remote sensing

MDPI

Article

Faster and Better: A Lightweight Transformer Network for Remote Sensing Scene Classification

Xinyan Huang [1,2,3,4], Fang Liu [1,2,3,4,*], Yuanhao Cui [1,2,3,4], Puhua Chen [1,2,3,4], Lingling Li [1,2,3,4] and Pengfang Li [1,2,3,4]

[1] Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China; xinyanh@stu.xidian.edu.cn (X.H.); yhcui@stu.xidian.edu.cn (Y.C.); phchen@xidian.edu.cn (P.C.); llli@xidian.edu.cn (L.L.); pfli33@stu.xidian.edu.cn (P.L.)
[2] International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an 710071, China
[3] Joint International Research Laboratory of Intelligent Perception and Computation, Xidian University, Xi'an 710071, China
[4] School of Artificial Intelligence, Xidian University, Xi'an 710071, China
* Correspondence: liuf63@xidian.edu.cn

Abstract: Remote sensing (RS) scene classification has received considerable attention due to its wide applications in the RS community. Many methods based on convolutional neural networks (CNNs) have been proposed to classify complex RS scenes, but they cannot fully capture the context in RS images because of the lack of long-range dependencies (the dependency relationship between two distant elements). Recently, some researchers fine-tuned the large pretrained vision transformer (ViT) on small RS datasets to extract long-range dependencies effectively in RS scenes. However, it usually takes more time to fine-tune the ViT on account of high computational complexity. The lack of good local feature representation in the ViT limits classification performance improvement. To this end, we propose a lightweight transformer network (LTNet) for RS scene classification. First, a multi-level group convolution (MLGC) module is presented. It enriches the diversity of local features and requires a lower computational cost by co-representing multi-level and multi-group features in a single module. Then, based on the MLGC module, a lightweight transformer block, LightFormer, was designed to capture global dependencies with fewer computing resources. Finally, the LTNet was built using the MLGC and LightFormer. The experiments of fine-tuning the LTNet on four RS scene classification datasets demonstrate that the proposed network achieves a competitive classification performance under less training time.

Keywords: lightweight transformer; convolutional neural network; scene classification; remote sensing

Citation: Huang, X.; Liu, F.; Cui, Y.; Chen, P.; Li, L.; Li, P. Faster and Better: A Lightweight Transformer Network for Remote Sensing Scene Classification. *Remote Sens.* **2023**, *15*, 3645. https://doi.org/10.3390/rs15143645

Academic Editor: Danfeng Hong

Received: 15 June 2023
Revised: 17 July 2023
Accepted: 18 July 2023
Published: 21 July 2023

1. Introduction

Remote sensing (RS) scene classification has attracted much attention due to its wide application requirements in practical scenarios, such as urban planning [1], natural hazards detection [2–4], land-cover classification [5], and geographic image retrieval [6,7]. Inspired by the great success of convolutional neural networks (CNNs) [8–13] in the computer vision research field, many CNN-based methods have been proposed to classify complex RS scenes. Specifically, some researchers fine-tuned large pretrained CNNs on small RS datasets [14–17], which improves the local feature representation capability in RS images. In addition, some methods aggregate multi-layer convolutional features [18–23] or capture contextual information [24,25] for RS scene classification. Although these CNN-based methods have significantly improved the classification performance, they cannot fully capture the long-range dependencies in complex RS scenes.

In 2017, Vaswani et al. [26] presented the well-known transformer architecture. It can leverage the self-attention mechanism to extract long-range dependencies effectively. Afterward, Dosovitskiy et al. [27] applied the transformer to image recognition, establishing

a pure transformer-based model (ViT) for the visual task. Motivated by it, many transformer architectures have been built for vision tasks, such as image classification [28,29], object detection [30,31], segmentation [32–34], and Video Anomaly Detection [35]. As an important vision task, some RS scene classification methods [36,37] attempted to introduce transformer architectures for extracting long-range dependencies. Bazi et al. [36] fine-tuned the large pretrained ViT on a small RS dataset. Ma et al. [37] used the ViT to learn global context information. However, these pure transformer model lacks good local feature representation, leading to limited performance improvements for RS scene classification.

For the complex RS scene classification task, extracting local representations and long-range dependencies is crucial, and combining a CNN and transformer in one network is an effective solution. CvT [28] and BoTNet [38] combine the transformer with the convolution to build hybrid transformers, which have the ability to represent long-range dependencies and local features simultaneously. However, their extraction processes require high-cost computation. Consequently, designing a lightweight pretrained model is necessary for RS scene classification. Some lightweight networks have been proposed to reduce the computational complexity of the traditional convolution and transformer. For instance, ordinary convolutions in MobileNets [39–41] are factorized into depthwise and pointwise convolutions. To further reduce the computational cost, group convolution with channel shuffle is introduced into pointwise convolutions in ShuffleNets [42,43]. Different from the above works, Han et al. [44] considered that the redundancy in feature maps is essential for a successful deep neural network and thus designed the cost-efficient Ghost module that uses cheap operations to acquire the redundancy in output feature maps. In addition, several approaches, such as ViT-Lite [45] and LeViT [46], focused on designing efficient lightweight hybrid transformer networks. These methods are effective in processing natural images. However, in RS scene classification, the question of how to design an efficient model that can learn local representation and long-range dependencies remains to be explored.

In this paper, we propose a lightweight transformer network (LTNet) for RS scene classification. Please see Figure 1 for its overview. First, we designed an efficient multi-level group convolution (MLGC) module in which multi-level and multi-group features are co-represented, enriching the diversity of local features and reducing the computational cost. Second, we developed a lightweight transformer (LightFormer) block based on the MLGC module. It can efficiently capture the global dependencies of RS scenes. Finally, we leveraged the proposed MLGC module and LightFormer block to construct an efficient lightweight transformer network (LTNet), which efficiently aggregates multi-level local features and long-range dependencies. We evaluated the LTNet on four common RS scene classification datasets. The experimental results show that both its computational cost and fine-tuning time are lower, while its performance is competitive in comparison with its counterparts.

The main contributions of this paper are listed as follows:

- An MLGC module with a low computational cost is proposed, which utilizes the co-representations of multi-level and multi-group features to enrich the diversity of local features in the RS scene.
- By introducing the MLGC module into the ordinary transformer block, we designed a LightFormer block with fewer parameters and FLOPs, which considers both rich multi-level local features and long-range dependencies in RS images.
- We built the efficient LTNet based on the MLGC module and LightFormer block for RS scene classification.
- Experiments on four RS scene classification datasets prove that the LTNet achieves a competitive classification performance with less training time.

