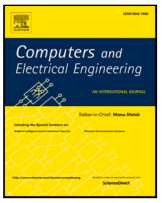
Computers and Electrical Engineering 108 (2023) 108698

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Mixture lightweight transformer for scene understanding☆

Quan Zhou [a,*], Zhenhan Sun [a], Linjie Wang [a], Bin Kang [b], Suofei Zhang [b], Xiaofu Wu [a]

[a] National Engineering Research Center of Communications and Networking, Nanjing University of Posts and Telecommunications, Nanjing, PR China

[b] Department of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing, PR China

ARTICLE INFO

Keywords:
Transformers
Lightweight backbone
Multi-scale pyramid pooling
Convolutional inception

ABSTRACT

In adapting Transformer from language to computer vision, the major obstacles are the high computational complexity and large model size of Transformer blocks, derived from the great quantity of visual tokens and high resolutions of input entities. To address these challenges, this paper presents a mixture lightweight Transformer (MLT) backbone for image understanding, where each Transformer block, called SH-Transformer, adopts Single-Head Self-Attention (SHSA) and Convolutional Inception Module (CIM). Unlike previous Transformers that compute Multi-Head Self-Attention (MHSA), SHSA limits the representation of input token into a single head, resulting in a low-dimensional embedding that greatly reduces computational complexity. In spite of adding a small number of model parameters, SHSA greatly minimizes the number of input tokens. As complementary of SHSA that only investigates global interactions, CIM is designed to explore multi-scale local information using lightweight convolutions in a multi-path parallel manner. Experimental results reveal that MLT yields competitive or state-of-the-art results with respect to recent transformers while keeping smaller model size and lower computational costs for different visual tasks, including image classification, semantic segmentation and object detection. Particularly, the proposed method has 4.2% improvement to the tiny version of Pyramid Vision Transformer on image classification of top-1 accuracy on ImageNet-1K.

1. Introduction

Inspired from the success in natural language processing (NLP) [1], Transformer-based backbones begin to show their potential in computer vision [2–5]. As a pioneer work, vision Transformer (ViT) [2] directly inherits Transformer from NLP for image classification. Since then, many researchers prefer to design pure transformer backbones for various dense prediction tasks [3,5,6]. However, due to the inductive bias that pure Transformers only consider global context, some CNN-transformer hybrids are proposed to inherit the merits of transformers and CNNs [7,8], where global and local features are investigated synchronously. Although these Transformer backbones have achieved remarkable progress in classification and downstream tasks, they are still inherently suffered from following limitations: **(1) Quadratic complexity of computing Multi-Head Self-Attention (MHSA).** The complexity of calculating MHSA is quadratic with respect to the resolution of input features [2]. When there are large number of input tokens embedded in a high-dimensional feature space, computing MHSA is extremely expensive. **(2) Huge model size of Feed Forward network (FFN).** The traditional FFN usually employs a multiple layer perception (MLP), where this fully-connected architecture, however, leads to huge number of parameters in each Transformer block, eventually slowing down the implementing efficiency of

☆ This paper is for regular issues of CAEE. Reviews processed and recommended for publication to the Editor-in-Chief by Huimin Lu.

* Corresponding author.
E-mail addresses: quan.zhou@njupt.edu.cn (Q. Zhou), 1020010522@njupt.edu.cn (Z. Sun).

https://doi.org/10.1016/j.compeleceng.2023.108698
Received 29 December 2022; Received in revised form 18 March 2023; Accepted 22 March 2023
Available online 8 April 2023

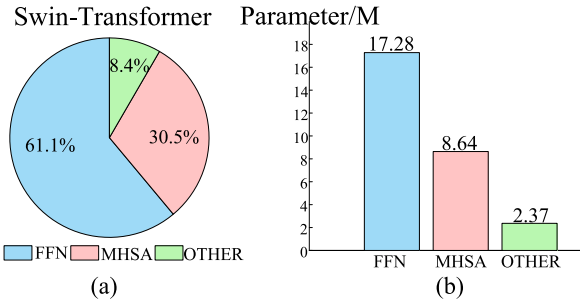


Fig. 1. Analysis of model size of tiny version of Swin-Transformer [3] backbone. (a) is the percentage proportion of different components, including MHSA, FFN, and others. (b) shows their absolute number of parameters.

entire backbone. Taking the tiny version of Swin-Transformer [3] as an example, Fig. 1 shows that model size of entire backbone has been dominate by FFN, achieving 61.1% percentage with 17.28M model parameters.

To address these limitations, this paper presents a mixture lightweight Transformer backbone, named mixture lightweight Transformer (MLT), served as an alternative to conventional Transformer-based backbone for many vision tasks, including image-level estimations as well as pixel-level dense predictions. More specifically, the core unit of our backbone is SH-Transformer block, which consists of two major components: Single-Head Self-Attention (SHSA) and Convolutional Inception Module (CIM). In contrast to previous transformers [2,3,5] that calculate self-attention in multi-head, our SHSA is designed to save computational costs from two perspectives: (1) compressing feature dimensions, and (2) shrinking token numbers. In the first aspect, we argue that the representation of an input token can be limited into a single head, resulting in a low-dimensional embedding that saves huge amount of computations in self-attention. In the second aspect, inspired from [3,5,9], we also adopt a token-to-region attention scheme, where a Pyramid Pooling module is employed to reduce the number of tokens.

On the other hand, inspired from Inception [10] that is widely adopted in CNN architecture [11–14], MLP is replaced by CIM, in SH-Transformer block to minimize model size, while maintaining powerful representation ability. Unlike earlier FFN [1,3] that involves huge amount of model parameters, CIM follows the design principle of Inception, adopting an multi-path structure to adaptively enlarge receptive fields using a series of depth-wise convolutions step-by-step, resulting in the lightweight and diverse transformations designed in each separated path. Furthermore, CIM inherits the advantage of local convolution, to some extend alleviating inductive bias that SHSA only consider global interactions.

In summary, the main contributions of this paper are four-fold:

(1) We design a lightweight hybrid Transformer that inherits the merits of transformers and CNNs, enabling MLT to be a fully lightweight versatile backbone for various vision tasks, including image classification, object detection, and semantic segmentation.

(2) Earlier Transformer backbones [2,3,5] calculate self-attention *independently* in each head of MHSA, resulting in no information interactions among different heads. In contrast, this is not required in our SHSA. Additionally, due to its elegant structure, SHSA remains powerful modeling capabilities, while greatly reducing computational complexity.

(3) Unlike traditional FFN [2,3] that involves large number of model parameters, CIM designs a lightweight local convolution network. The multi-path parallel architecture enables CIM a powerful representation ability to encode local features with different receptive fields, while keeping CIM lightweight synchronously.

(4) We have evaluated MLT on the tasks of image classification, object detection, and semantic segmentation. It only has half size of tiny-version of Swin-Transformer [3] with only 14.11M model parameters, yet yielding competitive or state-of-the-art results for image classification (79.3% top-1 accuracy on ImageNet-1K), semantic segmentation (77.4% and 44.3% mIoU on Cityscapes and ADE20K), and object detection (42.3% box AP and 38.6% mask AP on COCO test-dev).

The remainder of this paper is organized as follows. In Section 2, mainstream methods related to MLT are reviewed. In Section 3, the overall framework and detailed structure of MLT are described. And in Section 4, the effectiveness of MLT will be validated in different vision tasks and the results will be analyzed.

2. Related works

This section reviews recent backbones from three perspective: Attention based Full Precision Backbones, Lightweight Transformers and the Application of Transformers.

2.1. Self-Attention and transformer in vision

Both self-attention [15–17] and vision Transformers [2,18] are original from the success of NLP [1]. Before Transformers become popular in computer vision, self-attention is dominant to capture the global context in CNNs [15]. As the core idea of self-attention is to compute feature responses from all other positions, it can be considered to harvest global context of the entire scene.

Recently, vision Transformers [2–4] have started to show their potential for image classification [2,19], and downstream dense estimation tasks [6,20]. Different from self-attention layers that replace some convolution layers in the popular ResNet [19], vision Transformers construct backbones by directly stacking self-attention layers. For example, ViT [2] is the first work that applies a Transformer based on non-overlapping patches (known as tokens) for image classification. Due to the tendency of ViT to overfit on small datasets, DeiT [18] integrates a diverse set of effective training strategies to enhance the performance of ViT on smaller datasets, such as ImageNet-1K. Swin-Transformer [3] designs a pure Transformer backbone that captures hierarchical features using shifted windows. This paper also belongs to this category, yet we seek a lightweight Transformer backbone that achieves available trade-off in terms of recognition accuracy and running efficiency.

2.2. Lightweight transformer in vision

Some earlier attempts [4,9,21,22] also have a similar goals to our work. Those attempts are mainly divided into two directions, using lightweight convolution or pooling operation to compress the query, key and value tensors in Transformers.

The first category mainly uses depth-wise convolution for lightweight design. For example, CvT [22] utilizes depth-wise convolution to generate the query, key, and value tensors used for the computation of MHSA. ResT [4] builds a memory-efficient multi-head self-attention, which compresses the memory by a simple depth-wise convolution, and projects the interaction across the attention-heads dimension while keeping the diversity ability of multi-heads.

