Vision Transformers with Hierarchical Attention

Yun Liu[1], Yu-Huan Wu[2], Guolei Sun[3†], Le Zhang[4†], Ajad Chhatkuli[3] and Luc Van Gool[3]

[1]Institute for Infocomm Research (I2R), A*STAR, Singapore 138632.
[2]Institute of High Performance Computing (IHPC), A*STAR, Singapore 138632.
[3]Computer Vision Lab, ETH Zürich, Zürich 8092, Switzerland.
[4]School of Information and Communication Engineering, UESTC, Chengdu 611731, China.

[†]Corresponding authors.

Abstract

This paper tackles the high computational/space complexity associated with Multi-Head Self-Attention (MHSA) in vanilla vision transformers. To this end, we propose Hierarchical MHSA (H-MHSA), a novel approach that computes self-attention in a hierarchical fashion. Specifically, we first divide the input image into patches as commonly done, and each patch is viewed as a token. Then, the proposed H-MHSA learns token relationships within local patches, serving as local relationship modeling. Then, the small patches are merged into larger ones, and H-MHSA models the global dependencies for the small number of the merged tokens. At last, the local and global attentive features are aggregated to obtain features with powerful representation capacity. Since we only calculate attention for a limited number of tokens at each step, the computational load is reduced dramatically. Hence, H-MHSA can efficiently model global relationships among tokens without sacrificing fine-grained information. With the H-MHSA module incorporated, we build a family of Hierarchical-Attention-based Transformer Networks, namely HAT-Net. To demonstrate the superiority of HAT-Net in scene understanding, we conduct extensive experiments on fundamental vision tasks, including image classification, semantic segmentation, object detection, and instance segmentation. Therefore, HAT-Net provides a new perspective for vision transformers. Code and pretrained models are available at https://github.com/yun-liu/HAT-Net.

Keywords: Vision transformer, hierarchical attention, global attention, local attention, scene understanding

1 Introduction

In the last decade, convolutional neural networks (CNNs) have been the go-to architecture in computer vision, owing to their powerful capability in learning representations from images/videos [1–12]. Meanwhile, in another field of natural language processing (NLP), the transformer architecture [13] has been the de-facto standard to handle long-range dependencies [14, 15]. Transformers rely heavily on self-attention to model global relationships of sequence data. Although global modeling is also essential for vision tasks, the 2D/3D structures of vision data make it less straightforward to apply transformers therein. This predicament was recently broken by Dosovitskiy *et al.* [16], by applying a pure transformer to sequences of image patches.

Motivated by [16], a large amount of literature on vision transformers has emerged to resolve the problems caused by the domain gap between

computer vision and NLP [17–21]. From our point of view, one major problem of vision transformers is that the sequence length of image patches is much longer than that of tokens (words) in an NLP application, thus leading to high computational/space complexity when computing the Multi-Head Self-Attention (**MHSA**). Some efforts have been dedicated to resolving this problem. ToMe [22] improves the throughput of existing ViT models [16] by systematically merging similar tokens through the utilization of a general and light-weight matching algorithm. PVT [19] and MViT [21] downsample the feature to compute attention in a reduced length of tokens but at the cost of losing fine-grained details. Swin Transformer [18] computes attention within small windows to model local relationships, and it gradually enlarges the receptive field by shifting windows and stacking more layers. From this point of view, Swin Transformer [18] may still be suboptimal because it works in a similar manner to CNNs and needs many layers to model long-range dependencies [16].

Building upon the discussed strengths of downsampling-based transformers [19, 21] and window-based transformers [18], each with its distinctive merits, we aim to harness their complementary advantages. Downsampling-based transformers excel at directly modeling global dependencies but may sacrifice fine-grained details, while window-based transformers effectively capture local dependencies but may fall short in global dependency modeling. As widely accepted, both global and local information is essential for visual scene understanding. Motivated by this insight, our approach seeks to amalgamate the strengths of both paradigms, enabling the direct modeling of both global and local dependencies.

To achieve this, we introduce the **Hierarchical Multi-Head Self-Attention (H-MHSA)**, a novel mechanism that enhances the flexibility and efficiency of self-attention computation in transformers. Our methodology begins by segmenting an image into patches, treating each patch akin to a token [16]. Rather than computing attention across all patches, we further organize these patches into small grids, performing attention computation within each grid. This step is instrumental in capturing local relationships and generating more discriminative local representations. Subsequently, we amalgamate these smaller patches into larger ones and treat the merged patches as new tokens, resulting in a substantial reduction in their number. This enables the direct modeling of global dependencies by calculating self-attention for the new tokens. Ultimately, the attentive features from both local and global hierarchies are aggregated to yield potent features with rich granularities. Notably, as the attention calculation at each step is confined to a small number of tokens, our hierarchical strategy mitigates the computational and space complexity of vanilla transformers. Empirical observations underscore the efficacy of this hierarchical self-attention mechanism, revealing improved generalization results in our experiments.

By simply incorporating H-MHSA, we build a family of **Hierarchical-Attention-based Transformer Networks (HAT-Net)**. To evaluate the efficacy of HAT-Net in scene understanding, we experiment HAT-Net for fundamental vision tasks, including image classification, semantic segmentation, object detection, and instance segmentation. Experimental results demonstrate that HAT-Net performs favorably against previous backbone networks. Note that H-MHSA is based on a very simple and intuitive idea, so H-MHSA is expected to provide a new perspective for the future design of vision transformers.

2 Related Work

Convolutional neural networks. More than two decades ago, LeCun *et al.* [23] built the first deep CNN, *i.e.*, LeNet, for document recognition. About ten years ago, AlexNet [1] introduced pooling layers into CNNs and pushed forward the state of the art of ImageNet classification [24] significantly. Since then, CNNs have become the de-facto standard of computer vision owing to its powerful ability in representation learning. Brilliant achievements have been seen in this direction. VGGNet [2] investigates networks of increasing depth using small (3×3) convolution filters. ResNet [3] manages to build very deep networks by resolving the gradient vanishing/exploding problem with residual connections [25]. GoogLeNet [26] presents the inception architecture [27, 28] using multiple branches with different convolution kernels. ResNeXt [29] improves ResNet [3] by replacing the 3×3 convolution in the bottleneck with a grouped convolution. DenseNets

[30] present dense connections, *i.e.*, using the feature maps of all preceding layers as inputs for each layer. MobileNets [31, 32] decompose the traditional convolution into a pointwise convolution and a depthwise separable convolution for acceleration, and an inverted bottleneck is proposed for ensuring accuracy. ShuffleNets [33, 34] further decompose the pointwise convolution into pointwise group convolution and channel shuffle to reduce computational cost. MansNet [35] proposes an automated mobile neural architecture search approach to search for a model with a good trade-off between accuracy and latency. EfficientNet [36] introduces a scaling method to uniformly scale depth/width/resolution dimensions of the architecture searched by MansNet [35]. The above advanced techniques are the engines driving the development of computer vision in the last decade. This paper aims at improving feature representation learning by designing new transformers.