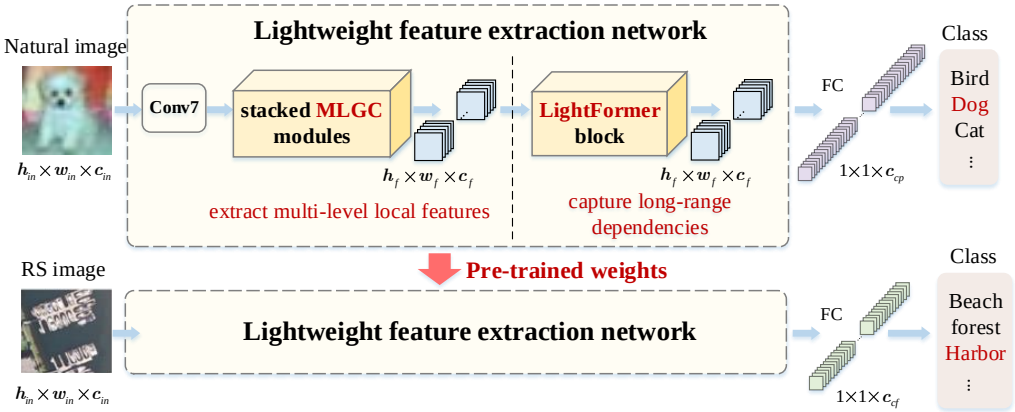

Figure 1. Overview of the LTNet. We first pretrained LTNet on the ImageNet dataset [47] and then fine-tuned it on the RS scene classification datasets [15,24,48,49]. In the LTNet, we used convolution with a 7×7 kernel size (Conv7) and multiple stacked MLGC modules to extract multi-level local features. These features were fed into a LightFormer block to capture long-range dependencies.

The remainder of this paper is described as follows. We briefly review the related work in Section 2. In Section 3, we describe the proposed MLGC module, LightFormer block, and efficient LTNet in detail. The experimental results and analysis are presented in Section 4. Finally, the conclusion is drawn in Section 5.

2. Related Work

In this section, we briefly review CNN-based RS scene classification methods, lightweight CNNs, and vision transformer networks, which are closely related to our work.

2.1. CNN-Based RS Scene Classification Methods

To exploit the utilization of existing CNNs for RS scene lassification, Nogueira et al. [14] compared the performance of the three strategies, including full training, fine-tuning, and using CNNs as feature extractors. Fine-tuning tends to be the best strategy in experiments, and many methods for RS scene classification are based on pre-trained CNNs. Cheng et al. [15] introduced the results of fine-tuning the existing AlexNet, GoogLeNet, and VGG. Bazi et al. [16] presented a simple yet effective fine-tuning method of CNNs to tackle the vanishing gradient problem of transferring knowledge to small datasets. Li et al. [17] proposed the deep convolutional neural network (TL-DeCNN) model and fine-tuned the classification module. Due to the complexity of RS scenes, some works aggregate multi-layer convolutional features or capture contextual dependencies to improve classification accuracy. A feature aggregation CNN (FACNN) [18] was designed to aggregate features according to different spatial structure information. A gated bidirectional network (GBNet) [19] was presented to aggregate the hierarchical features. He et al. [20] combined multi-layer feature maps obtained by pretrained CNNs for RS scene classification. Liu et al. [21] adopted a two-stage deep feature fusion model, which integrates two converted CNNs for RS scene classification. Xue et al. [22] developed a multi-structure deep features fusion (MSDFF) to fuse the deep features from three CNNs. Two-branch deep feature fusion (TDFF) is adopted in an enhanced feature pyramid network (EFPN) [23] to aggregate the features at different levels. Wang et al. [24] proposed an attention recurrent convolutional network (ARCNet) to select a series of attention regions adaptively. Tang et al. [25] developed an attention consistent network (ACNet) based on the Siamese network. However, these methods do not fully capture the long-range dependencies of complex RS scenes.

2.2. Vision Transformer Networks

The ViT [27], a pure transformer model, performs significantly well on the image classification task due to the ability to extract long-range dependencies. Based on this work, more researchers [50–53] focused on vision transformers. The ability is also important

for the complex RS scene classification task. Bazi et al. [36] used the ViT for RS scene classification. Ma et al. [37] proposed a homo–heterogenous transformer learning (HHTL) framework based on a vision transformer to obtain the context information. However, numerous parameters and FLOPs consume enormous computation resources and make fine-tuning the network take a long time. The lack of good local feature representation in the ViT limits the classification performance of the RS scene. Hence, some hybrid transformer models are proposed based on convolutions and transformers to solve the above issues. CvT [28] incorporates the convolutional token embedding and convolutional projection into the ViT. BoTNet [38] uses self-attention to replace convolutions in the last three bottleneck blocks of a ResNet. Graham et al. [46] applied convolutions to transformers and introduced the attention bias for the trade-off between accuracy and efficiency. Hassani et al. [45] presented three different models, ViT-Lite, CVT, and CCT, to address the "data-hungry" issue for the transformer. MobileViT [54] leverages the strengths of CNNs and ViTs to build a lightweight network for mobile vision tasks. Inspired by the above work, this paper introduces an LTNet for RS scene classification, and the LightFormer block was designed in the network to capture long-range dependencies.

2.3. Lightweight CNNs

Lightweight CNNs [55–59] ensure the accuracy of the model while effectively reducing the computational cost. Especially in recent years, lightweight architecture design has achieved remarkable performances. Howard et al. [39] used depthwise separable convolutions to build efficient MobileNets. Sandler et al. [40] presented an inverted residual with a linear bottleneck for MobileNetV2. Howard et al. [41] utilized a network architecture search (NAS) to search for the global network structures of MobileNetV3. Zhang et al. [42] described pointwise group convolution and channel shuffle to reduce the computational cost. Ma et al. [43] provided several practical guidelines for efficient network design and proposed ShuffleNetV2 according to these guidelines. Han et al. [44] established an efficient neural architecture, GhostNet, based on the Ghost module that generates more feature maps from cheap operations. Inspired by the above works that designed efficient convolutional operations or only acquired the redundancy of output feature maps cost-efficiently, the redundancy in the input and output feature maps is simultaneously considered in the proposed MLGC module to reduce the computational cost further.

3. Method

In this section, we first present an efficient MLGC module with a low computational cost. Then, based on the MLGC module, we outline a designed LightFormer block that learns long-range dependencies with fewer parameters. Finally, we introduce our LTNet.

As shown in Figure 1, for the original input image $X \in \mathbb{R}^{h_{in} \times w_{in} \times c_{in}}$, where h_{in} and w_{in} represent the height and width of the input image with c_{in} channels. Features extracted through stacked MLGC modules are denoted as $Z \in \mathbb{R}^{h_f \times w_f \times c_f}$, where h_f and w_f represent the height and width of the obtained feature maps, respectively. c_f is the number of channels. Then, local features Z are fed into the LightFormer block to capture global features $Z_1 \in \mathbb{R}^{h_f \times w_f \times c_f}$.