The second category contributes to building the pyramid structure for the vision Transformer using pooling operations [3–5,9]. Among them, PVT [5] and MViT [23] are the first two methods that adopt the single pooling operation to reduce the number of tokens in the calculation of MHSA. In this way, they conduct token-to-region instead of token-to-token modeling. But the features extracted by a single-scale pooling operation appear less powerful. In addition, the high computational complexity of MHSA and heavier FFN are not well addressed. Conversely, thanks to the proposed Single-Head Self-Attention (SHSA) and CIM, our Mixture Lightweight Transformer (MLT) shows its compatibility with a broad range of vision tasks, while significantly reducing computational complexity with very few model sizes.

The most related work to our method is P2T [9], yet there are several major differences between MLT and P2T [9]. In P2T [9], the pooling ratio of each layer has a small change so that the pooling regions do not strictly follow a hierarchical pyramid structure. Therefore, the pooling regions are spatially overlapped among different pooling levels, leading to great redundancy in hierarchical feature representation. In contrast, we carefully control the pooling ratio of SHSA so that there is no region overlapping in the produced hierarchy feature representation, resulting in a great token reduction compared with P2T [9]. More importantly, MLT adopts SHSA and CIM to save computational costs, which are different from the MHSA and FFN used in P2T [9].

2.3. Application of transformer in vision

Transformers have achieved great progress, such as image classification, object detection, semantic segmentation, position estimation, and image generation task. YOLOS [6] and SETR [20] build a pure Transformer backbone for object detection and semantic segmentation, respectively. Paul et al. [24] point out that a large percentage of attention heads can be removed at test time without significantly impacting the detection performance. Tim et al. [25] present a simpler and faster Transformer-based cell detection Transformer (Cell-DETR) for direct end-to-end instance segmentation. In the position estimation, Lin et al. [26] use a Transformer encoder to output 3D joint coordinates and mesh vertices without relying on any parametric mesh models. For image generation, Esser et al. [27] utilize Transformer to efficiently formulate their composition within high-resolution images. In addition, TransUNet [28] combines the merits of Transformers and UNet [29] for medical image segmentation.

3. Our method

3.1. Overall architecture

The overall architecture of mixture lightweight Transformer (MLT) is illustrated in Fig. 2. More specifically, MLT adopts a pyramid structure backbone similar with PVT [5], which has four stages that generate feature maps of different resolutions. Each stage consists of three components, one patch embedding module (or stem module) [4,5], one position encoding module [4], and a set of $D_i (i = \{1, 2, 3, 4\})$ SH-Transformer blocks, where i denotes the stage number.

The detailed settings of the proposed MLT are shown in Table 1. To produce a hierarchical representation, at the beginning of each stage, the patch embedding module(or stem module) [4,5] is adopted to reduce the number of tokens by embedding multiple low dimension tokens into high dimension ones. The position encoding module [4] is fused to restrain position information and strengthen the feature extracting ability of patch embedding. Thereafter, the embedded patches along with a position encoding are passed through the transformer encoder, producing the output features $\boldsymbol{F}_i \in \mathbb{R}^{d_i \times H_i \times W_i}$ in ith stage, where W_i, H_i, and d_i stand for width, height, and channel, respectively. For more details, please refer to Table 1.

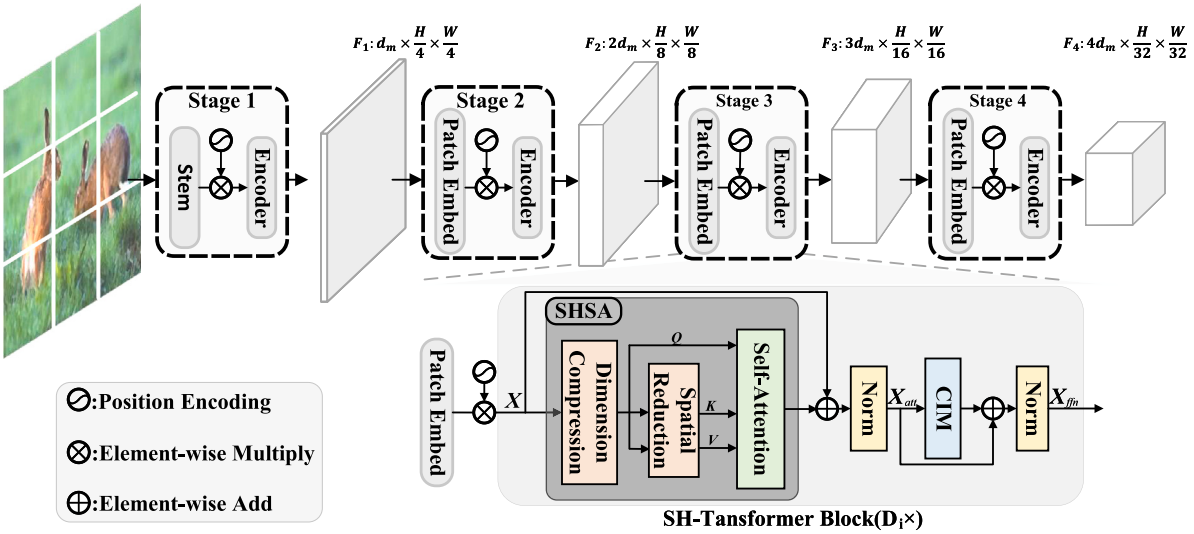


Fig. 2. Detail architecture of MLT and SH-Transformer block. The overall architecture of MLT and the specific structure inside one of the stages.

Table 1
Detailed settings of the proposed MLT. *Output Size* refers to the resolution of the output. *Operation* represents the operation in the ith stage, such as SH–Transformer Block. In addition, each SH–Transformer Block uses an compression ratio of s and the pooling ratios of p_j.

Stage	Output size	Operation	Parameter setting
1	$\boldsymbol{F}_1 : 128 \times 56 \times 56$	*Stem* *Position Encoding* $SH\text{–}TransformerBlock \times 2$	$s = 1$ $\{p_1 = 1, p_2 = 2,$ $p_3 = 4, p_4 = 8, p_5 = 16\}$
2	$\boldsymbol{F}_2 : 256 \times 28 \times 28$	*Patch Embedding* *Position Encoding* $SH\text{–}TransformerBlock \times 2$	$s = 2$ $\{p_1 = 1, p_2 = 2,$ $p_3 = 4, p_4 = 8, p_5 = 16\}$
3	$\boldsymbol{F}_3 : 512 \times 14 \times 14$	*Patch Embedding* *Position Encoding* $SH\text{–}TransformerBlock \times 6$	$s = 4$ $\{p_1 = 1, p_2 = 2, p_3 = 4, p_4 = 8\}$
4	$\boldsymbol{F}_4 : 1024 \times 7 \times 7$	*Patch Embedding* *Position Encoding* $SH\text{–}TransformerBlock \times 2$	$s = 8$ $\{p_1 = 1, p_2 = 2, p_3 = 4\}$

Algorithm 1 $\boldsymbol{X}_{ffn} \leftarrow SH\text{-}TransformerBlock(\boldsymbol{X})$

Input: $\boldsymbol{X} \in \mathbb{R}^{c \times h \times w}$, vector representations of the input sequence.
Output: $\boldsymbol{X}_{ffn} \in \mathbb{R}^{c \times h \times w}$, updated representations of tokens in X, vector representations of the output sequence.
Operation: $LayerNorm(\cdot)$ representations of the operation of layer normalization,
$Re(\cdot)$ representations of the operation of reshape;
$SHSA(\cdot)$ representations of the algorithm of ***S***ingle ***H***ead ***S***elf ***A***ttention;
$CIM(\cdot)$ representations of the algorithm of ***C***onvolutional ***I***nception ***M***odule;

function SH-TRANSFORMER BLOCK($\boldsymbol{X}$)
  $\boldsymbol{X}_{att} \leftarrow Re(LayerNorm(Re(\boldsymbol{X} + SHSA(\boldsymbol{X}))))$ $[\![\boldsymbol{X}_{att} \in \mathbb{R}^{c \times h \times w}]\!]$ \\ Eq.(1)
  $\boldsymbol{X}_{ffn} \leftarrow Re(LayerNorm(Re(\boldsymbol{X}_{att} + CIM(\boldsymbol{X}_{att}))))$ \\ Eq.(1)
  return $\boldsymbol{X}_{ffn}$
end function

3.2. SH-transformer block

In this section, we elaborate on the details of the core block SH-Transformer, as illustrated in Fig. 2, which consists of two major components: Single-Head Self-Attention (SHSA) and Convolutional Inception Module (CIM).