Self-attention mechanism. Inspired by the human visual system, the self-attention mechanism is usually adopted to enhance essential information and suppress noisy information. STN [37] presents the spatial attention mechanism through learning an appropriate spatial transformation for each input. Chen *et al.* [38] proposed the channel attention model and achieved promising results on the image captioning task. Wang *et al.* [39] explored self-attention in well-known residual networks [3]. SENet [40] applies channel attention to backbone network design and boosts the accuracy of ImageNet classification [24]. CBAM [41] sequentially applies channel and spatial attention for adaptive feature refinement in deep networks. BAM [42] produces a 3D attention map by combining channel and spatial attention. SK-Net [43] uses channel attention to selectively fuse multiple branches with different kernel sizes. Non-local network [44] presents non-local attention for capturing long-range dependencies. ResNeSt [45] is a milestone in this direction. It applies channel attention on different network branches to capture cross-feature interactions and learn diverse representations. Our work shares some similarities with these works by applying self-attention for learning feature representations. The difference is that we propose H-MHSA to learn global relationships rather than a simple feature recalibration using spatial or channel attention in these works.

Vision transformer. Transformer [13] entirely relies on self-attention to handle long-range dependencies of sequence data. It was first proposed for NLP tasks [14, 15]. In order to apply transformers on image data, Dosovitskiy *et al.* [16] split an image into patches and treated them as tokens. Then, a pure transformer [13] can be adopted. Such a vision transformer (ViT) attains competitive accuracy for ImageNet classification [24]. More recently, lots of efforts have been dedicated to improving ViT. T2T-ViT [46] proposes to split an image into tokens of overlapping patches so as to represent local structures by surrounding tokens. CaiT [47] builds a deeper transformer network by introducing a per-channel weighting and specific class attention. DeepViT [48] proposes Re-attention to re-generate attention maps to increase their diversity at different layers. DeiT [49] presents a knowledge distillation strategy for improving the training of ViT [16]. Srinivas *et al.* [50] tried to add the bottleneck structure to vision transformers. Some works build pyramid transformer networks to generate multi-scale features [17–21]. PVT [19] adopts convolution operation to downsample the feature map in order to reduce the sequence length in MHSA, thus reducing the computational load. Similar to PVT [19], MViT [21] utilizes pooling to compute attention on a reduced sequence length. Swin Transformer [18] computes attention within small windows and shifts windows to gradually enlarge the receptive field. CoaT [20] computes attention in the channel dimension rather than in the traditional spatial dimension. ToMe [22] enhances the throughput of existing ViT models [16] without requiring retraining, which is achieved by gradually combining similar tokens in a transformer using a matching algorithm. In this paper, we introduce a novel design to reduce the computational complexity of MHSA and learn both the global and local relationship modeling through vision transformers.

Vision MLP networks. While CNNs and vision transformers have been widely adopted for computer vision tasks, Tolstikhin *et al.* [51] challenged the necessity of convolutions and attention mechanisms. They introduced the MLP-Mixer architecture, which relies solely on multi-layer perceptrons (MLPs). MLP-Mixer incorporates two types of layers: one applies MLPs independently

to image patches, facilitating the mixing of per-location features, and the other applies MLPs across patches, enabling the mixing of spatial information. Despite lacking convolutions and attention, MLP-Mixer demonstrated competitive performance in image classification compared to state-of-the-art models. Liu *et al.* [52] introduced gMLP, an MLP-based model with gating, showcasing its comparable performance to transformers in crucial language and vision applications. In contrast to other MLP-like models that encode spatial information along flattened spatial dimensions, Vision Permutator [53] uniquely encodes feature representations along height and width dimensions using linear projections. Wang *et al.* [54] proposed a novel positional spatial gating unit, leveraging classical relative positional encoding to efficiently capture cross-token relations for token mixing. Despite these advancements, the performance of vision MLP networks still lags behind that of vision transformers. In this paper, we focus on the design of a new vision transformer network.

3 Methodology

In this section, we first provide a brief review of vision transformers [16] in Sec. 3.1. Then, we present the proposed H-MHSA and analyze its computational complexity in Sec. 3.2. Finally, we describe the configuration details of the proposed HAT-Net in Sec. 3.3.

3.1 Review of Vision Transformers

Transformer [13, 16] heavily relies on MHSA to model long-range relationships. Suppose $\mathbf{X} \in \mathbb{R}^{H\times W\times C}$ denotes the input, where H, W, and C are the height, width, and the feature dimension, respectively. We reshape $\mathbf{X}$ and define the query $\mathbf{Q}$, key $\mathbf{K}$, value $\mathbf{V}$ as

$$\begin{gathered}\mathbf{X} \in \mathbb{R}^{H\times W\times C} \rightarrow \mathbf{X} \in \mathbb{R}^{(H\times W)\times C}, \\ \mathbf{Q} = \mathbf{X}\mathbf{W}^q, \qquad \mathbf{K} = \mathbf{X}\mathbf{W}^k, \qquad \mathbf{V} = \mathbf{X}\mathbf{W}^v,\end{gathered} \tag{1}$$

where $\mathbf{W}^q \in \mathbb{R}^{C\times C}$, $\mathbf{W}^k \in \mathbb{R}^{C\times C}$, and $\mathbf{W}^v \in \mathbb{R}^{C\times C}$ are the trainable weight matrices of linear transformations. With a mild assumption that the input and output have the same dimension, the traditional MHSA can be formulated as

$$\mathbf{A} = \mathrm{Softmax}(\mathbf{Q}\mathbf{K}^{\mathrm{T}}/\sqrt{d})\mathbf{V}, \tag{2}$$

in which $\sqrt{d}$ means an approximate normalization, and the Softmax function is applied to the rows of the matrix. Note that we omit the concept of multiple heads here for simplicity. In Eq. (2), the matrix product of $\mathbf{Q}\mathbf{K}^{\mathrm{T}}$ first computes the similarity between each pair of tokens. Each new token is then derived over the combination of all tokens according to the similarity. After the computation of MHSA, a residual connection is further added to ease the optimization, like

$$\begin{gathered}\mathbf{X} \in \mathbb{R}^{(H\times W)\times C} \rightarrow \mathbf{X} \in \mathbb{R}^{H\times W\times C}, \\ \mathbf{A}' = \mathbf{A}\mathbf{W}^p + \mathbf{X},\end{gathered} \tag{3}$$

in which $\mathbf{W}^p \in \mathbb{R}^{C\times C}$ is a trainable weight matrix for feature projection. At last, a multi-layer perceptron (MLP) is adopted to enhance the representation, like

$$\mathbf{Y} = \mathrm{MLP}(\mathbf{A}') + \mathbf{A}', \tag{4}$$

where $\mathbf{Y}$ denotes the output of a transformer block.

It is easy to infer that the computational complexity of MHSA (Eq. (2)) is

$$\Omega(\mathrm{MHSA}) = 3HWC^2 + 2H^2W^2C. \tag{5}$$

Similarly, the space complexity (memory consumption) also includes the term of $O(H^2W^2)$. As commonly known, $O(H^2W^2)$ could become very large for high-resolution inputs. This limits the applicability of transformers for vision tasks. Motivated by this, we aim at improving MHSA to reduce such complexity and maintain the capacity of global relationship modeling without the risk of sacrificing performances.