3.1. MLGC Module

In an ordinary convolutional layer, rich local features that contain much redundancy can be extracted by a large number of convolution kernels. As mentioned in the literature [44], redundancy is essential for a successful deep neural network, but it also increases the computational burden. The proper retention of redundancy features can increase the model complexity and improve the model's ability to fit complex data, thus improving the network's performance. There exists much redundancy (i.e., some similar feature map pairs) in output feature maps. Instead of avoiding the redundant feature maps, the Ghost module generates them in a low-cost way. The output feature maps of the Ghost module are split into s parts (s = 2, 3, 4, or 5). The first part is calculated by the ordinary

convolutions from all input feature maps, and the other parts are obtained by linear operations (depthwise convolutions) from the first part. Depthwise convolution is to apply a single filter for each input channel. The Ghost module obtains a superior performance at $s = 2$. As shown in Figure 2a, given the input feature maps $F \in \mathbb{R}^{h \times w \times c}$, where h and w represent the height and width of the input feature map with c input channels, it can be seen that the output feature maps $F' \in \mathbb{R}^{h' \times w' \times c'}$ are obtained by concatenating two parts, $F_1' \in \mathbb{R}^{h' \times w' \times (c'/2)}$ and $F_2' \in \mathbb{R}^{h' \times w' \times (c'/2)}$, where h' and w', respectively, represent the height and width of the output feature maps and c' is the number of channels.

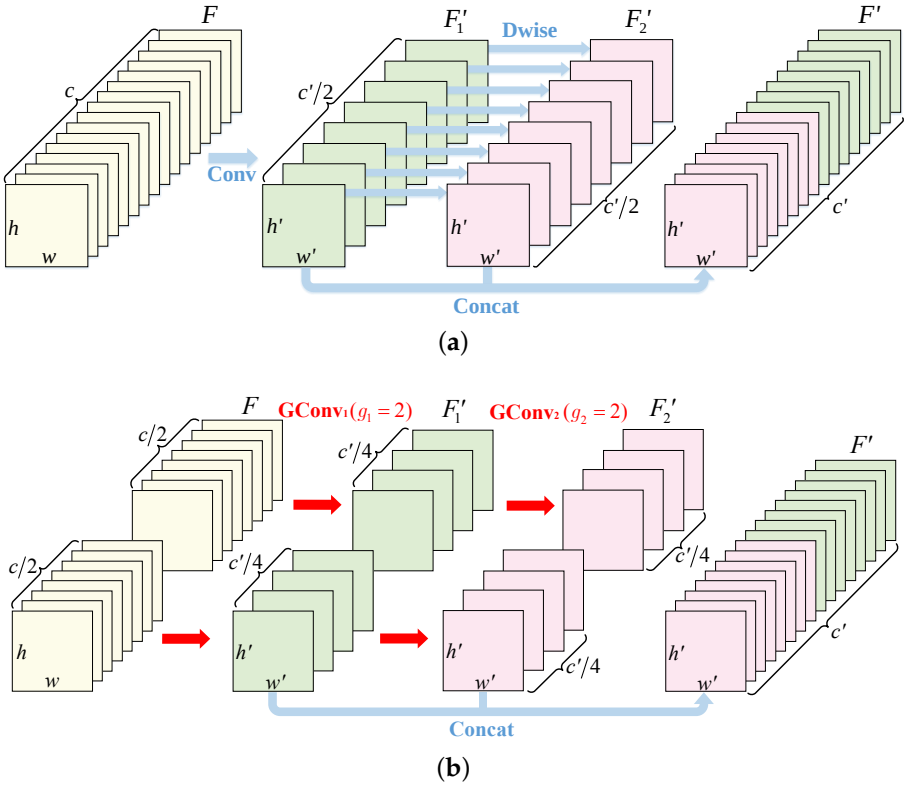

Figure 2. An illustration of the Ghost module and our MLGC module: (**a**) Ghost module ($s = 2$) and (**b**) MLGC module ($g_1 = 2$, $g_2 = 2$). The output feature maps F' are made up of $s = 2$ parts (F_1' and F_2'). F_1' and F_2' are generated from all input features F and the first output part F_1', respectively. Instead of using the ordinary convolution (Conv) and depthwise convolution (Dwise) to calculate two output parts like the Ghost module, we use two group convolutions, GConv_1 with g_1 groups and GConv_2 with g_2 groups, in our MLGC module.

Generate the output feature maps F' formulated as

$$F_1{}' = \text{Conv}(F), \tag{1}$$

$$F_2{}' = \text{Dwise}(F_1{}'), \tag{2}$$

$$F' = \text{Concat}(F_1{}', F_2{}'), \tag{3}$$

where $k_1 \times k_1$ convolution operator $\text{Conv}()$ could produce the intrinsic feature map $F_1{}'$ from all input data F. $\text{Dwise}()$ is a $k_2 \times k_2$ depthwise convolution, which generates Ghost feature maps $F_2{}'$ from the first part $F_1{}'$. Feature maps $F_1{}'$ and $F_2{}'$ are concatenated to acquire F'.

The parameters of the Ghost module are

$$P_G = k_1 \cdot k_1 \cdot c \cdot \frac{c'}{2} + k_2 \cdot k_2 \cdot \frac{c'}{2} = \frac{c'}{2} \cdot \left(c \cdot {k_1}^2 + {k_2}^2 \right). \tag{4}$$

The computational cost of the Ghost module is

$$C_G = h \cdot w \cdot k_1 \cdot k_1 \cdot c \cdot \frac{c'}{2} + h \cdot w \cdot k_2 \cdot k_2 \cdot \frac{c'}{2} = \frac{c'}{2} \cdot h \cdot w \cdot \left(c \cdot {k_1}^2 + {k_2}^2 \right). \tag{5}$$

Although the GhostNet achieves a great performance, only the redundancy in the output feature maps F' of convolutional layers is considered in the Ghost module. As described in Equation (1), the first part ${F_1}'$ of the output feature maps F' is calculated by the ordinary convolutions for all input feature maps (i.e., the output feature maps of the previous layer). Hence, input feature maps also have much redundancy, which is not considered in the Ghost module. Furthermore, the second part ${F_2}'$ is obtained by linear operations (depthwise convolutions) from the first part (Equation (2)), but each channel in the first part of the output feature maps is not always similar to one channel in the second part. Using linear operations reduces the diversity of output features.

To solve the above issues, this paper presents an MLGC module that simultaneously considers the redundancy in the input and output feature maps, as shown in Figure 2b. The module retains sufficient redundancy in the output feature maps. Specifically, in the MLGC module, besides considering much redundancy in the output features F' (split F' into two parts ${F_1}'$ and ${F_2}'$), the group convolution is used to generate the first part ${F_1}'$ from all input feature maps F (Equation (6)), which simultaneously considers the redundancy of input feature maps. In addition, the group convolution on ${F_1}'$ is utilized to acquire ${F_2}'$ (Equation (7)). Each feature map in the second part is a combination of multiple feature maps in the first part, which captures rich features and helps the network achieve an excellent performance. We call ${F_1}'$ and ${F_2}'$ first-level and second-level features, respectively. The output feature maps F' of each MLGC module include multi-level information.