Let $\boldsymbol{X} \in \mathbb{R}^{c \times h \times w}$ and $\boldsymbol{X}_{att} \in \mathbb{R}^{c \times h \times w}$ be the input of the SHSA and CIM, where w, h, and c stand for width, height, and channel number, respectively. Unlike traditional Transformer block [2,3], CIM is proposed to replace the traditional Multilayer Perceptron

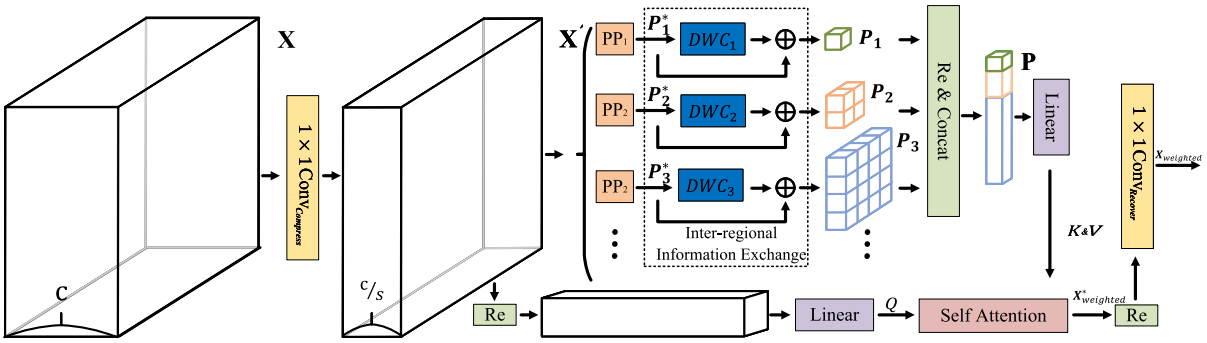


Fig. 3. The structure of SHSA. The illustration of SHSA which consists of two parts, an efficient pyramid pooling calculation method for fusing multi-scale information and self-attention. 1×1 $Conv_{Compress}(\cdot)$ and 1×1 $Conv_{Recover}(\cdot)$ represent two 1×1 standard convolutions for compression and recovery the input dimension, $PP_j(\cdot)$ denotes pyramid pooling operation in the jth level, $Re(\cdot)$ stands for reshape operation, $Concat(\cdot)$ represents concatenating operation, $DWC_j(\cdot)$ indicates the depth-wise convolution operation in the jth levels, and $Linear(\cdot)$ is the operation of linear projection.

(MLP) layer [1] in the Transformer encoder for feature projection. A residual connection and LayerNorm [30] are applied again to produce the output $\boldsymbol{X}_{ffn} \in \mathbb{R}^{c \times h \times w}$:

$$\begin{aligned} \boldsymbol{X}_{att} &= Re(LayerNorm(Re(\boldsymbol{X} + SHSA(\boldsymbol{X})))) \\ \boldsymbol{X}_{ffn} &= Re(LayerNorm(Re(\boldsymbol{X}_{att} + CIM(\boldsymbol{X}_{att})))) \end{aligned} \tag{1}$$

where Re is the reshape operation. Algorithm 1 summarizes the whole process of computing SH-Transformer Block.

3.2.1. SHSA

To save the huge computational costs, we propose an efficient SHSA module (illustrated in Fig. 3) that is mainly divided into three computational steps: dimension compression, token-to-region projection and self-attention computation. Immediately below, we introduce each component in detail.

The first procedure is the dimension compression. The input $\boldsymbol{X}$ first passes through an 1×1 convolution $Conv_{Compress}(\cdot)$, producing a low-dimensional embedding $\boldsymbol{X'} \in \mathbb{R}^{c/s \times h \times w}$, whose channel numbers is c/s, where s is compression ratio as shown in Table 1:

$$\boldsymbol{X'} = Conv_{Compress}(\boldsymbol{X}) \tag{2}$$

The second procedure is token-to-region projection that mainly contains the pyramid pooling operation (pyramid pooling module) and relative positional encoding operation (Inter-regional Information Exchange module). Let $PP_j(\cdot)$ be the operation of jth levels's pooling with the pooling ratio p_j as shown in Table 1. In order to implement token-to-region projection, we first apply multi-level pooling operation $PP_j(\cdot)$ on input features $\boldsymbol{X'}$, parallelly producing multi-level hierarchical representations $\boldsymbol{P}_j^* \in \mathbb{R}^{c/s \times p_j \times p_j}$, $p_j^2 \ll h \times w$. As a result, each element in $\boldsymbol{P}_j^*$ represents an image region that includes hw/p_j^2 pixels in $\boldsymbol{X'}$. Let $\boldsymbol{P}_j^*$, and $\boldsymbol{P}_j \in \mathbb{R}^{c/s \times p_j \times p_j}$ be the input and output of Inter-regional Information Exchange (IIE) module, respectively. $\boldsymbol{P}_j^*$ is first fed into a 3×3 depth-wise convolution $DWC_j(\cdot)$ to encode the positional information. Then the positional encoding serves as a residual function that is added with the $\boldsymbol{P}_j^*$:

$$\begin{aligned} \boldsymbol{P}_j^* &= PP_j(\boldsymbol{X'}) \\ \boldsymbol{P}_j &= DWC_j(\boldsymbol{P}_j^*) + \boldsymbol{P}_j^* \end{aligned} \quad j = 1, 2, \ldots, L (3 \le L \le 5) \tag{3}$$

The output of token-to-region projection needs to be flattened and concatenated, facilitating computations of forthcoming self-attention. Let the $Concat(\cdot)$, $LayerNorm(\cdot)$ and $Re(\cdot)$ be the concatenation, layer normalization and reshape operation, respectively. We collect all pyramid pooling representation $\boldsymbol{P}_j$ together, producing the stacked features $\boldsymbol{P} \in \mathbb{R}^{m \times c/s}$ ($m = \sum_j p_j^2$, $m \ll h \times w$) that is the output of the whole token-to-region projection:

$$\boldsymbol{P} = LayerNorm\left(Concat\left(Re(\boldsymbol{P}_1), \ldots, Re(\boldsymbol{P}_j)\right)\right) \tag{4}$$

The final procedure is computing self-attention. Note that the query, key, and value tensors are not split into multiple heads for calculation. Suppose the query, key, and value tensors are $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$, respectively. To obtain the $\boldsymbol{Q} \in \mathbb{R}^{n \times c/s}$ ($n = h \times w$, $c/s \le c$), SHSA maps the reshaped $\boldsymbol{X'}$ to the query tensor $\boldsymbol{Q}$ by adopting the weight matrix $\boldsymbol{W^q}$. On the other hand, the key $\boldsymbol{K} \in \mathbb{R}^{m \times c/s}$ and value $\boldsymbol{V} \in \mathbb{R}^{m \times c/s}$ tensors ($c/s \le c, m \ll n$) are instead derived from features $\boldsymbol{P}$ through the other two weight matrices $\boldsymbol{W^k}$ and $\boldsymbol{W^v}$, respectively:

$$(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (Re(\boldsymbol{X'})\boldsymbol{W^q} + \boldsymbol{b}_q\boldsymbol{1}^\top, \boldsymbol{P}\boldsymbol{W^k} + \boldsymbol{b}_k\boldsymbol{1}^\top, \boldsymbol{P}\boldsymbol{W^v} + \boldsymbol{b}_v\boldsymbol{1}^\top) \tag{5}$$

Then, $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ tensors are fed into the attention module to compute the attention tensor $\boldsymbol{X}_{weighted}^* \in \mathbb{R}^{n \times c/s}$:

$$\boldsymbol{X}_{weighted}^* = Softmax\left(\frac{\boldsymbol{Q} \cdot \boldsymbol{K}^\top}{\sqrt{c/s}}\right) \times \boldsymbol{V} \tag{6}$$

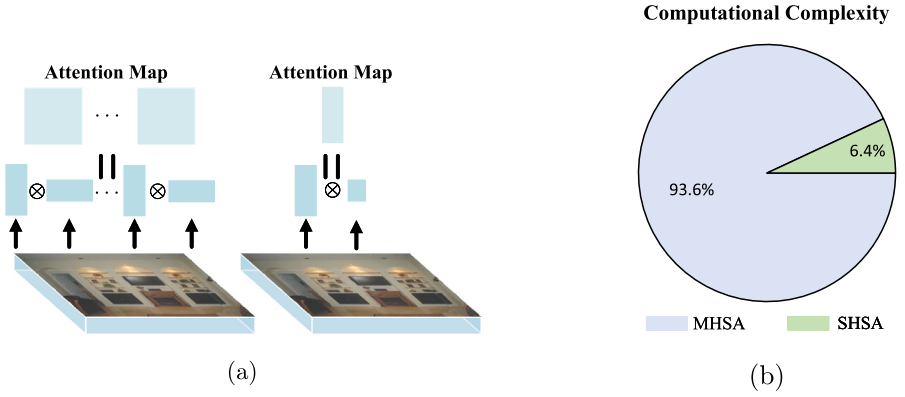


Fig. 4. Attention calculation analysis. Figure (a) intuitively shows the difference in computational method between single-head and multi-head. Figure (b) compares the relative size of computational complexity of the third stage's SH-Transformer Block with the latest proposed Transformer-based backbone ViT(MHSA) [2]. Because MLT's Transformer block in the third stage repeats the most times.