3.2 Hierarchical Multi-Head Self-Attention

In this section, we present an approach to alleviate the computational and space demands associated with Eq. (2) through the utilization of our proposed H-MHSA mechanism. Rather than computing attention over the entire input, we adopt a hierarchical strategy, allowing each step to process only a limited number of tokens.

The initial step concentrates on local attention computation. Assuming the input feature map is

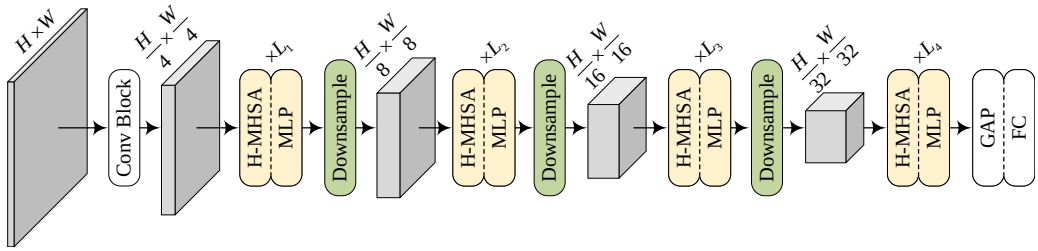

Fig. 1 Illustration of the proposed HAT-Net. GAP: global average pooling; FC: fully-connected layer. $\times L_i$ means that the transformer block is repeated for L_i times. H and W denote the height and width of the input image, respectively.

Table 1 Network configurations of HAT-Net. The settings of building blocks are shown in brackets, with the number of blocks stacked. For the first stage, each convolution has C channels and a stride of S. For the other four stages, each MLP uses a $K \times K$ DW-Conv and an expansion ratio of E. Note that we omit the downsampling operation after the t-th stage ($t = \{2, 3, 4\}$) for simplicity. "#Param" refers to the number of parameters.

Stage	Input Size	Operator	HAT-Net-Tiny	HAT-Net-Small	HAT-Net-Medium	HAT-Net-Large
1	224×224	3×3 conv.	$C = 16$, $S = 2$ $C = 48$, $S = 2$	$C = 16$, $S = 2$ $C = 64$, $S = 2$	$C = 16$, $S = 2$ $C = 64$, $S = 2$	$C = 16$, $S = 2$ $C = 64$, $S = 2$
2	56×56	H-MHSA MLP	$\begin{bmatrix} C = 48 \\ K = 3 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 64 \\ K = 3 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 64 \\ K = 5 \\ E = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 64 \\ K = 3 \\ E = 8 \end{bmatrix} \times 3$
3	28×28	H-MHSA MLP	$\begin{bmatrix} C = 96 \\ K = 3 \\ E = 8 \end{bmatrix} \times 2$	$\begin{bmatrix} C = 128 \\ K = 3 \\ E = 8 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 128 \\ K = 3 \\ E = 8 \end{bmatrix} \times 6$	$\begin{bmatrix} C = 128 \\ K = 3 \\ E = 8 \end{bmatrix} \times 8$
4	14×14	H-MHSA MLP	$\begin{bmatrix} C = 240 \\ K = 3 \\ E = 4 \end{bmatrix} \times 6$	$\begin{bmatrix} C = 320 \\ K = 3 \\ E = 4 \end{bmatrix} \times 8$	$\begin{bmatrix} C = 320 \\ K = 5 \\ E = 4 \end{bmatrix} \times 18$	$\begin{bmatrix} C = 320 \\ K = 3 \\ E = 4 \end{bmatrix} \times 27$
5	7×7	H-MHSA MLP	$\begin{bmatrix} C = 384 \\ K = 3 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 512 \\ K = 3 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 512 \\ K = 3 \\ E = 4 \end{bmatrix} \times 3$	$\begin{bmatrix} C = 640 \\ K = 3 \\ E = 4 \end{bmatrix} \times 3$
	1×1	-	Global Average Pooling, 1000-d FC, Softmax			
#Param			12.7M	25.7M	42.9M	63.1M

denoted as $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$, we partition the feature map into small grids of size $G_1 \times G_1$ and reshape it as follows:

$$\begin{aligned} \mathbf{X} \in \mathbb{R}^{H \times W \times C} &\rightarrow \mathbf{X}_1 \in \mathbb{R}^{(\frac{H}{G_1} \times G_1) \times (\frac{W}{G_1} \times G_1) \times C} \\ &\rightarrow \mathbf{X}_1 \in \mathbb{R}^{(\frac{H}{G_1} \times \frac{W}{G_1}) \times (G_1 \times G_1) \times C}. \end{aligned} \tag{6}$$

The query, key, and value are then calculated by

$$\mathbf{Q}_1 = \mathbf{X}_1 \mathbf{W}_1^q, \quad \mathbf{K}_1 = \mathbf{X}_1 \mathbf{W}_1^k, \quad \mathbf{V}_1 = \mathbf{X}_1 \mathbf{W}_1^v, \tag{7}$$

where $\mathbf{W}_1^q, \mathbf{W}_1^k, \mathbf{W}_1^v \in \mathbb{R}^{C \times C}$ are trainable weight matrices. Subsequently, Eq. (2) is applied to generate the local attentive feature $\mathbf{A}_1$. To ease network optimization, we reshape $\mathbf{A}_1$ back to the shape of $\mathbf{X}$ through

$$\begin{aligned} \mathbf{A}_1 &\in \mathbb{R}^{(\frac{H}{G_1} \times \frac{W}{G_1}) \times (G_1 \times G_1) \times C} \\ \rightarrow \mathbf{A}_1 &\in \mathbb{R}^{(\frac{H}{G_1} \times G_1) \times (\frac{W}{G_1} \times G_1) \times C} \\ \rightarrow \mathbf{A}_1 &\in \mathbb{R}^{H \times W \times C}, \end{aligned} \tag{8}$$

and incorporate a residual connection:

$$\mathbf{A}_1 = \mathbf{A}_1 + \mathbf{X}. \tag{9}$$

As the local attentive feature $\mathbf{A}_1$ is computed within each small $G_1 \times G_1$ grid, a substantial reduction in computational and space complexity is achieved.

The second step focuses on global attention calculation. Here, we downsample $\mathbf{A}_1$ by a factor of G_2 during the computation of key and value matrices. This downsampling enables efficient global attention calculation, treating each $G_2 \times G_2$ grid as a token. This process can be expressed as

$$\widehat{\mathbf{A}}_1 = \text{AvePool}_{G_2}(\mathbf{A}_1), \tag{10}$$

where $\text{AvePool}_{G_2}(\cdot)$ denotes downsampling a feature map by G_2 times using average pooling with both the kernel size and stride set to G_2. Consequently, we have $\widehat{\mathbf{A}}_1 \in \mathbb{R}^{\frac{H}{G_2} \times \frac{W}{G_2} \times C}$. We then reshape $\mathbf{A}_1$ and $\widehat{\mathbf{A}}_1$ as follows:

$$\begin{aligned} \mathbf{A}_1 \in \mathbb{R}^{H \times W \times C} &\rightarrow \mathbf{A}_1 \in \mathbb{R}^{(H \times W) \times C}, \\ \widehat{\mathbf{A}}_1 \in \mathbb{R}^{\frac{H}{G_2} \times \frac{W}{G_2} \times C} &\rightarrow \widehat{\mathbf{A}}_1 \in \mathbb{R}^{(\frac{H}{G_2} \times \frac{W}{G_2}) \times C}. \end{aligned} \tag{11}$$