Generate output feature maps F' formulated as

$${F_1}' = \mathrm{GConv}_1(F), \tag{6}$$

$${F_2}' = \mathrm{GConv}_2({F_1}'), \tag{7}$$

where $\mathrm{GConv}_1()$ and $\mathrm{GConv}_2()$ denote the $k_1 \times k_1$ group convolution operator with g_1 group and $k_2 \times k_2$ group convolution operator with g_2 group, respectively. Two-level feature maps ${F_1}'$ and ${F_2}'$ are concatenated to obtain F', which is composed of two parts at different levels.

The parameters of the MLGC module are

$$\begin{aligned} P_M =& k_1 \cdot k_1 \cdot \frac{c}{g_1} \cdot \frac{c'}{2g_1} \cdot g_1 + k_2 \cdot k_2 \cdot \frac{c'}{2g_2} \cdot \frac{c'}{2g_2} \cdot g_2 \\ =& \frac{c'}{2} \cdot \left(\frac{c}{g_1} \cdot {k_1}^2 + \frac{c'}{2g_2} \cdot {k_2}^2 \right). \end{aligned} \tag{8}$$

The computational cost of the MLGC module is

$$\begin{aligned} C_M =& h \cdot w \cdot k_1 \cdot k_1 \cdot \frac{c}{g_1} \cdot \frac{c'}{2g_1} \cdot g_1 + h \cdot w \cdot k_2 \cdot k_2 \cdot \frac{c'}{2g_2} \cdot \frac{c'}{2g_2} \cdot g_2 \\ =& \frac{c'}{2} \cdot h \cdot w \cdot \left(\frac{c}{g_1} \cdot {k_1}^2 + \frac{c'}{2g_2} \cdot {k_2}^2 \right). \end{aligned} \tag{9}$$

The ratio of the Ghost module and MLGC module in the computational cost is

$$R = \frac{C_G}{C_M} = \frac{\frac{c'}{2} \cdot h \cdot w \cdot \left(c \cdot {k_1}^2 + {k_2}^2\right)}{\frac{c'}{2} \cdot h \cdot w \cdot \left(\frac{c}{g_1} \cdot {k_1}^2 + \frac{c'}{2g_2} \cdot {k_2}^2\right)} \approx \frac{2g_1 \cdot g_2 \cdot (c+1)}{g_1 \cdot c' + 2g_2 \cdot c}, \tag{10}$$

where the value of k_1 is equal to or similar to k_2, such as 1 or 3, so $k_1 \approx k_2 = k$. If $R > 1$, the computational cost of the proposed MLGC module is lower than that of the Ghost Module. A larger R indicates that the advantage of our module is more obvious than the Ghost module.

3.2. LightFormer Block

Recently, based on the self-attention mechanism, transformers have built long-range dependencies and have been successfully applied to vision tasks. In this paper, we designed a LightFormer block, which consumes less computing resources to capture the long-range dependencies. Figure 3 shows the LightFormer block. It has five differences from the ViT [27]: (i) The linear operation in the ViT was replaced by the MLGC module (the kernel size of 1×1), reducing parameters and FLOPs. We also tried MLGC with a kernel size of 3×3 in the experiment. It was found that the accuracy was not improved over the 1×1 MLGC module; (ii) Because of the strong multi-level feature representation ability of the MLGC module, we removed MLP layers that need many computational resources; (iii) Since batch normalization is used in the MLGC module, layer normalization in the ViT was removed; (iv) Adding class tokens to the sequence is unnecessary. The LightFormer block is followed by the FC layer for classification; (v) The MLGC module already contains position information. Since the local context structure can be acquired by convolutions, the MLGC module can learn local spatial information. It is not necessary to preserve position embedding for the LightFormer block.

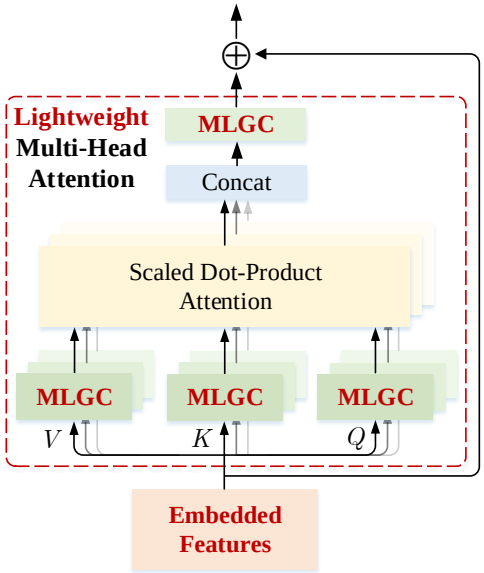

Figure 3. LightFormer block. We utilized MLGC modules for replacing linear projection, and the LayerNorm (LN) and MLP layers were removed. The local features extracted by the stacked MLGC module were fed into the LightFormer block as embedded features.

In Figure 3, the lightweight multi-head attention (LMHA) is given in the proposed LightFormer block and is followed by the use of the residual connection.

$$Z_1 = \mathrm{LMHA}(Z) + Z, \tag{11}$$

$$\mathrm{LMHA}(Q, K, V) = \mathrm{MLGC}^{A}(\mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_m)), \tag{12}$$

$$\mathrm{head}_j = \mathrm{Attention}(\mathrm{MLGC}_j^{Q}(Q), \mathrm{MLGC}_j^{K}(K), \mathrm{MLGC}_j^{V}(V)), \tag{13}$$

where $\mathrm{MLGC}_j^{Q}()$, $\mathrm{MLGC}_j^{K}()$, and $\mathrm{MLGC}_j^{V}()$ are MLGC convolution operations for query matrix Q, key matrix K, and value matrix V with the jth head. The number of heads is $m = 8$. $\mathrm{MLGC}^{A}()$ is an MLGC convolution operation for all heads.

3.3. LTNet

In this work, we aim to develop a lightweight vision transformer network for RS scene classification. To simply and clearly show how we design the network based on the MLGC module and LightFormer block, we give the details of building the LightFormer-VGG-16 and then further present the details of building LTNet for RS scene classification.

As shown in Table 1, VGG-16 [60] is a common convolutional neural network, which is built on the ordinary convolution, and Ghost-VGG-16 [44] is obtained by plugging the Ghost module into VGG-16. We replaced each Ghost module in Ghost-VGG-16 with the MLGC module to acquire MLGC-VGG-16. The LightFormer block was introduced into our MLGC-VGG-16 to establish the LightFormer-VGG-16, which uses fewer layers than the other three networks. The stacked MLGC modules can be used to learn the ability of the inductive biases and extract richer local features that are fed into a single LightFormer block to capture long-range dependencies. Given $g_1 = g_2 = 2$, the ratio between the Ghost module and MLGC module in the computational cost is $R = \frac{4(c+1)}{2c+c'}$. Since most layers in Ghost-VGG-16 or MLGC-VGG-16 have the same number of input and output channels ($c' = c$) and a few other layers is $c' = 2c$, the corresponding R is calculated as $R = \frac{4(c+1)}{3c}$ and $R = \frac{c+1}{c}$. On the whole, $R > 1$ shows that the computational cost of the MLGC module is reduced. It can be seen from Table 1 that our LightFormer-VGG-16 uses fewer layers, weights, and FLOPs than the VGG-16 and Ghost-VGG-16.