Finally, the reshape operation and 1×1 convolution named $Conv_{Recover}(\cdot)$ are used to recover the same resolutions and dimensions of the input $\boldsymbol{X}$, producing the weighted output features $\boldsymbol{X}_{weighted} \in \mathbb{R}^{c \times h \times w}$.

$$\boldsymbol{X}_{weighted} = Conv_{Recover}(Re(\boldsymbol{X}^{*}_{weighted})) \tag{7}$$

It can be seen from Fig. 4(a) that SHSA is quite different from traditional MHSA. Since $\boldsymbol{K}$ and $\boldsymbol{V}$ have less token numbers and feature channels than $\boldsymbol{X}$, and $\boldsymbol{Q}$ also has less feature channels compared with $\boldsymbol{X}$, SHSA produces only one attention map. In contrast, due to multiple head, traditional MHSA generates multiple attention maps, where each one has more elements than attention map calculated from SHSA that directly yields great saving of computational costs. As shown in Fig. 4(b), The relative computational complexity of SHSA is much smaller than that of MHSA. On the other hand, since $\boldsymbol{K}$ and $\boldsymbol{V}$ contains multi-scale information, the proposed SHSA has a stronger capability in global contextual dependency modeling, which is helpful for scene understanding. Algorithm 2 shows the entire computing process of SHSA.

Algorithm 2 $\boldsymbol{X}_{weighted} \leftarrow$ **S**ingle **H**ead **S**elf **A**ttention$(X, \mathcal{W}_{qkv})$

Input: $\boldsymbol{X} \in \mathbb{R}^{c \times h \times w}$, vector representations of the input sequence.
Output: $\boldsymbol{X}_{weighted} \in \mathbb{R}^{c \times h \times w}$, updated representations of tokens in $\boldsymbol{X}$, vector representations of the output sequence.
Operation: $LayerNorm(\cdot)$ representations of the operation of layer normalization,
$Re(\cdot)$ representations of the operation of reshape;
$PP_j(\cdot)$ representations of the operation of the j-th level pyramid pooling whith the pooling ratios p_j;
$Conv_{Compress}(\cdot)$ representations of the compression operation of 1×1 convolution;
$Conv_{Recover}(\cdot)$ representations of the recover operation of 1×1 convolution;
$DWC(\cdot)$ representations of the operation of Depth-wise convolution.
Parameters: $\mathcal{W}_{qkv}$ consisting of:
$\boldsymbol{W}_q \in \mathbb{R}^{n \times n}$ $(n = h \times w)$, $\boldsymbol{b_q} \in \mathbb{R}^n$
$\boldsymbol{W}_k \in \mathbb{R}^{m \times m}$ $(m = \sum_j p_j^2)$, $\boldsymbol{b_k} \in \mathbb{R}^m$
$\boldsymbol{W}_v \in \mathbb{R}^{m \times m}$, $\boldsymbol{b_v} \in \mathbb{R}^m$

function **S**INGLE **H**EAD **S**ELF **A**TTENTION$(X, \mathcal{W}_{qkv})$
  $\boldsymbol{X}' \leftarrow Conv_{Compress}(X)$ \\ Eq.(2)
  $\boldsymbol{Q} \leftarrow \boldsymbol{W}_q Re(\boldsymbol{X}') + \boldsymbol{b}_q \boldsymbol{1}^{\top}$ $[\![\boldsymbol{Query} \in \mathbb{R}^{n \times c/s}]\!]$ \\ Eq.(5)
  $\boldsymbol{P}_j \leftarrow DWC_j(PP_j(\boldsymbol{X}')) + PP_j(\boldsymbol{X}')$ $[\![\boldsymbol{P}_j \in \mathbb{R}^{c/s \times p_j \times p_j}]\!]$ \\ Eq.(3)
  $\boldsymbol{P} \leftarrow LayerNorm(Concat(\boldsymbol{P}_1; \boldsymbol{P}_2; \cdots; \boldsymbol{P}_L))$ $[\![j = 1, 2, \cdots, L (3 \le L \le 5), \boldsymbol{P} \in \mathbb{R}^{m \times c/s}]\!]$ \\ Eq.(4)
  $\boldsymbol{K} \leftarrow \boldsymbol{W}_k \boldsymbol{P} + \boldsymbol{b}_k \boldsymbol{1}^{\top}$ $[\![\boldsymbol{Key} \in \mathbb{R}^{m \times c/s}]\!]$ \ Eq.(5)
  $\boldsymbol{V} \leftarrow \boldsymbol{W}_q \boldsymbol{P} + \boldsymbol{b}_v \boldsymbol{1}^{\top}$ $[\![\boldsymbol{Value} \in \mathbb{R}^{m \times c/s}]\!]$ \ Eq.(5)
  $\boldsymbol{X}^{*}_{weighted} \leftarrow Softmax\left(\boldsymbol{Q} \cdot \boldsymbol{K}^{\top} / \sqrt{c/s}\right) \times \boldsymbol{V}$ $[\![\boldsymbol{X}^{*}_{weighted} \in \mathbb{R}^{n \times c/s}]\!]$ \\ Eq.(6)
  $\boldsymbol{X}_{weighted} \leftarrow Conv_{Recover}(Re(\boldsymbol{X}^{*}_{weighted}))$ $[\![\boldsymbol{X}_{weighted} \in \mathbb{R}^{c \times h \times w}]\!]$ \\ Eq.(7)
  return $\boldsymbol{X}_{weighted}$
end function

Table 2
The comparison of required operations. The comparison of required operations using MHSA [2], W-MHSA [22], and our SHSA, when attention is computed. n is the number of tokens, M is the number of shift windows in Swin-T [3], c is feature dimension of the input tokens, m is the number of output tokens after pyramid pooling, and s is the compression ratio.

Method	Computational complexity
MHSA (ViT) [2]	$O(4nc^2 + 2n^2c)$
W-MHSA (Swin-T) [3]	$O(4nc^2 + 2M^2nc)$
SHSA	$O\left(2nc^2/s + (n+2\ m)(c/s)^2 + (2\ nm)c/s\right)$

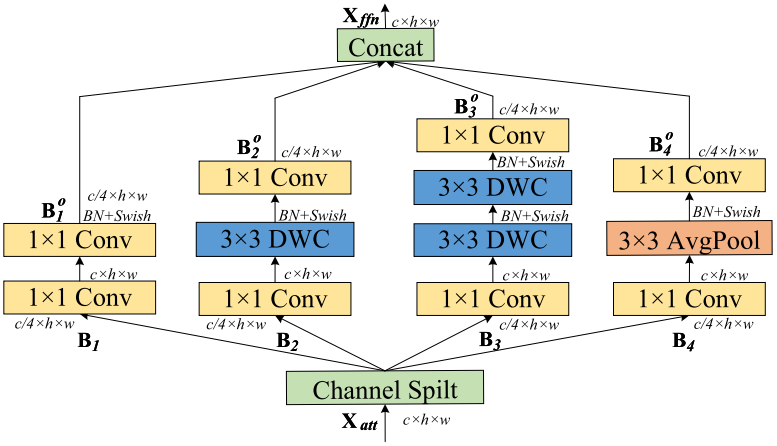


Fig. 5. Convolutional Inception Module (CIM). $AvgPool$ represents the standard average pooling. And $Conv$ stands for standard convolution, while DWC denotes depth-wise convolution.

Computational Complexity Analysis. We also analyze the computational complexity of proposed SHSA, and compare it with recent vision transformer networks [2,3] in Table 2, as they all have powerful representation capability. Previous methods [2,3] and MLT both involve two computational steps: feature embedding and attention calculating.

In the MHSA (ViT) [2], feature embedding which contains four fully-connected layers to embed the feature before and after the attention calculation demands $4nc^2$ operations and attention calculating requires $2n^2c$ operations, leading to a quadratic complexity of input spatial and channel dimensions. To alleviate the quadratic computation of MHSA (ViT) [2], W-MHSA (Swin-T) [3] proposes a method which is same as MHSA(ViT) [2] in feature embedding and attention calculating requires $2M^2nc$ operations, leading to a linear complexity of input spatial dimensions.

The computational complexity of SHSA is primarily comprised of three components: two 1×1 convolutions, three fully-connected layers, and two matrix multiplications. From Fig. 3, the standard 1×1 convolutions have different usages, including compressing dimensions with $Conv_{Compress}(\cdot)$ and recovering dimensions with $Conv_{Recover}(\cdot)$. Since the computational costs of two convolutions are the same, we select the $Conv_{Compress}(\cdot)$ as an example to analyze its computational complexity. Let $\boldsymbol{X} \in \mathbb{R}^{c \times h \times w}$ be the input feature map, where h is height, w is width, c is the number of channels, and $\boldsymbol{X'} \in \mathbb{R}^{c/s \times h \times w}$ be the output feature map, where c/s is the number of filters. Thus, the computational costs of the $Conv_{Compress}(\cdot)$ are $O\left(nc^2/s\right)$, where $n = h \times w$. Similarly, the input and output feature maps of the $Conv_{Recover}(\cdot)$ are $Re(\boldsymbol{X}^*_{weighted}) \in \mathbb{R}^{c/s \times h \times w}$ and $\boldsymbol{X}_{weighted} \in \mathbb{R}^{c \times h \times w}$, respectively. This indicates that its computational costs are $O\left(nc^2/s\right)$. Therefore, the computational complexity of both convolutions is equal to $O\left(2nc^2/s\right)$.

