2]$ blocks in four stages. In stage i, we set the number of clusters in our feature-space local attention (FSLA) module of the BLA block to 2^{8-2i}. That is, the number of levels in our balanced hierarchical clustering is $K = 8 - 2i$. The detailed specifications of our BOAT-Swin are given in Table 7.

B.2 BOAT-CSWin

BOAT-CSWin also has 4 stages. In all different versions, our BOAT-CSWin has the same number of blocks as the corresponding version of CSWin. Except for the 4-th stage of BOAT-CSwin-Tiny, BOAT-CSWin replaces half of the blocks in each stage of CSwin with BLA blocks. Our BOAT-CSWin-Tiny has $[1, 2, 21, 1]$ blocks in four stages, our BOAT-CSWin-Small has $[2, 4, 32, 2]$ blocks in four stages, and our BOAT-CSWin-Base has $[2, 4, 32, 2]$ blocks in four stages. In stage i, we set the number of clusters in our feature-space local attention (FSLA) module of the BLA block to 2^{8-2i}. That is, the number of levels in our balanced hierarchical clustering is $K = 8 - 2i$. The detailed specifications of our BOAT-CSWin are given in Table 8.

	BOAT-Swin-T	BOAT-Swin-S	BOAT-Swin-B
	concat 4×4	concat 4×4	concat 4×4
stage 1	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 96, head 3} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 96, head 3} \\ \text{BLA-Block} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 96, head 3} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 96, head 3} \\ \text{BLA-Block} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 128, head 4} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 128, head 4} \\ \text{BLA-Block} \end{bmatrix} \times 1$
	concat 2×2	concat 2×2	concat 2×2
stage 2	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 192, head 6} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 192, head 6} \\ \text{BLA-Block} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 192, head 6} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 192, head 6} \\ \text{BLA-Block} \end{bmatrix} \times 1$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 256, head 8} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 256, head 8} \\ \text{BLA-Block} \end{bmatrix} \times 1$
	concat 2×2	concat 2×2	concat 2×2
stage 3	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 384, head 12} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 384, head 12} \\ \text{BLA-Block} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 384, head 12} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 384, head 12} \\ \text{BLA-Block} \end{bmatrix} \times 9$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 512, head 16} \\ \text{Swin-Block} \\ \text{win. sz. } 7\times7, \\ \text{dim 512, head 16} \\ \text{BLA-Block} \end{bmatrix} \times 9$
	concat 2×2	concat 2×2	concat 2×2
stage 4	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 768, head 24} \\ \text{Swin-Block} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 768, head 24} \\ \text{Swin-Block} \end{bmatrix} \times 2$	$\begin{bmatrix} \text{win. sz. } 7\times7, \\ \text{dim 1024, head 32} \\ \text{Swin-Block} \end{bmatrix} \times 2$

Table 7: Detailed network architecture of BOAT-Swin.

	BOAT-CSWin-T	BOAT-CSWin-S	BOAT-CSWin-B
	conv	conv	conv
stage 1	[win. sz. 7×7, dim 64, head 2 BLA-Block] × 1	[win. sz. 7×7, dim 64, head 2 BLA-Block win. sz. 7×7, dim 64, head 2 CSWin-Block] × 1	[win. sz. 7×7, dim 96, head 4 BLA-Block win. sz. 7×7, dim 96, head 4 CSWin-Block] × 1
	conv	conv	conv
stage 2	[win. sz. 7×7, dim 128, head 4 BLA-Block win. sz. 7×7, dim 128, head 4 CSwin-Block] × 1	[win. sz. 7×7, dim 128, head 4 BLA-Block win. sz. 7×7, dim 128, head 4 CSWin-Block] × 2	[win. sz. 7×7, dim 192, head 8 BLA-Block win. sz. 7×7, dim 192, head 8 CSWin-Block] × 2
	conv	conv	conv
stage 3	[win. sz. 7×7, dim 256, head 8 BLA-Block win. sz. 7×7, dim 256, head 8 CSWin-Block] × 10 [win. sz. 7×7, dim 768, head 8 BLA-Block] × 1	[win. sz. 7×7, dim 256, head 8 BLA-Block win. sz. 7×7, dim 256, head 8 CSWin-Block] × 16	[win. sz. 7×7, dim 384, head 16 BLA-Block win. sz. 7×7, dim 384, head 16 CSWin-Block] × 16
	conv	conv	conv
stage 4	[win. sz. 7×7, dim 512, head 16 BLA-Block] × 1	[win. sz. 7×7, dim 512, head 16 BLA-Block win. sz. 7×7, dim 512, head 16 CSWin-Block] × 1	[win. sz. 7×7, dim 768, head 32 BLA-Block win. sz. 7×7, dim 768, head 32 CSWin-Block] × 1

Table 8: Detailed network architecture of BOAT-CSWin.