Table 1. Architectures, weights, and FLOPs of VGG-16 [60], Ghost-VGG-16 [44], MLGC-VGG-16, and LightFormer-VGG-16 on CIFAR-10 [61]. Conv, Ghost, and MLGC denote the ordinary convolution, Ghost module, and MLGC module, respectively. The value before and after the hyphen (-) represents the kernel size and the number of output channels, respectively. For example, Conv3-64 means the ordinary convolution with a 3×3 kernel size and 64 output channels. LightFormer-512 represents the LightFormer block with 512 channels based on the 1×1 MLGC module.

Layer Name	Output Size	VGG-16	Ghost-VGG-16	MLGC-VGG-16	LightFormer-VGG-16
v1	32×32	Conv3-64			
v2		Conv3-64	Ghost3-64	MLGC3-64	MLGC3-64
		Max pool			
v3	16×16	Conv3-128	Ghost3-128	MLGC3-128	MLGC3-128
v4		Conv3-128	Ghost3-128	MLGC3-128	MLGC3-128
		Max pool			
v5	8×8	Conv3-256	Ghost3-256	MLGC3-256	MLGC3-256
v6		Conv3-256	Ghost3-256	MLGC3-256	-
v7		Conv3-256	Ghost3-256	MLGC3-256	-

Table 1. *Cont.*

Layer Name	Output Size	VGG-16	Ghost-VGG-16	MLGC-VGG-16	LightFormer-VGG-16
		Max pool			
v8	4 × 4	Conv3-512	Ghost3-512	MLGC3-512	LightFormer-512
v9		Conv3-512	Ghost3-512	MLGC3-512	-
v10		Conv3-512	Ghost3-512	MLGC3-512	-
		Max pool			
v11	2 × 2	Conv3-512	Ghost3-512	MLGC3-512	-
v12		Conv3-512	Ghost3-512	MLGC3-512	-
v13		Conv3-512	Ghost3-512	MLGC3-512	-
		Max pool, FC-512, FC-10, Soft-max			
Weights (M)		15.0	7.7	6.0	0.8
FLOPs (M)		315	159	127	55

As shown in Table 2, a building block $[\cdot]$ with three ordinary convolutions was used to build ResNet-50 [62]. Ghost-ResNet-50 [44] is obtained by plugging the Ghost module into ResNet-50. We replaced the Ghost module in Ghost-ResNet-50 with the MLGC module and obtained MLGC-ResNet-50 to maintain the richer diversity of feature maps, given $g_1 = 2$ and $g_2 = 1$. The computational cost ratio of the Ghost module and MLGC module is $R = \frac{2(c+1)}{c+c'}$. It has three types of relations between input and output channels, including $c' = \frac{1}{2}c$, $c' = c$, and $c' = 4c$. The values of R for each of them are $\frac{4(c+1)}{3c}$, $\frac{c+1}{c}$, and $\frac{2(c+1)}{5c}$, respectively. Since R of the first two types is $R > 1$, which can reduce the network computational cost, we use the MLGC module to replace the Ghost module on the corresponding $c' = \frac{1}{2}c$ and $c' = c$ layers. In other words, our module has an advantage when the input channels are greater than or equal to the output channels. Combining two MLGC modules with one Ghost module in a building block can reduce computational costs. By plugging the LightFormer block into MLGC-ResNet-50, we established the LTNet, which is pretrained on the ImageNet dataset and fine-tuned on RS scene classification datasets. When fine-tuning, change the 1000 classes of the FC layer in Table 2 to the number of classes on the RS dataset. For example, we used FC-21 for the Merced dataset. Compared with the Ghost-ResNet-50 and MLGC-ResNet-50, the LTNet only has one building block in layer r5, which uses fewer layers, weights, and FLOPs.

Table 2. Architectures, weights, and FLOPs of ResNet-50 [62], Ghost-ResNet-50 [44], MLGC-ResNet-50, and LTNet on ImageNet [47]. The square brackets $[\cdot]$ before the multiplication sign × denote a building block of the network, and the value after × represents the number of stackings in this building block.

Layer Name	Output Size	ResNet-50	Ghost-ResNet-50	MLGC-ResNet-50	LTNet
r1	112 × 112	Conv7-64, Stride 2			
r2	56 × 56	Max pool, Stride2			
		$\begin{bmatrix} \text{Conv1-64} \\ \text{Conv3-64} \\ \text{Conv1-256} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{Ghost1-64} \\ \text{Ghost3-64} \\ \text{Ghost1-256} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{MLGC1-64} \\ \text{MLGC3-64} \\ \text{Ghost1-256} \end{bmatrix} \times 3$	$\begin{bmatrix} \text{MLGC1-64} \\ \text{MLGC3-64} \\ \text{Ghost1-256} \end{bmatrix} \times 3$
r3	28 × 28	$\begin{bmatrix} \text{Conv1-128} \\ \text{Conv3-128} \\ \text{Conv1-512} \end{bmatrix} \times 4$	$\begin{bmatrix} \text{Ghost1-128} \\ \text{Ghost3-128} \\ \text{Ghost1-512} \end{bmatrix} \times 4$	$\begin{bmatrix} \text{MLGC1-128} \\ \text{MLGC3-128} \\ \text{Ghost1-512} \end{bmatrix} \times 4$	$\begin{bmatrix} \text{MLGC1-128} \\ \text{MLGC3-128} \\ \text{Ghost1-512} \end{bmatrix} \times 4$

Table 2. *Cont.*

Layer Name	Output Size	ResNet-50	Ghost-ResNet-50	MLGC-ResNet-50	LTNet
r4	14 × 14	[Conv1-256, Conv3-256, Conv1-1024] × 6	[Ghost1-256, Ghost3-256, Ghost1-1024] × 6	[MLGC1-256, MLGC3-256, Ghost1-1024] × 6	[MLGC1-256, MLGC3-256, Ghost1-1024] × 6
r5	7 × 7	[Conv1-512, Conv3-512, Conv1-2048] × 3	[Ghost1-512, Ghost3-512, Ghost1-2048] × 3	[MLGC1-512, MLGC3-512, Ghost1-2048] × 3	[MLGC1-512, LightFormer-512, Ghost1-2048] × **1**
		Average pool, FC-1000, Soft-max			
Weights (M)		25.6	13.9	13.1	8.2
FLOPs (B)		4.1	2.2	1.9	1.7

4. Experiments

We conducted extensive experiments on four common RS scene classification datasets and two natural image classification benchmarks to evaluate the performance of the proposed method. First, to confirm the efficacy of the LTNet for RS scene classification, we performed experiments on four RS datasets (Merced [48], AID [49], Optimal-31 [24], and NWPU [15] datasets). Then, to verify the effectiveness of the proposed MLGC module and LightFormer block and the designed network architecture of the LTNet in detail, we performed experiments on two natural image classification benchmarks (CIFAR-10 [61] and ImageNet ILSVRC 2012 [47] datasets).