For the second part, let $Re(\boldsymbol{X'}) \in \mathbb{R}^{n \times c/s}$ be the input of the first fully-connected layer to produce $\boldsymbol{Q} \in \mathbb{R}^{n \times c/s}$, $\boldsymbol{P} \in \mathbb{R}^{m \times c/s}$ be the input of the remaining two fully-connected layers, and $\boldsymbol{K} \in \mathbb{R}^{m \times c/s}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times c/s}$ are key and value tensors of the remaining two fully-connected layers, respectively. Therefore, generating $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ require $O\left((n+2\ m)(c/s)^2\right)$ operations. Finally, the two matrix multiplications are performed between the matrices $\boldsymbol{QK}^\top$, and $\boldsymbol{QK}^\top\boldsymbol{V}$, respectively. Taking $\boldsymbol{QK}^\top$ as an example, as $\boldsymbol{Q} \in \mathbb{R}^{n \times c/s}$ and $\boldsymbol{K}^\top \in \mathbb{R}^{c/s \times m}$, computing $\boldsymbol{Q} \cdot \boldsymbol{K}^\top$ requires $O((nm)c/s)$ operations. On the other hand, as $\boldsymbol{QK}^\top \in \mathbb{R}^{n \times m}$ and $\boldsymbol{V} \in \mathbb{R}^{m \times c/s}$, the computational costs of multiplying $\boldsymbol{QK}^\top$ and $\boldsymbol{V}$ is $O((nm)c/s)$. Therefore the computational complexity of two matrix multiplications is $O((2\ nm)c/s)$. Finally, the total computational costs are:

$$O\left(2nc^2/s + (n+2\ m)(c/s)^2 + (2\ nm)c/s\right) \tag{8}$$

3.2.2. CIM

The detailed architecture of CIM is shown in Fig. 5, following the split-transform-merge architecture used in Inception [10]. At the beginning of each CIM, the input $\boldsymbol{X}_{att} \in \mathbb{R}^{c \times h \times w}$ is first split into four low-dimensional parts, where each one has the quarter channels of $\boldsymbol{X}_{att}$:

$$\{\boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3, \boldsymbol{B}_4\} = channel\_split(\boldsymbol{X}_{att}) \tag{9}$$

Algorithm 3 $\boldsymbol{X}_{Concat} \leftarrow$ **C**onvolutional **I**nception **M**odule ($\boldsymbol{X}_{att}$)

Input: $\boldsymbol{X}_{att} \in \mathbb{R}^{c\times h\times w}$, vector representations of the input sequence.
Output: $\boldsymbol{X}_{Concat} \in \mathbb{R}^{c\times h\times w}$, updated representations of tokens in $\boldsymbol{X}_{att}$, vector representations of the output sequence.
Operation: $LayerNorm(\cdot)$ representations of the operation of layer normalization,
$Re(\cdot)$ representations of the operation of reshape;
$Conv_{Upsample}(\cdot)$ representations of the compression operation of 1×1 convolution;
$Conv_{Recover}(\cdot)$ representations of the recover operation of 1×1 convolution;
$DWC(\cdot)$ representations of the operation of 3×3 Depth-wise convolution.

function CONVOLUTIONAL INCEPTION MODULE ($\boldsymbol{X}_{att}$)
  $[\boldsymbol{B}^1;\boldsymbol{B}^2;\boldsymbol{B}^3;\boldsymbol{B}^4] \leftarrow channel\_split(\boldsymbol{X}_{att})$ $[\![\boldsymbol{B}_k \in \mathbb{R}^{c/4\times h\times w}(k=1,\cdots,4)]\!]$ \\ Eq.(9)
  $\boldsymbol{B}_o^1 \leftarrow Conv_{Recover}(Conv_{Upsample}(\boldsymbol{B}^1))$ $[\![\boldsymbol{B}_o^1 \in \mathbb{R}^{c/4\times h\times w}]\!]$
  $\boldsymbol{B}_o^2 \leftarrow Conv_{Recover}(DWC(Conv_{Upsample}(\boldsymbol{B}^2)))$ $[\![\boldsymbol{B}_o^2 \in \mathbb{R}^{c/4\times h\times w}]\!]$
  $\boldsymbol{B}_o^3 \leftarrow Conv_{Recover}(DWC(DWC(Conv_{Upsample}(\boldsymbol{B}^3))))$ $[\![\boldsymbol{B}_o^3 \in \mathbb{R}^{c/4\times h\times w}]\!]$
  $\boldsymbol{B}_o^4 \leftarrow Conv_{Recover}(Avg\_Pooling(Conv_{Upsample}(\boldsymbol{B}^4)))$ $[\![\boldsymbol{B}_o^4 \in \mathbb{R}^{c/4\times h\times w}]\!]$
  $\boldsymbol{X}_{Concat} \leftarrow Concat(\boldsymbol{B}_o^1,\boldsymbol{B}_o^2,\boldsymbol{B}_o^3,\boldsymbol{B}_o^4)$ \\ Eq.(10)
  return $\boldsymbol{X}_{Concat}$
end function

where $\boldsymbol{B}_k \in \mathbb{R}^{c/4\times h\times w}(k=1,\ldots,4)$ is the input of each branch of the CIM.

According to the traditional FFN [9,18], each branch adopts the idea of inverted residual bottleneck [9,18,31]. Specifically, the input feature dimension of each branch is first expanded to four times of the original dimension, and then compressed to the input feature dimension after feature extraction. In order to extract features with different receptive fields, a series of convolution and pooling operations are applied to each branch with similar structure. And we define the output of the kth branch as $\boldsymbol{B}_k^o \in \mathbb{R}^{c/4\times h\times w}$ $(k=1,\ldots,4)$.

In the case of third branch, it first applies 1×1 convolution to project the dimension from $c/4$ to c. After that, two 3×3 depth-wise convolutions are stacked for extracting features in the receptive field of 5×5. Finally, an 1×1 convolution is utilized to reduce the dimension to the quarter of $\boldsymbol{X}_{att}$ for the final concatenating operation. As for the second and the fourth branch, two depth-wise convolutions are replaced by one depth-wise convolution and an average pooling, respectively. Besides, the first branch is only composed of two 1×1 convolutions to extract the feature information under the 1×1 receptive field.

Finally, a concatenation operation is applied to acquire the final output $\boldsymbol{X}_{Concat} \in \mathbb{R}^{c\times h\times w}$:

$$\boldsymbol{X}_{Concat} = Concat(\boldsymbol{B}_1^o,\boldsymbol{B}_2^o,\boldsymbol{B}_3^o,\boldsymbol{B}_4^o) \tag{10}$$

The usage of MLP as FFN in previous Transformers is weak in learning 2D spatial information, and brings huge number of parameters at the same time. In contrast, using depth-wise convolutions and average pooling strengthens the ability of 2D modeling, and reduces model size occupied by FFN. Algorithm 3 summarizes the computing process of CIM.

4. Results and discussion

The experiments are conducted on various commonly-used benchmarks, including ImageNet-1K [32] for image classification, COCO [33] for object detection and instance segmentation, and ADE20K [34] and Cityscapes [35] for semantic segmentation. In the following, we compare our mixture lightweight Transformer (MLT) with previous state-of-the-art Transformers for above tasks. In addition, we carry on a series of ablation studies and hyper-parameters setting experiments to uncover the underlying impact of various components of our method on the performance. Experimental results show that our MLT achieves very few model size, yet yields competitive performance.

4.1. Image classification

Experimental setup. ImageNet-1K [32] contains 1.28M and 50K training and validation images, respectively, which are divided into 1000 classes. For fair comparison, all models are trained on the training set, and the results are reported in terms of top-1 error on validation set. The initial learning rate is set to 1×10^{-4}, together with a momentum of 0.9 and a weight decay of 5×10^{-2}. Following [3,5], the cosine schedule learning scheme is employed to train MLT using AdamW optimizer [36] for 200 epochs with 20 epochs linear warm-up and the cross-entropy [37] loss function is used. As shown in Fig. 6(a) that our method can converge effectively on the ImageNet-1K [32] dataset. And the size of input sequences of MLT is $224\times 224\times 3$ which is the same input size as the traditional and most advanced Transformer backbone [3,5] to obtain a fair comparison result.

Results. As reported in Table 3, we observe that MLT achieves available trade-off in terms of top-1 accuracy, model size and GFlops. Although there are 2.0% and 0.5% performance drop compared with the tiny version of Swin-T [3] and DeiT-S [18], MLT only has nearly half size with respect to [3,18]. Table 3 shows that MLT outperforms other state-of-the-art backbones, including traditional CNN-based backbones, e.g., ResNet [38], and recent Transformer-based backbones, e.g, PVT-T [5] and ResT-Lite [4] by a large margin (4.2% and 2.1%, respectively). Particularly, MLT suppresses light version of ResNet by nearly 10%, but with

Table 3
Comparison with state-of-the-art backbones on ImageNet-1k benchmark. The number of GFlops is reported with the input size of 224 × 224. "#param." refers to the number of parameters. The bold number indicates the best performance among all approaches.