Following this, we compute the query, key, and value as

$$\mathbf{Q}_2 = \mathbf{A}_1 \mathbf{W}_2^q, \quad \mathbf{K}_2 = \widehat{\mathbf{A}}_1 \mathbf{W}_2^k, \quad \mathbf{V}_2 = \widehat{\mathbf{A}}_1 \mathbf{W}_2^v, \tag{12}$$

where $\mathbf{W}_2^q, \mathbf{W}_2^k, \mathbf{W}_2^v \in \mathbb{R}^{C \times C}$ are trainable weight matrices. It is easy to derive that we have $\mathbf{Q}_2 \in \mathbb{R}^{(H \times W) \times C}$, $\mathbf{K}_2 \in \mathbb{R}^{(\frac{H}{G_2} \times \frac{W}{G_2}) \times C}$, and $\mathbf{V}_2 \in \mathbb{R}^{(\frac{H}{G_2} \times \frac{W}{G_2}) \times C}$. Subsequently, Eq. (2) is called to obtain the global attentive feature $\mathbf{A}_2 \in \mathbb{R}^{(H \times W) \times C}$, followed by a reshaping operation:

$$\mathbf{A}_2 \in \mathbb{R}^{(H \times W) \times C} \rightarrow \mathbf{A}_2 \in \mathbb{R}^{H \times W \times C}. \tag{13}$$

The final output of H-MHSA is given by

$$\text{H-MHSA}(\mathbf{X}) = (\mathbf{A}_1 + \mathbf{A}_2)\mathbf{W}^p + \mathbf{X}, \tag{14}$$

where $\mathbf{W}^p$ has the same meaning as in Eq. (3). In this way, H-MHSA effectively models both local and global relationships, akin to vanilla MHSA.

The computational complexity of H-MHSA can be expressed as

$$\Omega(\text{H-MHSA}) = HWC(4C + 2G_1^2) + 2\frac{HW}{G_2^2}C(C + HW). \tag{15}$$

Compared to Eq. (5), this represents a reduction in computational complexity from $O(H^2W^2)$ to $O(HWG_1^2 + \frac{H^2W^2}{G_2^2})$. The same conclusion can be easily derived for space complexity.

We continue by comparing H-MHSA with existing vision transformers, highlighting distinctive features. Swin Transformer [18] focuses on modeling local relationships, progressively expanding the receptive field through shifted windows and additional layers. Conversely, PVT [19] prioritizes global relationships through downsampling key and value matrices but overlooks local information. In contrast, our proposed H-MHSA excels by concurrently capturing both local and global relationships. While Swin Transformer employs a fixed window size (*i.e.*, a fixed-size bias matrix), and PVT uses a constant downsampling ratio (*i.e.*, a convolution with the kernel size equal to the stride), these approaches necessitate retraining on the ImageNet dataset [24] for any reparameterization. In contrast, the parameter-free nature of G_1 and G_2 in H-MHSA allows flexible configuration adjustments for downstream vision tasks without the need for retraining on ImageNet.

In computer vision, achieving a comprehensive understanding of scenes relies on the simultaneous consideration of both global and local information. Within the framework of our proposed H-MHSA, global self-attention calculation (Eq. (10)-(13)) is instrumental in establishing the foundation for scene interpretation, enabling the recognition of overarching patterns and aiding in high-level decision-making processes. Concurrently, local self-attention calculation (Eq. (6)-(9)) is crucial for refining the understanding of individual components within the larger context, facilitating more detailed and nuanced scene analysis. H-MHSA excels in striking the delicate balance between global and local information, thereby facilitating a nuanced and accurate comprehension of diverse scenes. In essence, the seamless integration of global and local self-attention within the H-MHSA framework empowers transformers to navigate the intricacies of scene understanding, facilitating context-aware decision-making.

3.3 Network Architecture

This part introduces the network architecture of HAT-Net. We follow the common practice in CNNs to use a global average pooling layer and a fully connected layer to predict image classes [18]. This is different from existing transformers which rely on another 1D class token to make predictions [16, 17, 19–21, 46–49, 55–57]. We also observe that

existing transformers [16–21, 46–49] usually adopt the GELU function [58] for nonlinear activation. However, GELU is memory-hungry during network training. We empirically found that the SiLU function [59], originally coined in [58], performs on-par with GELU and is more memory-friendly. Hence, HAT-Net uses SiLU [59] for nonlinear activation. Besides, we add a depthwise separable convolution (DW-Conv) [31] inside the MLP as widely done.

The overall architecture of HAT-Net is illustrated in Fig. 1. At the beginning of HAT-Net, instead of flattening image patches [16], we apply two sequential vanilla 3×3 convolutions, each of which has a stride of 2, to downsample the input image into 1/4 scale. Then, we stack H-MHSA and MLP alternatively, which can be divided into four stages with pyramid feature scales of 1/4, 1/8, 1/16, and 1/32, respectively. For feature downsampling at the end of each stage, a vanilla 3×3 convolution with a stride of 2 is used. The configuration details of HAT-Net are summarized in Table 1. We provide four versions of HAT-Net: HAT-Net-Tiny, HAT-Net-Small, HAT-Net-Medium, and HAT-Net-Large, whose number of parameters is similar to ResNet18, ResNet50, ResNet101, and ResNet152 [3], respectively. We only adopt simple parameter settings without careful tuning to demonstrate the effectiveness and generality of HAT-Net. The dimension of each head in the multi-head setting is set to 48 for HAT-Net-Tiny and 64 for other versions.

To enhance the applicability of HAT-Net across diverse vision tasks, we present guidelines for configuring the parameter-free G_1 and G_2. While established models like Swin Transformer [18] adhere to a fixed window size of 7, and PVT [19] employs a set of constant downsampling ratios $8, 4, 2$ for the t-th stage ($t = 2, 3, 4$), we advocate for certain adjustments. Practically, we find that a window size of 8 is more pragmatic than 7, given that input resolutions often align with multiples of 8. Moreover, augmenting downsampling ratios serves to mitigate computational complexity. Consequently, for image classification on the ImageNet dataset [24], where the standard input resolution is 224×224 pixels, we designate $G_1 = 8, 7, 7$ and $G_2 = 8, 4, 2$ for the t-th stage ($t = 2, 3, 4$). Here, a window size of 7 is necessitated by the chosen resolution, and small downsampling rates are in line with the approach taken by PVT [19]. In scenarios involving downstream tasks like semantic segmentation, object detection, and instance segmentation, where input resolutions tend to be larger, we opt for $G_1 = 8, 8, 8$ for convenience and $G_2 = 16, 8, 4$ to curtail computational expenses. For a comprehensive analysis of the impact of different G_1 and G_2 settings, we conduct an ablation study in Sec. 4.4.

4 Experiments

To show the superiority of HAT-Net in feature representation learning, this section evaluates HAT-Net for image classification, semantic segmentation, object detection and instance segmentation.