4.1. Experiments on Four RS Scene Classification Datasets

4.1.1. Dataset Description

Merced dataset (Figure 4): The University of California (UC) Merced land-use dataset contains 2100 RGB images of size 256 × 256 pixels, extracted from the United States Geological Survey National Map. The dataset is composed of 21 scenes, and each class consists of 100 images.

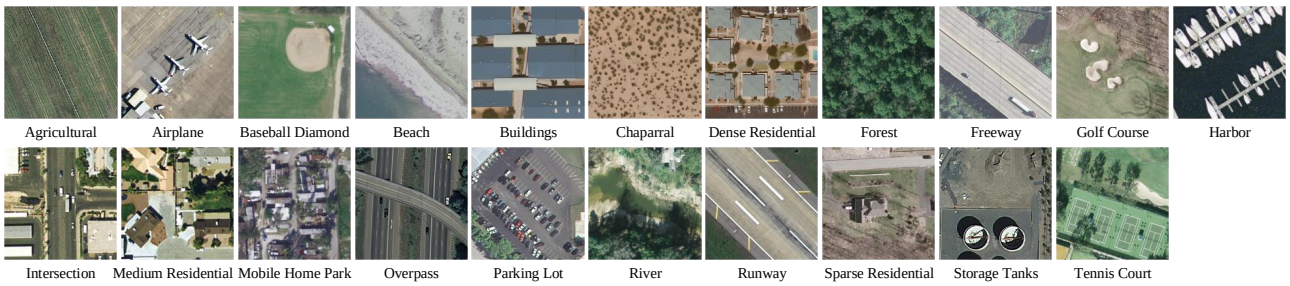

Figure 4. Some examples of the Merced dataset.

AID dataset (Figure 5): The aerial image dataset (AID) contains 10,000 aerial scene images of size 600 × 600 pixels extracted from Google Earth imagery. The dataset is composed of 30 different classes, and each class consists of 220 to 420 images.

Optimal-31 dataset (Figure 6): The Optimal-31 dataset contains 1860 RS images of size 256 × 256 pixels, extracted from Google Earth imagery. The dataset is composed of 31 classes, and each class consists of 60 images.

NWPU dataset (Figure 7): The NWPU-RESISC45 (NWPU) dataset contains 31,500 RGB images of size 256 × 256 pixels extracted from Google Earth imagery. The dataset is composed of 45 scene classes, and each class consists of 700 images.

Figure 5. Some examples of the AID dataset.

Figure 6. Some examples of the Optimal-31 dataset.

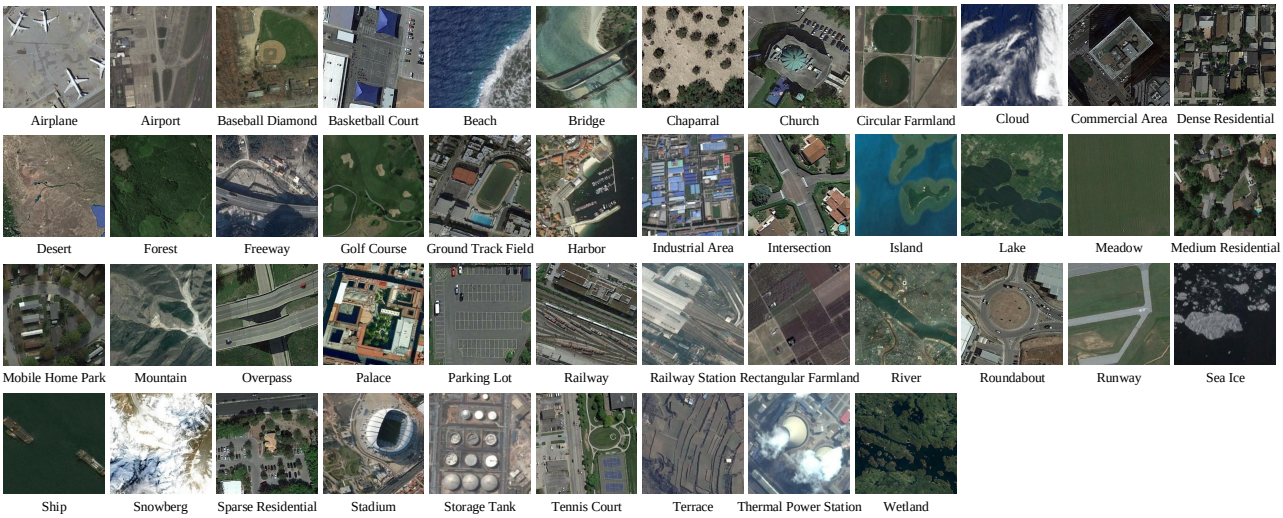

Figure 7. Some examples of the NWPU dataset.

4.1.2. Experimental Settings

We used the Adam optimizer to train our model for 100 iterations and resize all the input images into 224 × 224 on four RS scene classification datasets. We set the batch size to 32, the initial learning rate to 0.0001, and adopted a cosine decay learning rate scheduler with a linear warm-up. Random rotation and horizontal and vertical flipping were used for

data augmentation. All the experiments for RS scene classification were implemented on Pytorch [63] with NVIDIA TITAN Xp. In addition, following the experimental setup in [36], 50%, 20%, 10%, and 80% of the images in each scene category of the Merced, AID, NWPU, and Optimal-31 datasets were randomly selected to train our models, respectively. The experimental results are presented in overall accuracy (OA) to evaluate the performance of our models. We randomly sampled the dataset based on the same training sample proportion in [36]. All the experiments were run five times, and we report the mean and standard deviation of the overall accuracies (OAs).

4.1.3. Experimental Results

On four RS scene classification datasets, we fine-tuned the proposed MLGC-ResNet-50 and LTNet, which are initialized by the pretrained parameters training on the ImageNet dataset [47]. For the MLGC-ResNet-50 and LTNet, $g_1 = 2$ and $g_2 = 1$. Table 3 shows weights and FLOPs for common networks based on pretrained CNNs and ViTs and the OA of different RS scene classification methods based on these networks. The experimental results in Table 3 present the effectiveness of our models. Specifically, fine-tuning the pretrained MLGC-ResNet-50 achieves the best performance for the Merced dataset, by using similar or fewer computing resources than methods based on pretrained VGG-16 [60], AlexNet [64], Inception-v3 [65], ResNet-34 [62], Ghost-ResNet-50 [44], EfficientNet-B3 [66], ViT-L/16 [27], and ViT-B/16 [27]. Fine-tuning the LTNet achieves the second-best performance with lower computational complexity. Fine-tuning the pre-trained LTNet on the AID and Optimal31 datasets achieves the best performance.