Method	#param. (M)	FLOPs (G)	top-1 (%)	top-5 (%)
ConvNet				
ResNet-18 [38]	11.7	1.8	69.7	89.1
ResNet-50 [38]	25.6	4.1	**79.0**	94.4
Transformer				
DeiT-Tiny/16 [18]	**5.7**	**1.3**	72.2	–
DeiT-Small/16 [18]	22.1	4.6	79.8	94.9
PVT-Tiny [5]	13.2	1.9	75.1	92.4
TNT-S [39]	23.8	5.2	81.3	–
Swin-Tiny [3]	28.29	4.5	81.3	**95.5**
ResT-Lite [4]	10.49	1.4	77.2	93.7
ViL-Tiny-RPB [40]	6.7	1.3	76.7	–
T2T-ViT-12 [21]	6.9	–	76.5	–
MLT	14.11	2.9	79.3	95.0

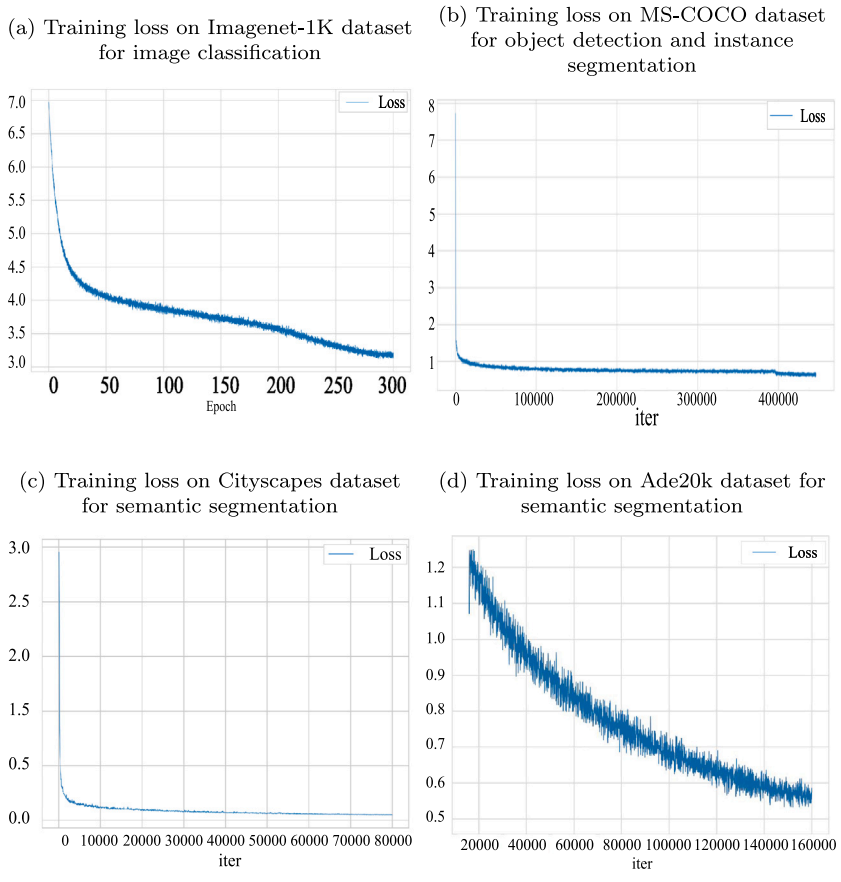


Fig. 6. Training loss on different datasets using our method. Note the horizontal coordinate of (a) is the epoch, and the rest are iteration numbers.

similar number of model parameters. In the aspect of efficiency, Table 3 also shows that the GFlops and model size of MLT lower than other state-of-the-art backbones, including traditional CNN-based backbones, e.g., ResNet [38] (2.1 gflops and 14.19 M lower, respectively), and recent Transformer-based backbones, e.g., TNT-S [39] and Swin-Tiny [3] (2.1 gflops, 9.69 M and 1.4 gflops, 14.18M lower, respectively). Such improvement mainly stems from the design of SHSA and CIM in our SH-Transformer block.

Table 4
Comparison with state-of-the-art networks on COCO dataset for Object detection and instance segmentation. AP^b and AP^m stand for bounding box AP and mask AP. The bold number indicates the best performance among all approaches.

Backbone	#param. (M)	Mask R-CNN [41]					
		AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
R-18 [38]	**31.2**	34.0	54.0	36.7	31.2	51.0	32.7
R-50 [38]	44.2	38.0	58.6	41.4	34.4	55.1	36.7
PVT-T [5]	32.9	36.7	59.2	39.3	35.1	56.7	37.3
ResT-S [4]	33.3	39.6	62.9	42.3	37.2	59.8	39.7
Swin-T [3]	47.8	**42.7**	**65.2**	**46.8**	**39.3**	**62.2**	**42.2**
MLT	33.5	42.3	64.0	45.7	38.6	60.7	40.7

4.2. Object detection and instance segmentation

Experimental setup. MS-COCO dataset [33] is a large-scale challenging dataset for object detection and instance segmentation. Similar to [3–5], the dataset is divided into 118k/5K/20K images for training, validation, and testing, respectively. All selected baselines and MLT are trained on training set and evaluated on validation set. We evaluate the effectiveness of MLT backbones, pre-trained on ImageNet-1K [32], on top of standard detector: Mask R-CNN [41], and the results are reported in terms of box AP on object detection task and mask AP on instance segmentation task, respectively. More specifically, AP is averagely evaluated at IoU (intersection area over the union area between the predicted bounding boxes and ground-truth bounding boxes), ranged from 0.5 to 0.95 with updated step 0.05, reflecting the comprehensive performance of the detector. And AP_{50} and AP_{75} are computed when IoU is 0.5 or 0.75, respectively. The software code is based on an open source repository for object detection using MMdetection framework [42]. Following [43], a multi-scale training scheme is adopted, where an input image is resized to have a shorter side of 800 pixels, while the longer side does not exceed 1333 pixels. Except the batch size is set to 8, we employ the same hyper-parameter settings used in image classification in a 1× training schedule (12 epochs). There are two losses used in detection task: cross-entropy [37] loss for object classification and segmentation, and L1 [44] loss for bounding box regression. As shown in Fig. 6(b) that our method can converge effectively on the MS-COCO dataset [33].

Results. The quantitative comparisons are report in Table 4. For fair comparison, the comparisons are performed by only changing backbones, yet remaining other settings unchanged. As shown in Table 4, except to Swin-T [3], MLT has comparable number of model size, yet significantly surpasses selected models. For example, the AP^b and AP^m of our method is 10.5 and 7.4 points higher than ResNet-18 [38]. On the other hand, although our MLT deliver slightly AP^b drop (0.4% and 0.5) with respect to Swin-T [3], MLT has fewer model size (33.5M vs. 47.8M), indicating that MLT can be a valid lightweight alternative backbone for object detection.

4.3. Semantic segmentation

Experimental setup. ADE20K [34] covers a broad range of 150 semantic categories. It has 25K images in total, with 20K/2K/3K for training, validation, and testing. Cityscapes [35] consists of 5000 high-quality images of 2048 × 1024 resolution,with 2975/500/1525 for training, validation, and testing. We evaluate MLT backbones on the basis of UperNet [45] and semantic FPN [46] using MMseg [47] toolbox, and report results in terms of mIoU. The baseline backbones are pre-trained using ImageNet-1K or ImageNet-21K, and other layers are initialized using the Xavier [48]. Following [15], the poly learning schedule is adopted in this setting, where initial learning rate, weight decay, and γ are set as 5×10^{-5}, 10^{-2}, and 9×10^{-1}, respectively. We train our method using AdamW optimizer [36] in 80K and 160K iterations for Cityscapes and ADE20K datasets, together with 1500 iterations of linear warm-up and the cross-entropy [37] loss function is used. As shown in Fig. 6(c) and (d) that our method can converge effectively on the ADE20K [34] and Cityscapes [35] dataset.

Results on ADE20K and Cityscapes. Table 5 shows the comparison results on the validation set of ADE20K and Cityscapes dataset, respectively. In Table 5, our MLT backbone achieves competitive results with Swin-T [3] and DeiT-S [18], but has fewer number of parameters (47.6M vs. 60.0M and 47.6M vs. 52.0M). We also stress that, in spite of only using ImageNet-1k training set, MIL surpasses some state-of-the-arts, such as ResNet-50 [38] and DeiT-S [18], which train their models using the more annotated data, e.g., ImageNet-21k. From Table 5, it can be seem that MLT outperforms PVT-Tiny [5] by a large margin (5.7% mIoU), yet only slightly increase 3M model size (20.1 vs. 17.0).