4.1 Image Classification

Experimental setup. The ImageNet dataset [24] consists of 1.28M training images and 50K validation images from 1000 categories. We adopt the training set to train our networks and the validation set to test the performance. We implement HAT-Net using the popular PyTorch framework [60]. For a fair comparison, we follow the same training protocol as DeiT [49], which is the standard protocol for training transformer networks nowadays. Specifically, the input images are randomly cropped to 224×224 pixels, followed by random horizontal flipping and *mixup* [61] for data augmentation. Label smoothing [27] is used to avoid overfitting. The AdamW optimizer [62] is adopted with the momentum of 0.9, the weight decay of 0.05, and a mini-batch size of 128 per GPU by default. The initial learning rate is set to $1e$-3, which decreases following the cosine learning rate schedule [63]. The training process lasts for 300 epochs on eight NVIDIA Tesla V100 GPUs. Note that for ablation studies, we utilize a mini-batch size of 64 and 100 training epochs to save training time. Moreover, we set $G_1 = \{8, 7, 7\}$ and $G_2 = \{8, 4, 2\}$ for t-th stage ($t = \{2, 3, 4\}$), respectively. The fifth stage can be processed directly using the vanilla MHSA mechanism. For model evaluation, we apply a center crop of 224×224 pixels on validation images to evaluate the recognition accuracy. We report the top-1 classification accuracy on the ImageNet validation set [24] as well as the number of parameters and the number of FLOPs for each model.

Table 2 Comparison to state-of-the-art methods on the ImageNet validation set [24]. "*" indicates the performance of a method using the default training setting in the original paper. "#Param" and "#FLOPs" refer to the number of parameters and the number of FLOPs, respectively. "†" marks models that use the input size of 384×384; otherwise, models use the input size of 224×224.

Arch.	Models	#Param	#FLOPs	Top-1 Acc.
CNN	ResNet18* [3]	11.7M	1.8G	69.8
	ResNet18 [3]	11.7M	1.8G	68.5
Trans	DeiT-Ti/16 [49]	5.7M	1.3G	72.2
	PVT-Tiny [19]	13.2M	1.9G	75.1
	PVTv2-B1 [64]	13.1M	2.1G	78.7
	HAT-Net-Tiny	12.7M	2.0G	79.8
CNN	ResNet50* [3]	25.6M	4.1G	76.1
	ResNet50 [3]	25.6M	4.1G	78.5
	ResNeXt50-32x4d* [29]	25.0M	4.3G	77.6
	ResNeXt50-32x4d [29]	25.0M	4.3G	79.5
	RegNetY-4G [65]	20.6M	4.0G	80.0
	ResNeSt-50 [45]	27.5M	5.4G	81.1
Trans	ToMe-ViT-S/16 [22]	22.1M	2.7G	79.4
	DeiT-S/16 [49]	22.1M	4.6G	79.8
	T2T-ViT$_t$-14 [46]	21.5M	5.2G	80.7
	TNT-S [55]	23.8M	5.2G	81.3
	CvT-13 [66]	20.0M	4.5G	81.6
	PVT-Small [19]	24.5M	3.8G	79.8
	PVTv2-B2 [64]	25.4M	4.0G	82.0
	Swin-T [18]	28.3M	4.5G	81.3
	Twins-SVT-S [67]	24.0M	2.8G	81.7
	HAT-Net-Small	25.7M	4.3G	82.6
CNN	ResNet101* [3]	44.7M	7.9G	77.4
	ResNet101 [3]	44.7M	7.9G	79.8
	ResNeXt101-32x4d* [29]	44.2M	8.0G	78.8
	ResNeXt101-32x4d [29]	44.2M	8.0G	80.6
	RegNetY-8G [65]	39.2M	8.0G	81.7
	ResNeSt-101 [45]	48.3M	10.3G	83.0
Trans	T2T-ViT$_t$-19 [46]	39.2M	8.4G	81.4
	CvT-21 [66]	31.5M	7.1G	82.5
	MViT-B-16 [21]	37.0M	7.8G	82.5
	PVT-Medium [19]	44.2M	6.7G	81.2
	PVTv2-B3 [64]	45.2M	6.9G	83.2
	Swin-S [18]	49.6M	8.7G	83.0
	HAT-Net-Medium	42.9M	8.3G	84.0
CNN	ResNet152* [3]	60.2M	11.6G	78.3
	ResNeXt101-64x4d* [29]	83.5M	15.6G	79.6
	ResNeXt101-64x4d [29]	83.5M	15.6G	81.5
Trans	ViT-B/16† [16]	86.6M	55.4G	77.9
	ViT-L/16† [16]	304.3M	190.7G	76.5
	ToMe-ViT-L/16 [22]	304.3M	22.3G	84.2
	DeiT-B/16 [49]	86.6M	17.6G	81.8
	MViT-B-24 [21]	53.5M	10.9G	83.1
	TNT-B [55]	65.6M	14.1G	82.8
	PVT-Large [19]	61.4M	9.8G	81.7
	PVTv2-B4 [64]	62.6M	10.1G	83.6
	Swin-B [18]	87.8M	15.4G	83.3
	Twins-SVT-B [67]	56.0M	8.3G	83.2
	HAT-Net-Large	63.1M	11.5G	84.2

Experimental results. We compare HAT-Net with state-of-the-art network architectures, including CNN-based ones like ResNet [3], ResNeXt [29], RegNetY [65], ResNeSt [45], and transformer-based ones like ViT [16], DeiT [49], T2T-ViT [46], TNT [55], CvT [66], MViT [21], PVT [19], Swin Transformer [18], Twins [67], ToMe [22]. The results are summarized in Table 2. We can observe that HAT-Net achieves state-of-the-art performance. Specifically, with similar numbers of parameters and FLOPs, HAT-Net-Tiny, HAT-Net-Small, HAT-Net-Medium, and HAT-Net-Large outperforms the second best results by 1.1%, 0.6%, 0.8%, and 0.6% in terms of the top-1 accuracy, respectively. Since the performance for image classification implies the ability of a network for learning feature representations, the above comparison suggests that the proposed HAT-Net has great potential for generic scene understanding.

Table 3 Experimental results on the ADE20K validation dataset [68] for semantic segmentation. We replace the backbone of Semantic FPN [69] with various network architectures. The number of FLOPs is calculated with the input size of 512×512.

Backbone	Semantic FPN [69]		
	#Param (M) ↓	FLOPs (G) ↓	mIoU (%) ↑
ResNet-18 [3]	15.5	31.9	32.9
PVT-Tiny [19]	17.0	32.1	35.7
PVTv2-B1 [64]	17.8	33.1	41.5
HAT-Net-Tiny	15.9	33.2	43.6
ResNet-50 [3]	28.5	45.4	36.7
PVT-Small [19]	28.2	42.9	39.8
Swin-T [18]	31.9	46	41.5
Twins-SVT-S [67]	28.3	37	43.2
PVTv2-B2 [64]	29.1	44.1	46.1
HAT-Net-Small	29.5	49.6	46.6
ResNet-101 [3]	47.5	64.8	38.8
ResNeXt-101-32x4d [29]	47.1	64.6	39.7
PVT-Medium [19]	48.0	59.4	41.6
Swin-S [18]	53.2	70	45.2
Twins-SVT-B [67]	60.4	67	45.3
PVTv2-B3 [64]	49.0	60.7	47.3
HAT-Net-Medium	46.7	74.7	49.3
ResNeXt-101-64x4d [29]	86.4	104.2	40.2
PVT-Large [19]	65.1	78.0	42.1
Swin-B [18]	91.2	107	46.0
Twins-SVT-L [67]	102	103.7	46.7
PVTv2-B4 [64]	66.3	79.6	48.6
HAT-Net-Large	66.8	96.4	49.5

4.2 Semantic Segmentation

Experimental setup. We continue by applying HAT-Net to a fundamental downstream vision task, semantic segmentation, which aims at predicting a class label for each pixel in an image. We follow [19, 67] to replace the backbone of the well-known segmentor, Semantic FPN [69], with

Table 4 Object detection results with RetinaNet [70] and instance segmentation results with Mask R-CNN [5] on the MS-COCO `val2017` set [71]. "R" and "X" represent ResNet [3] and ResNeXt [29], respectively. The number of FLOPs is computed with the input size of 800 × 1280.