Table 3. Comparison of experimental results (%) on the Merced, AID, Optimal-31, and NWPU datasets.

PreTrained Network	Weights	FLOPs	Method	Merced (50% Train)	AID (20% Train)	Optimal31 (80% Train)	NWPU (10% Train)
VGG-16 [60]	138.4 M	15.5 G	Fine-tuning [19]	96.57 ± 0.38	89.49 ± 0.34	89.52 ± 0.26	-
			Fine-tuning [15]	-	-	-	87.15 ± 0.45
			ARCNet-VGG16 [24]	96.81 ± 0.14	88.75 ± 0.40	92.70 ± 0.35	-
			GBNet + global feature [19]	97.05 ± 0.19	92.20 ± 0.23	93.28 ± 0.27	-
			+MSCP [20]	-	91.52 ± 0.21	-	85.33 ± 0.17
			ACNet [25]	-	93.33 ± 0.29	-	91.09 ± 0.13
AlexNet [64]	61.0 M	724 M	Fine-tuning [15]	-	-	-	81.22 ± 0.19
			ARCNet-AlexNet [24]	-	-	85.75 ± 0.35	-
			+MSCP [20]	-	88.99 ± 0.38	-	81.70 ± 0.23
Inception-v3 [65]	24 M	5.7 G	Inception-v3-aux [16]	97.63 ± 0.20	93.52 ± 0.21	94.13 ± 0.35	89.32 ± 0.33
ResNet-34 [62]	21.8 M	3.7 G	ARCNet-ResNet34 [24]	-	-	91.28 ± 0.45	-
Ghost-ResNet-50 [44]	13.9 M	2.0 G	Fine-tuning	98.25 ± 0.22	94.66±0.12	94.73 ± 0.58	91.79 ± 0.16
EfficientNet-B3 [66]	12 M	1.8 G	EfficientNet-B3-aux [16]	98.22 ± 0.49	94.19 ± 0.15	94.51 ± 0.75	91.08 ± 0.14
GoogLeNet [67]	6.7 M	1.5G	Fine-tuning [15]	-	-	-	82.57 ± 0.12
			GoogLeNet-aux [16]	97.90 ± 0.34	93.25 ± 0.33	93.11 ± 0.55	89.22 ± 0.25
EfficientNet-B0 [66]	5.3 M	0.4 G	EfficientNet-B0-aux [16]	98.01 ± 0.45	93.69 ± 0.11	93.97 ± 0.13	89.96 ± 0.27
ViT-L/16 [27]	304.4 M	61.6 G	Fine-tuning	98.24 ± 0.21	94.44 ± 0.26	94.89 ± 0.24	90.85 ± 0.16
ViT-B/16 [27]	86.6 M	17.6 G	Fine-tuning [36]	98.14 ± 0.47	94.97 ± 0.01	95.07 ± 0.12	92.60 ± 0.10
MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$)	13.1 M	1.9 G	Fine-tuning	98.48 ± 0.28	94.73 ± 0.15	95.27 ± 0.36	92.16 ± 0.08
LTNet ($g_1 = 2, g_2 = 1$)	8.2 M	1.7 G	Fine-tuning	98.36 ± 0.25	94.98 ± 0.08	95.70 ± 0.29	92.21 ± 0.11

Although the OA of the proposed method on the NWPU dataset is slightly lower than that of ViT-B/16, the weights, and FLOPs are significantly reduced. The parameters and FLOPs of ViT-L/16 are about 37 and 36 times that of LTNet, and the parameters and FLOPs of ViT-B/16 are about 11 and 10 times that of MLGC-ResNet-50, respectively. In addition, fine-tuning the ViT-L/16 and ViT-B/16 requires a longer time-frame than LTNet,

as shown in Table 4. While the pretrained networks GoogLeNet [67] and EfficientNet-B0 [66] are lower than LTNet in terms of parameters and FLOPs, their performance is significantly lower. Especially on Optimal31 and NWPU datasets, compared to the LTNet, the classification accuracy of using GoogLeNet as a pretraining network is decreased by 2.59% and 2.99%, respectively, and the classification accuracy of using EfficientNet as a pretraining network is reduced by 1.93% and 2.25%, respectively. The inference times of ViT-L/16, ViT-B/16, MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$), and LTNet ($g_1 = 2, g_2 = 1$) are shown in Figure 8. We can see that our method can achieve both lower FLOPs and less inference time.

Table 4. Training times on the Merced, AID, Optimal-31, and NWPU datasets.

PreTrained Network	Weights	FLOPs	Method	Merced (50% Train)	AID (20% Train)	Optimal31 (80% Train)	NWPU (10% Train)
ViT-L/16 [27]	304.4 M	61.6 G	Fine-tuning	70 min	234 min	80 min	629 min
ViT-B/16 [27]	86.6 M	17.6 G		40 min	132 min	45 min	336 min
MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$)	13.1 M	1.9 G		16 min	59 min	18 min	117 min
LTNet ($g_1 = 2, g_2 = 1$)	8.2 M	1.7 G		17 min	61 min	20 min	124 min

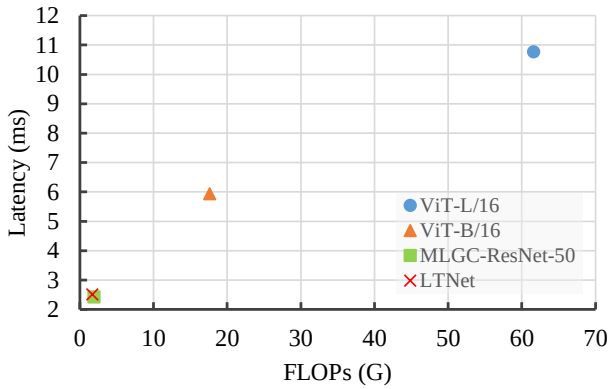

Figure 8. Inference time of ViT-L/16, ViT-B/16, MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$), and LTNet ($g_1 = 2, g_2 = 1$).

4.2. Experiments on Two Natural Image Classification Datasets

4.2.1. Dataset Description

We verified the effectiveness of the proposed MLGC module, LightFormer block, and designed LTNet on two image classification benchmarks, CIFAR-10 [61] and ImageNet ILSVRC 2012 [47] datasets. The CIFAR-10 dataset consists of 32 × 32 pixel RGB images corresponding to 10 classes, including 50,000 training and 10,000 test images. Random crops were used as a standard data augmentation scheme. The ImageNet dataset contains 1000 classes with 1.2 million training images and 50,000 validation images. Random crop and flip [62] were used as the data preprocessing strategy.