4.4. Ablation study

Ablation Study for Components of Backbone. The experimental setup for the ablation study for components of backbone remains the same as that trained on the ImageNet-1k dataset in 4.1. Table 6(a) presents ablation studies that quantify the contributions of different components in backbone, where MHSA and MLP is firstly used to build up our baseline, then Single-Head Self-Attention (SHSA) and Convolutional Inception Module (CIM) replace the MHSA and MLP step-by-step. This experiment shows that each of these components consistently reduces the size of the model. Among all components, it is observed that SHSA brings significantly improvements that the number of parameters is 13.92M less than the baseline together with slight decrease of

Table 5
Comparison with state-of-the-art backbones on Cityscapes and ADE20K validation set. The bold number indicates the best performance among all approaches.

Backbone	ADE20K Method	pre	mIoU (%)	#param. (M)	Cityscapes Method	pre	mIoU (%)	#param. (M)
ConvNet								
ResNet-101 [38]	UperNet [45]	21k	44.9	86.0	DANet [15]	21k	**77.6**	69.0
ResNet-50 [38]	UperNet [45]	21k	40.7	66.5	Semantic FPN [46]	1k	74.5	25.8
Transformer								
DeiT-S [18]	UperNet [45]	21k	44.0	52.0	–	–	–	–
Swin-T [3]	UperNet [45]	1k	**46.1**	60.0	–	–	–	–
PVT-Tiny [5]	–	–	–	–	Semantic FPN [46]	1k	71.7	**17.0**
MLT	UperNet [45]	1k	44.3	**47.6**	Semantic FPN [46]	1k	77.4	20.1

Table 6
Experimental results of ablation study. Ablation experimental results on the validation set of the ImageNet-1K [32] dataset and Cityscapes [35] dataset. MLP is the traditional feed forward network with the expansion coefficient of 4.

(a)

ImageNet-1K				top-1	#params.
MHSA	MLP	SHSA	CIM	(%)	(M)
✓	✓	×	×	73.3	50.0
×	✓	✓	×	71.2 (↓ 2.1)	36.1 (↓ 13.9)
×	×	✓	✓	**79.3** (↑ 6.0)	**14.0** (↓ 36.0)

(b)

Cityscapes	mIoU	#params.
IIE	(%)	(M)
×	51.4	40.5
✓	**57.8** (↑ 6.4)	**42.8** (↑ 2.3)

classification performance, demonstrating the advantage of SHSA architecture. In addition, CIM outperforms the counterpart MLP by 7% top-1 and the model size is reduced to 14.03M at the same time.

Ablation Study for Inter-regional Information Exchange (IIE). The experimental setup for the ablation study for IIE remains the same as that trained on the Cityscapes dataset in 4.3, with the only difference being that the number of iterations is reduced to 40,000. The IIE is used to capture the position information. And the segmentation task is more sensitive to position information and easier to compare gaps. Therefore, this section evaluates the effect of introduced by fixed backbone on Cityscapes [35] dataset with the semantic FPN decoder head [46]. Baselines were constructed using a combination of SHSA and MLP, except for the omission of IIE from all SHSA modules. The comparison results are reported in Table 6(b). Compared with baseline, a slight increase of model size demonstrates that IIE is a lightweight and efficient module, yet it obtains significant improvement of 6.39% mIoU.

4.5. Analysis of parameter setting

This section evaluates SHSA under various settings of pyramid pooling ratios and compression ratios. The pyramid pooling structure and dimension compression ratios are crucial when applying Transformer to dense prediction tasks. Hence, the semantic segmentation task with the semantic FPN decoder head [46] is used to analyze the parameter as it is sensitive to the multi-scale spatial information. The experimental setup for the parameter setting experiments remains the same as that trained on the Cityscapes dataset in 4.3, with the only difference being that the number of iterations is reduced to 40,000.

Parameter Setting of Pyramid Pooling ratios. We perform parameter setting studies of the pyramid pooling ratios for different stages to validate the significance of using multiple pooling ratios. The baseline consists of SHSA, MLP, relative position encoding module, and patch embedding module, where SHSA apply the compression ratios 1,2,4 and 8 in each stage respectively. Since the last two stage only contain a few tokens, we do not perform parameter setting study at stage 3 and 4. The pyramid pooling ratios are fixed at $\{1, 2, 4, 8\}$ and $\{1, 2, 4\}$ in the stage 3 and 4, respectively. Since the number of tokens needs to be significantly less than the token number before the pyramid pooling operation, the pooling ratios of the first two stages cannot be greater than 16. Hence, The pooling ratio of the first two stages is set to $\{1, 2, 4, 8, 16\}$ or $\{1, 2, 4, 8\}$. Results are reported in Table 7(a). We can observe that the non-overlapped pyramid pooling operation with bigger pooling ratios at the first two stages can improve the segmentation performance. And the performance becomes higher when the first two stages are applied with the $\{1, 2, 4, 8, 16\}$ pooling ratios.

Parameter Setting of Compression ratios. We conduct experiments for different compression ratios operations, as shown in Table 7(b). There are three typical choices, *i.e.*, $\{1, 2, 4, 8\}$, $\{1, 2, 4, 4\}$ and $\{1, 2, 2, 2\}$, where each of them represents the compression ratio in the four stages. The baseline consists of SHSA, MLP, relative position encoding module, and patch embedding module, where SHSA apply the pooling ratios 1, 2, 4, 8 and 16 in each level at the first two stage, respectively. Since the dimension of the first two stage is less, we do not perform parameter setting study at stage 1 and 2, which are set to the fixed ratios 1 and 2. The results are shown in Table 7(b). As can be seen, the dimension compression operation with large compression ratios (*e.g.*, 4, 8) has a better performance. And the performance becomes higher when the four stages are applied with the $\{1, 2, 4, 8\}$ compression ratios.

Table 7
Experimental results of parameter setting. Parameter setting experimental results on the validation set of the Cityscapes [35] dataset. Pyramid pooling ratios represent the pooling output size of different levels of pyramid pooling. The compression ratios represent the dimensionality reduction factor for each stage.

(a)

Cityscapes					mIoU (%)	#params. (M)
Pyramid pooling ratios						
Level 1	Level 2	Level 3	Level 4	Level 5		
1	**2**	**4**	**8**	**16**	57.8	42.8
1	**2**	**4**	**8**	–	**57.7** (↓ 0.1)	42.8 (–)

(b)

Cityscapes				mIoU (%)	#params. (M)
Compression ratios					
Stage 1	Stage 2	Stage 3	Stage 4		
1	**2**	**2**	**2**	50.6	47.6
1	**2**	**4**	**4**	52.0 (↑ 1.4)	43.6 (↓ 4.0)
1	**2**	**4**	**8**	**57.7** (↑ 7.1)	**42.8** (↓ 4.8)

5. Conclusions

This paper has presented a lightweight Transformer-based backbone, call-ed mixture lightweight Transformer (MLT), for 2D scene understanding tasks. The designed mixture lightweight backbone enables us to mitigate computational complexity and reduce model size, yet at the same time maintains competitive accuracy. Furthermore, MLT performs well in dense prediction tasks, since the convolution is integrated into the backbone. To reduce the high computational complexity caused by the self attention mechanism, a lightweight self-attention module, Single-Head Self-Attention (SHSA), is designed in backbone. We also raise the Convolutional Inception Module (CIM) in feed forward network, which greatly reduces number of parameters while provides multi-scale features. We have evaluated our method on several computer vision tasks: image classification, semantic segmentation, object detection and instance segmentation. The experimental results demonstrate that MLT achieves state-of-the-art trade-off in terms of accuracy and model size. In the future, we are interested in two directions to improve MLT. As shown in Table 3, there is still a large performance gap between MLT and high-accuracy Transformer-based backbone in the image classification task, requiring further efforts to improve our model. In addition to achieving superior performance for image classification, semantic segmentation, object detection and instance segmentation, we believe that MLT can be easily used for other visual tasks, such as medical image segmentation [49–51] and scene text recognition [52,53].

CRediT authorship contribution statement

Quan Zhou: Conceptualization, Resources, Writing – original draft. **Zhenhan Sun:** Methodology, Formal analysis, Writing – original draft. **Linjie Wang:** Methodology, Software, Visualization – original draft. **Bin Kang:** Review & editing – revised draft. **Suofei Zhang:** Review & editing – revised draft. **Xiaofu Wu:** Review & editing – revised draft.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.compeleceng.2023.108698.

Data availability

No data was used for the research described in the article

Acknowledgments

The authors would like to thank associate editor and all anonymous reviewers for their helpful and insight comments. This work was supported by the National Natural Science Foundation of China under Grants 61876093, 62171232.

References

[1] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I. Attention is all you need. In: NIPS. 2017, p. 5998–6008.
[2] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR. 2021.
[3] Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B. Swin Transformer: hierarchical vision transformer using shifted windows. In: ICCV. 2021, p. 10012–22.
[4] Zhang Q, Yang Y. ResT: An efficient transformer for visual recognition. 2021, CoRR abs/2105.13677.
[5] Wang W, Xie E, Li X, Fan D, Song K, Liang D, Lu T, Luo P, Shao L. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV. 2021, p. 568–78.
[6] Fang Y, Liao B, Wang X, Fang J, Qi J, Wu R, Niu J, Liu W. You only look at one sequence: Rethinking transformer in vision through object detection. In: NIPS. 2021.
[7] Chen Y, Dai X, Chen D, Liu M, Dong X, Yuan L, Liu Z. Mobile-former: Bridging MobileNet and transformer. 2021, CoRR abs/2108.05895.
[8] Ge C, Liang Y, Song Y, Jiao J, Wang J, Luo P. Revitalizing CNN attentions via transformers in self-supervised visual representation learning. 2021, CoRR abs/2110.05340.