Backbone	Object Detection								Instance Segmentation							
	#Param (M) ↓	#FLOPs (G)↓	RetinaNet [70]						#Param (M) ↓	#FLOPs (G) ↓	Mask R-CNN [5]					
			AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L			AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
R-18 [3]	21.3	190	31.8	49.6	33.6	16.3	34.3	43.2	31.2	209	34.0	54.0	36.7	31.2	51.0	32.7
ViL-Tiny [72]	16.6	204	40.8	61.3	43.6	26.7	44.9	53.6	26.9	223	41.4	63.5	45.0	38.1	60.3	40.8
PVT-Tiny [19]	23.0	205	36.7	56.9	38.9	22.6	38.8	50.0	32.9	223	36.7	59.2	39.3	35.1	56.7	37.3
HAT-Net-Tiny	21.6	212	42.5	63.3	45.8	26.9	46.1	56.6	31.8	231	43.1	65.4	47.4	39.7	62.5	42.4
R-50 [3]	37.7	239	36.3	55.3	38.6	19.3	40.0	48.8	44.2	260	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small [19]	34.2	261	40.4	61.3	43.0	25.0	42.9	55.7	44.1	280	40.4	62.9	43.8	37.8	60.1	40.3
Swin-T [18]	38.5	248	41.5	62.1	44.2	25.1	44.9	55.5	47.8	264	42.2	64.6	46.2	39.1	61.6	42.0
ViL-Small [72]	35.7	292	44.2	65.2	47.6	28.8	48.0	57.8	45.0	310	44.9	67.1	49.3	41.0	64,2	44.1
Twins-SVT-S [67]	34.3	236	43.0	64.2	46.3	28.0	46.4	57.5	44.0	254	43.4	66.0	47.3	40.3	63.2	43.4
HAT-Net-Small	35.5	286	44.8	65.8	48.1	28.8	48.6	59.5	45.4	303	45.2	67.6	49.9	41.6	64.6	44.7
R-101 [3]	56.7	315	38.5	57.8	41.2	21.4	42.6	51.1	63.2	336	40.4	61.1	44.2	36.4	57.7	38.8
X-101-32x4d [29]	56.4	319	39.9	59.6	42.7	22.3	44.2	52.5	62.8	340	41.9	62.5	45.9	37.5	59.4	40.2
PVT-Medium [19]	53.9	349	41.9	63.1	44.3	25.0	44.9	57.6	63.9	367	42.0	64.4	45.6	39.0	61.6	42.1
Swin-S [18]	59.8	336	44.5	65.7	47.5	27.4	48.0	59.9	69.1	354	44.8	66.6	48.9	40.9	63.4	44.2
HAT-Net-Medium	52.7	405	45.9	66.9	49.2	29.7	50.0	61.6	62.6	424	47.0	69.0	51.5	42.7	66.0	46.0
X-101-64x4d [29]	95.5	473	41.0	60.9	44.0	23.9	45.2	54.0	101.9	493	42.8	63.8	47.3	38.4	60.6	41.3
PVT-Large [19]	71.1	450	42.6	63.7	45.4	25.8	46.0	58.4	81.0	469	42.9	65.0	46.6	39.5	61.9	42.5
Twins-SVT-B [67]	67.0	376	45.3	66.7	48.1	28.5	48.9	60.6	76.3	395	45.2	67.6	49.3	41.5	64.5	44.8
HAT-Net-Large	73.1	519	46.3	67.2	49.6	30.0	50.6	62.4	82.7	537	47.4	69.3	52.1	43.1	66.5	46.6

HAT-Net or other backbone networks for a fair comparison. Experiments are conducted on the challenging ADE20K dataset [68]. This dataset has 20000 training images, 2000 validation images, and 3302 testing images. We train Semantic FPN [69] using the training set and evaluate on the validation set. The training optimizer is AdamW [62] with weight decay of 1e-4. We apply the *poly* learning rate schedule with $\gamma = 0.9$ and the initial learning rate of 1e-4. During training, the batch size is 16, and each image has a resolution of 512 × 512 through resizing and cropping. During testing, each image is resized to the shorter side of 512 pixels, without multi-scale testing or flipping. We adopt the well-known MMSegmentation toolbox [73] for the above experiments. We set $G_1 = \{8, 8, 8\}$ and $G_2 = \{16, 8, 4\}$ for the t-th stage ($t = \{2, 3, 4\}$), respectively.

Experimental results. The results are depicted in Table 3. We compare with typical CNN networks, *i.e.*, ResNets [3] and ResNeXts [29], and transformer networks, *i.e.*, Swin Transformer [18], PVT [19], PVTv2 [64] and Twins-SVT [67]. As can be observed, the proposed HAT-Net achieves significantly better performance than previous competitors. Specifically, HAT-Net-Tiny, HAT-Net-Small, HAT-Net-Medium, and HAT-Net-Large attain 1.9%, 0.4%, 1.9%, and 0.7% higher mIoU than the second better results with similar number of parameters and FLOPs. This demonstrates the superiority of HAT-Net in learning effective feature representations for dense prediction tasks.

4.3 Object Detection and Instance Segmentation

Experimental setup. Since object detection and instance segmentation are also fundamental downstream vision tasks, we apply HAT-Net to both tasks for further evaluating its effectiveness. Specifically, we utilize two well-known detectors, *i.e.*, RetinaNet [70] for object detection and Mask R-CNN [5] for instance segmentation. HAT-Net is compared to some well-known CNN and transformer networks by only replacing the backbone of the above two detectors. Experiments are conducted on the large-scale MS-COCO dataset [71] by training on the `train2017` set (∼118K images) and evaluating on the `val2017` set (5K images).

Table 5 Ablation studies for the hierarchical attention in HAT-Net. The configuration of HAT-Net-Small is adopted for all experiments. "✔" indicates that we replace the window attention [18] with the hierarchical attention at the i-th stage. "Top-1 Acc" is the top-1 accuracy on the ImageNet validation dataset [24]. "mIoU" is the mean IoU for semantic segmentation on the ADE20K dataset [68].

Design	#Stage 2	3	4	Top-1 Acc	mIoU
1				78.2	42.1
2	✔			78.2	42.4
3	✔	✔		78.4	42.5
4	✔	✔	✔	79.3	43.4

We adopt MMDection toolbox [74] for experiments and follow the experimental settings of PVT [19] for a fair comparison. During training, we initialize the backbone weights with the ImageNet-pretrained models. The detectors are fine-tuned using the AdamW optimizer [62] with an initial learning rate of $1e$-4 that is decreased by 10 times after the 8^{th} and 11^{th} epochs, respectively. The whole training lasts for 12 epochs with a batch size of 16. Each image is resized to a shorter side of 800 pixels, but the longer side is not allowed to exceed 1333 pixels. We set $G_1 = \{8, 8, 8\}$ and $G_2 = \{16, 8, 4\}$ for the t-th stage ($t = \{2, 3, 4\}$), respectively.

Experimental results. The results are displayed in Table 4. As can be seen, HAT-Net substantially improves the accuracy over other network architectures with a similar number of parameters. Twins-SVT [67] combines the advantages of PVT [19] and Swin Transformer [18] by alternatively stacking their basic blocks. When RetinaNet [70] is adopted as the detector, HAT-Net-Small attains 1.8%, 1.6% and 1.8% higher results than Twins-SVT-S [67] in terms of AP, AP_{50} and AP_{75}, respectively. Correspondingly, HAT-Net-Large gets 1.0%, 0.5% and 1.5% higher results than Twins-SVT-B [67]. With Mask R-CNN [5] as the detector, HAT-Net-Large achieves 2.2%, 1.7% and 2.8% higher results than Twins-SVT-B [67] in terms of bounding box metrics AP^b, AP^b_{50} and AP^b_{75}, respectively. HAT-Net-Large achieves 1.6%, 2.0% and 1.8% higher results than Twins-SVT-B [67] in terms of mask metrics AP^m, AP^m_{50} and AP^m_{75}, respectively. Such significant improvement in object detection and instance segmentation shows the superiority of HAT-Net in learning effective feature representations.

Table 6 Ablation studies for the settings of G_1 and G_2 in HAT-Net. The performance assessment is conducted using Mask R-CNN [5] with HAT-Net-Small as the backbone. Evaluation results are reported on the MS-COCO `val2017` dataset [71].

Settings		#FLOPs (G)↓	Mask R-CNN [5]					
G_1	G_2		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
{8, 8, 8}	{12, 4, 2}	338	45.7	67.8	50.4	41.7	64.8	44.7
{8, 8, 8}	{12, 6, 3}	313	45.3	67.6	49.7	41.6	64.8	44.9
{8, 8, 8}	{16, 8, 4}	303	45.2	67.6	49.9	41.6	64.6	44.7
{8, 8, 8}	{32, 16, 8}	291	44.6	66.8	49.1	41.0	63.8	44.3
{16, 16, 8}	{16, 8, 4}	309	45.1	67.5	49.5	41.3	64.5	44.4
{4, 4, 4}	{16, 8, 4}	300	45.1	67.1	49.5	41.3	64.3	44.5

4.4 Ablation Studies

In this part, we evaluate various design choices of the proposed HAT-Net. As discussed above, we only train all ablation models for 100 epochs to save training time. The batch size and learning rate are also reduced by half accordingly. HAT-Net-Small is adopted for these ablation studies.

Effect of the proposed H-MHSA. Starting from the window attention [18] based transformer network, we gradually replace the window attention with our proposed H-MHSA at different stages. The results are summarized in Table 5. Since the feature map at the fifth stage is small enough for directly computing MHSA, the fifth stage is excluded from Table 5. Note that the first stage of HAT-Net only consists of convolutions so that it is also excluded. From Table 5, we can observe that the performance for both image classification and semantic segmentation is improved when more stages adopt H-MHSA. This verifies the effectiveness of the proposed H-MHSA in feature presentation learning. It is interesting to find that the usage of H-MHSA at the fourth stage leads to more significant improvement than other stages. Intuitively, the fourth stage has the most transformer blocks, so the changes at this stage would lead to more significant effects.

A pure transformer version of HAT-Net *vs.* PVT [19]. When we remove all depth-wise separable convolutions from HAT-Net and train the resulting transformer network for 100 epochs, it achieves 77.7% top-1 accuracy on the ImageNet validation set [24]. In contrast, the well-known transformer network, PVT [19], attains 75.8% top-1 accuracy under the same condition.

This suggests that our proposed H-MHSA is very effective in feature representation learning.

SiLU [59] *vs.* GELU [58]. We use SiLU function [59] for nonlinearization rather than the widely-used GELU function [58] in transformers [13, 16]. Here, we evaluate the effect of this choice. HAT-Net with SiLU [59] attains 82.6% top-1 accuracy on the ImageNet validation set [24] when trained for 300 epochs. HAT-Net with GELU [58] gets 82.7% top-1 accuracy, slightly higher than SiLU [59]. However, HAT-Net with GELU [58] only obtains 45.7% mIoU on the ADE20K dataset, 0.8% lower than HAT-Net with SiLU. When using a batch size of 128 per GPU, HAT-Net with SiLU [59] occupies 20.2GB GPU memory during training, while HAT-Net with GELU [58] occupies 23.8GB GPU memory. Hence, HAT-Net with SiLU [59] can achieve slightly better performance with less GPU memory consumption.

Settings of G_1 and G_2. In HAT-Net, the parameters G_1 and G_2 play pivotal roles, controlling grid sizes for local attention calculation and downsampling rates for global attention calculation, respectively. In this evaluation, we assess the model's performance under various configurations of G_1 and G_2. By default, for tasks such as object detection and instance segmentation, we employ $G_1 = 8, 8, 8$ and $G_2 = 16, 8, 4$ for the t-th stage ($t = 2, 3, 4$), respectively. Subsequently, we systematically vary G_1 and G_2, evaluating the performance of Mask R-CNN [5] with HAT-Net-Small as the backbone. The evaluation results, conducted on the MS-COCO `val2017` dataset [71], are presented in Table 6. The findings indicate that HAT-Net demonstrates robustness across different G_1 and G_2 settings. Notably, altering G_1 from its default $8, 8, 8$ configuration has a marginal impact on performance, resulting in slight performance reduction. Similarly, adjusting the values of G_2 yields a trade-off: decreasing values enhances performance at the expense of increased computational cost, while increasing values reduces computational cost at the cost of slightly degraded performance. Our default choice of $G_1 = 8, 8, 8$ and $G_2 = 16, 8, 4$ strikes a favorable balance between accuracy and efficiency, offering a practical configuration for general use.

5 Conclusion

This paper addresses the inefficiency inherent in vanilla vision transformers due to the elevated computational and space complexity associated with MHSA. In response to this challenge, we introduce a novel hierarchical framework for MHSA computation, denoted as H-MHSA, aiming to alleviate the computational and space demands. Compared to existing approaches in this domain, such as PVT [19] and Swin Transformer [18], H-MHSA distinguishes itself by directly capturing both global dependencies and local relationships. Integrating the proposed H-MHSA, we formulate the HAT-Net family, showcasing its prowess through comprehensive experiments spanning image classification, semantic segmentation, object detection, and instance segmentation. Our results affirm the efficacy and untapped potential of HAT-Net in advancing representation learning.