[9] Wu Y, Liu Y, Zhan X, Cheng M. P2T: pyramid pooling transformer for scene understanding. 2021, CoRR abs/2106.12011.

[10] Szegedy C, Ioffe S, Vanhoucke V, Alemi AA. Inception-v4, inception-ResNet and the impact of residual connections on learning. In: AAAI. 2017.

[11] Khan A, Sohail A, Zahoora U, Qureshi AS. A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 2020;5455–516.

[12] Chaki J, Woźniak M. Deep learning for neurodegenerative disorder (2016 to 2022): A systematic review. Biomed Signal Process Control 2023;80:104223.

[13] Praveen S, Srinivasu P, Shafi J, Wozniak M, Ijaz M. ResNet-32 and FastAI for diagnoses of ductal carcinoma from 2D tissue slides. Sci Rep 2022;12.

[14] Woźniak M, Wieczorek M, Siłka J. Deep neural network with transfer learning in remote object detection from drone. In: Proceedings of the 5th international ACM mobicom workshop on drone assisted wireless communications for 5g and beyond. Association for Computing Machinery; 2022.

[15] Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, Lu H. Dual attention network for scene segmentation. In: CVPR. 2019, p. 3146–54.

[16] Li S, Zhou Q, Liu J. DCM: A dense-attention context module for semantic segmentation. In: 2020 IEEE international conference on image processing (ICIP). 2020, p. 1431–5.

[17] Shi H, Zhou Q, Ni Y, Wu X. DPNET: Dual-path network for efficient object detection with lightweight self-attention. In: 2022 IEEE international conference on image processing (ICIP). 2022, p. 771–5.

[18] Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H. Training data-efficient image transformers & distillation through attention. In: ICML. 2021, p. 10347–57.

[19] Hu H, Zhang Z, Xie Z, Lin S. Local relation networks for image recognition. In: ICCV. 2019, p. 3463–72.

[20] Zheng S, Lu J, Zhao H, Zhu X, Luo Z, Wang Y, Fu Y, Feng J, Xiang T, Torr PHS, Zhang L. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR. 2021, p. 6881–90.

[21] Yuan L, Chen Y, Wang T, Yu W, Shi Y, Jiang Z-H, Tay FE, Feng J, Yan S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In: CVPR. 2021, p. 558–67.

[22] Wu H, Xiao B, Codella N, Liu M, Dai X, Yuan L, Zhang L. CvT: Introducing convolutions to vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV). 2021, p. 22–31.

[23] Fan H, Xiong B, Mangalam K, Li Y, Yan Z, Malik J, Feichtenhofer C. Multiscale vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV). 2021, p. 6824–35.

[24] Michel P, Levy O, Neubig G. Are sixteen heads really better than one? In: Neural information processing systems. 2019.

[25] Prangemeier T, Reich C, Koeppl H. Attention-based transformers for instance segmentation of cells in microstructures. In: 2020 IEEE international conference on bioinformatics and biomedicine (BIBM). 2020, p. 700–7.

[26] Lin K, Wang L, Liu Z. End-to-end human pose and mesh reconstruction with transformers. In: 2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2021, p. 1954–63. http://dx.doi.org/10.1109/CVPR46437.2021.00199.

[27] Esser P, Rombach R, Ommer B. Taming transformers for high-resolution image synthesis. In: 2021 IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2021, p. 12868–78.

[28] Chen J, Lu Y, Yu Q, Luo X, Adeli E, Wang Y, Lu L, Yuille AL, Zhou Y. TransUNet: Transformers make strong encoders for medical image segmentation. 2021, CoRR abs/2102.04306.

[29] Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. 2015, p. 234–41.

[30] Ba J, Kiros JR, Hinton GE. Layer normalization. 2016, ArXiv abs/1607.06450.

[31] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications. 2017, CoRR abs/1704.04861.

[32] Deng J, Dong W, Socher R, Li L, Li K, Fei-Fei L. ImageNet: A large-scale hierarchical image database. In: CVPR. 2009, p. 248–55.

[33] Lin T, Maire M, Belongie SJ, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL. Microsoft COCO: common objects in context. In: ECCV. 2014, p. 740–55.

[34] Zhou B, Zhao H, Puig X, Xiao T, Fidler S, Barriuso A, Torralba A. Semantic understanding of scenes through the ADE20K dataset. Int J Comput Vis 2019;302–21.

[35] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B. The cityscapes dataset for semantic urban scene understanding. In: CVPR. 2016, p. 3213–23.

[36] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: ICLR. 2015.

[37] Wang Y, Ma X, Chen Z, Luo Y, Yi J, Bailey J. Symmetric cross entropy for robust learning with noisy labels. In: 2019 IEEE/CVF international conference on computer vision (ICCV). 2019, p. 322–30. http://dx.doi.org/10.1109/ICCV.2019.00041.

[38] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: CVPR. 2016, p. 770–8.

[39] Han K, Xiao A, Wu E, Guo J, Xu C, Wang Y. Transformer in transformer. 2021, CoRR abs/2103.00112.

[40] Zhang P, Dai X, Yang J, Xiao B, Yuan L, Zhang L, Gao J. Multi-scale vision longformer: A new vision transformer for high-resolution image encoding. In: CVPR. 2021, p. 2998–3008.

[41] He K, Gkioxari G, Dollár P, Girshick RB. Mask R-CNN. In: ICCV. 2017, p. 2980–8.

[42] Chen K, Wang J, Pang J, Cao Y, Xiong Y, Li X, Sun S, Feng W, Liu Z, Xu J, Zhang Z, Cheng D, Zhu C, Cheng T, Zhao Q, Li B, Lu X, Zhu R, Wu Y, Dai J, Wang J, Shi J, Ouyang W, Loy CC, Lin D. MMDetection: Open MMlab detection toolbox and benchmark. 2019, CoRR abs/1906.07155.

[43] Sun P, Zhang R, Jiang Y, Kong T, Xu C, Zhan W, Tomizuka M, Li L, Yuan Z, Wang C, Luo P. Sparse R-CNN: end-to-end object detection with learnable proposals. In: CVPR. 2021, p. 14454–63.

[44] Zhao H, Gallo O, Frosio I, Kautz J. Loss functions for image restoration with neural networks. IEEE Trans Comput Imaging 2017;3(1):47–57. http://dx.doi.org/10.1109/TCI.2016.2644865.

[45] Xiao T, Liu Y, Zhou B, Jiang Y, Sun J. Unified perceptual parsing for scene understanding. In: ECCV. 2018, p. 432–48.

[46] Kirillov A, Girshick RB, He K, Dollár P. Panoptic feature pyramid networks. In: CVPR. 2019, p. 6399–408.

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

[48] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. 2010, p. 249–56.

[49] Shamshad F, Khan S, Zamir. Transformers in medical imaging: A survey. 2022, arXiv preprint arXiv:2201.09873.

[50] Valanarasu JMJ, Oza P. Medical transformer: Gated axial-attention for medical image segmentation. In: Medical image computing and computer assisted intervention – MICCAI 2021. 2021, p. 36–46.

[51] Chen J, Lu Y, a QY. TransUNet: Transformers make strong encoders for medical image segmentation. 2021, ArXiv abs/2102.04306.

[52] Wang K, Babenko B, Belongie S. End-to-end scene text recognition. In: 2011 International conference on computer vision. 2011, p. 1457–64.

[53] Li M, Lv T, Chen J, Cui L, Lu Y, Florencio D, Zhang C, Li Z, Wei F. TrOCR: Transformer-based optical character recognition with pre-trained models. In: AAAI 2023. 2023.

Quan Zhou received Ph.D. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China in 2013. Now he is an associated professor in the college of Telecommunications and Information engineering at Nanjing University of Posts and Telecommunications. His research interests include computer vision and pattern recognition.

Zhenhan Sun received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2023. His research interests include computer vision and pattern recognition.

Linjie Wang received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2023. His research interests include computer vision and pattern recognition.

Bin Kang received the Ph.D. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2016. He is currently an Associate Professor with the College of Internet of Things, Nanjing University of Posts and Telecommunications. His research interests include computer vision and pattern recognition.

Suofei Zhang received Ph.D. degree from the School of Information Science and Engineering, Southeast University, Nanjing, China, in 2013. He is currently a Lecturer with the School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing. His current research interests include image processing, machine learning, computer vision, and artificial intelligence.

Xiaofu Wu received Ph.D. degree in electrical engineering from Peking University, Beijing, China, in 2005. Since 2012, he has been with Nanjing University of Posts and Telecommunications, Nanjing, where he is currently a Full Professor. His research interests are in coding and information theory, information-theoretic security, machine learning, and computer vision.