Applications of HAT-Net. The versatility of HAT-Net extends its utility across diverse real-world scenarios and downstream vision tasks. As a robust backbone network for feature extraction, HAT-Net seamlessly integrates with existing prediction heads and decoder networks, enabling proficient execution of various scene understanding tasks. Furthermore, HAT-Net's adaptability to different input resolutions and computational resource constraints is facilitated by the flexible adjustment of parameters, specifically G_1 and G_2. Users can tailor HAT-Net to their specific requirements, selecting from different HAT-Net versions to align with their objectives.

In conclusion, HAT-Net not only presents a pragmatic solution to the limitations of vanilla vision transformers but also opens avenues for innovation in the future design of such architectures. The simplicity of the proposed H-MHSA underscores its potential as a transformative element in the evolving landscape of vision transformer development.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inform. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. Learn. Represent.*, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 770–778.

[4] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2016.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[6] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2881–2890.

[7] Y. Liu, M.-M. Cheng, X. Hu, J.-W. Bian, L. Zhang, X. Bai, and J. Tang, "Richer convolutional features for edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1939–1946, 2019.

[8] Y. Liu, M.-M. Cheng, D.-P. Fan, L. Zhang, J.-W. Bian, and D. Tao, "Semantic edge detection with diverse deep supervision," *Int. J. Comput. Vis.*, vol. 130, no. 1, pp. 179–198, 2022.

[9] Y. Liu, M.-M. Cheng, X.-Y. Zhang, G.-Y. Nie, and M. Wang, "DNA: Deeply supervised nonlinear aggregation for salient object detection," *IEEE Trans. Cybernetics*, vol. 52, no. 7, pp. 6131–6142, 2021.

[10] Y. Liu, Y.-C. Gu, X.-Y. Zhang, W. Wang, and M.-M. Cheng, "Lightweight salient object detection via hierarchical visual perception learning," *IEEE Trans. Cybernetics*, vol. 51, no. 9, pp. 4439–4449, 2020.

[11] Y. Liu, Y.-H. Wu, Y. Ban, H. Wang, and M.-M. Cheng, "Rethinking computer-aided tuberculosis diagnosis," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 2646–2655.

[12] Y. Liu, Y.-H. Wu, P. Wen, Y. Shi, Y. Qiu, and M.-M. Cheng, "Leveraging instance-, image-and dataset-level information for weakly supervised instance segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 3, pp. 1415–1428, 2020.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Adv. Neural Inform. Process. Syst.*, 2017, pp. 6000–6010.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[15] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *ACL*, 2019, pp. 2978–2988.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Int. Conf. Learn. Represent.*, 2021.

[17] B. Heo, S. Yun, D. Han, S. Chun, J. Choe, and S. J. Oh, "Rethinking spatial dimensions of vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 11 936–11 945.

[18] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Int. Conf. Comput. Vis.*, 2021, pp. 10 012–10 022.

[19] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in

Int. Conf. Comput. Vis., 2021, pp. 568–578.

[20] W. Xu, Y. Xu, T. Chang, and Z. Tu, "Co-Scale conv-attentional image transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 9981–9990.

[21] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, "Multiscale vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 6824–6835.

[22] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, "Token merging: Your ViT but faster," in *Int. Conf. Learn. Represent.*, 2023.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[25] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2015, pp. 1–9.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 2818–2826.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 1492–1500.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 4700–4708.

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 4510–4520.

[33] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 6848–6856.

[34] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.

[35] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 2820–2828.

[36] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[37] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," in *Adv. Neural Inform. Process. Syst.*, 2015, pp. 2017–2025.

[38] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 5659–5667.

[39] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 3156–3164.

[40] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, 2020.

[41] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.

[42] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "BAM: Bottleneck attention module," in *Brit. Mach. Vis. Conf.*, 2018, p. 147.

[43] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 510–519.

[44] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7794–7803.

[45] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha *et al.*, "ResNeSt: Split-attention networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 2736–2746.

[46] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan, "Tokens-to-token ViT: Training vision transformers from scratch on ImageNet," in *Int. Conf. Comput. Vis.*, 2021, pp. 558–567.

[47] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 32–42.

[48] D. Zhou, B. Kang, X. Jin, L. Yang, X. Lian, Q. Hou, and J. Feng, "DeepViT: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886*, 2021.

[49] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Int. Conf. Mach. Learn.*, 2021, pp. 10 347–10 357.

[50] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, "Bottleneck transformers for visual recognition," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 16 519–16 529.

[51] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-Mixer: An all-MLP architecture for vision," *Adv. Neural Inform. Process. Syst.*, pp. 24 261–24 272, 2021.

[52] H. Liu, Z. Dai, D. So, and Q. V. Le, "Pay attention to MLPs," *Adv. Neural Inform. Process. Syst.*, pp. 9204–9215, 2021.

[53] Q. Hou, Z. Jiang, L. Yuan, M.-M. Cheng, S. Yan, and J. Feng, "Vision Permutator: A permutable MLP-like architecture for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 1328–1334, 2022.

[54] Z. Wang, Y. Hao, X. Gao, H. Zhang, S. Wang, T. Mu, and X. He, "Parameterization of cross-token relations with relative positional encoding for vision MLP," in *ACM Int. Conf. Multimedia*, 2022, pp. 6288–6299.

[55] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," in *Adv. Neural Inform. Process. Syst.*, 2021, pp. 15 908–15 919.

[56] Y. Li, K. Zhang, J. Cao, R. Timofte, and L. Van Gool, "LocalViT: Bringing locality to vision transformers," *arXiv preprint arXiv:2104.05707*, 2021.

[57] K. Yuan, S. Guo, Z. Liu, A. Zhou, F. Yu, and W. Wu, "Incorporating convolution designs into visual transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 579–588.

[58] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv preprint arXiv:1606.08415*, 2016.

[59] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.

[60] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Adv. Neural Inform. Process. Syst.*, 2019, pp. 8026–8037.

[61] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *Int. Conf. Learn. Represent.*, 2018.

[62] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Int. Conf. Learn. Represent.*, 2019.

[63] ——, "SGDR: Stochastic gradient descent with warm restarts," in *Int. Conf. Learn. Represent.*, 2017.

[64] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "PVTv2: Improved baselines with pyramid vision transformer," *Computational Visual Media*, vol. 8, no. 3, pp. 415–424, 2022.

[65] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 10 428–10 436.

[66] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang, "CvT: Introducing convolutions to vision transformers," in *Int. Conf. Comput. Vis.*, 2021, pp. 22–31.

[67] X. Chu, Z. Tian, Y. Wang, B. Zhang, H. Ren, X. Wei, H. Xia, and C. Shen, "Twins: Revisiting the design of spatial attention in vision transformers," in *Adv. Neural Inform. Process. Syst.*, 2021, pp. 9355–9366.

[68] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 633–641.

[69] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 6399–6408.

[70] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.

[71] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[72] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, "Multi-scale vision Longformer: A new vision transformer for high-resolution image encoding," in *Int. Conf. Comput. Vis.*, 2021, pp. 2998–3008.

[73] M. Contributors, "MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark," https://github.com/open-mmlab/mmsegmentation, 2020.

[74] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu *et al.*, "MMDetection: Open MMLab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.