4.2.2. Experimental Settings

All experiments were implemented using the PyTorch [63] deep learning library and we used stochastic gradient descent (SGD) to optimize the module. The momentum was 0.9, and the weight decay was 0.0001. The mini-batch sizes for CIFAR-10 and ImageNet were 96 on 2 GPUs and 2048 on 8 GPUs, respectively. We utilized cosine shape learning rates for VGG-16-based and ResNet-56-based networks, starting at 0.1 with 400 trained epochs, and for ResNet-50-based networks we started at 0.8 and 300 epochs. In addition, according to [27], we used eight heads in lightweight multi-head attention.

4.2.3. Comparison Experiments on CIFAR-10 and ImageNet

We trained the proposed MLGC-VGG-16 and LightFormer-VGG-16 on the CIFAR-10 dataset. In Table 5, the proposed models achieve competitive performances with lower computational costs than the Ghost-VGG-16. In particular, the LightFormer-VGG-16 only has 0.8 M weights (about 1/10 of the Ghost-VGG-16 and about 1/4 of the CCT-6/3 × 2). In addition, for ResNet-56 we replaced the Ghost module in Ghost-ResNet-56 with the MLGC module to obtain MLGC-ResNet-56 and then plugged the LightForm block into MLGC-ResNet-56 to acquire LightForm-ResNet-56.The experimental comparison results are shown in Table 6. The LightFormer-ResNet-56 achieves a higher performance with fewer computations.

Table 5. Comparison experiments on CIFAR-10.

Model	Weights (M)	FLOPs (M)	Acc. (%)
MobileNetV1 [39]	3.2	47	92.5
MobileNetV2 [40]	2.3	68	93.2
ShuffleNetV1(g = 3) [42]	0.9	**45**	92.8
ShuffleNetV2 [43]	1.3	**45**	93.5
EfficientNet-B0 [66]	4.0	64	93.8
Ghost-VGG-16 [44]	7.7	160	93.5
GhostV2-VGG-16 [68]	9.4	188	93.6
CCT-6/3 × 2 [45]	3.3	241	93.6
CCT-4/3 × 2 [45]	0.5	46	91.5
MLGC-VGG-16 ($g_1 = 2, g_2 = 2$)	2.8	97	93.8
LightFormer-VGG-16 ($g_1 = 2, g_2 = 2$)	0.8	55	93.9

Table 6. Weights, FLOPs, and accuracy of replacing the corresponding layer in Ghost-ResNet-56 with MLGC module and LightFormer Block on CIFAR-10.

Model	Weights (M)	FLOPs (M)	Acc. (%)
Ghost-ResNet-56	0.44	67	92.7
MLGC-ResNet-56 ($g_1 = 2, g_2 = 1$)	0.43	65	93.0
LightFormer-ResNet-56 ($g_1 = 2, g_2 = 1$)	0.33	58	93.1

The ResNet-50, Ghost-ResNet-50, MLGC-ResNet-50, and LTNet network architectures for ImageNet are shown in Table 2. Our MLGC-ResNet-50 and LTNet have fewer weights and FLOPs than Ghost-VGG-16 [44]. As can be seen from the results in Table 7, the proposed MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$) and LTNet ($g_1 = 2, g_2 = 1$) achieve a lower number of parameters while improving the accuracy of the Ghost-ResNet-50. Compared with the advanced methods, such as Thinet [69], and the universal filter [70], our method can achieve a better performance. Observing the experiment on CIFAR-10 shows that $g_1 = 2$ works best. For ImageNet, a richer diversity of feature maps is needed, so $g_2 = 1$. Experiments on the ImageNet dataset demonstrate that the LTNet model with 8.2 M weights achieves better results than the Ghost-ResNet-50 model with 13.9 M weights. The best results came from MLGC-ResNet-56 (75.5%) and LightFormer-ResNet-56 (75.4%), but the latter needs less weight and FLOPs.

Table 7. Top-1 validation accuracy, weights, and FLOPs comparisons for compressing ResNet-50 on ImageNet dataset.

Model	Weights (M)	FLOPs (B)	Top-1 Acc. (%)
Thinet-ResNet-50 [69]	16.9	24.9	72.0
Versatile-ResNet-50 [70]	11.0	3.0	74.5
Ghost-ResNet-50 ($s = 2$) [44]	13.9	2.2	74.7
MLGC-ResNet-50 ($g_1 = 2, g_2 = 1$)	13.1	1.9	75.5
LTNet ($g_1 = 2, g_2 = 1$)	8.2	1.7	75.4

4.2.4. Ablation Experiments on CIFAR-10

We used the MLGC module to replace the Ghost module in the Ghost-VGG-16 to obtain the MLGC-VGG-16. The experimental comparison results of different s for the Ghost-VGG-16 and g_1 and g_2 for the MLGC-VGG-16 are shown in Table 8. It can be found that the overall results of the MLGC-VGG-16 with $g_1 = 2$ have a superior performance. In particular, the MLGC-VGG-16 obtained a higher accuracy when $g_1 = g_2 = 2$, with fewer weights and FLOPs than the Ghost-VGG-16 ($s = 2$). The weights/FLOPs decreased from Ghost-VGG-16 with 7.7 M/160 M to the MLGC-VGG-16 with 6.0 M/127 M, proving the MLGC module's effectiveness. In addition, we observe that there is a significant drop in accuracy particularly when g_1 is high (16 or 32) and g_2 is low (1). This is because g_1 is high (16 or 32), which weakens the representation ability of first-level features and further affects the learning of the second-level features. Therefore, even if g_2 is low (1) and the network has higher FLOPs it still has a lower performance. For Ghost-VGG-16 (s = 4), its output feature maps are divided into four parts. The first part is learned from all input feature maps through ordinary convolution. The feature representation ability is strong. Based on this, the other three parts of the output feature map can learn better feature representation.

Table 8. Weights, FLOPs, and accuracy of Ghost module with different s and the proposed MLGC module with different g_1 and g_2 on CIFAR-10. Split the output channels of the ordinary convolutional layer into s parts of each Ghost module, and g_1 and g_2 are the number of groups of two group convolutions in the MLGC module.

Model	s	g_1	g_2	Weights (M)	FLOPs (M)	Acc. (%)
Ghost-VGG-16	2	-	-	7.7	160	93.5
MLGC- VGG-16	-	2	1	8.0	173	94.2
		4	1	6.2	134	93.4
		2	2	6.0	127	93.6
		8	1	5.3	115	92.7
		2	4	5.0	104	93.5
Ghost-VGG-16	3	-	-	5.2	109	92.8
MLGC- VGG-16	-	16	1	4.8	105	90.8
		32	1	4.6	100	90.4
		2	8	4.5	92	93.1
		2	16	4.2	87	93.1
		4	2	4.1	88	92.9
		2	32	4.1	84	93.0
Ghost-VGG-16	4	-	-	4.0	82	92.6

As seen from Table 9, to acquire LightFormer-VGG-16, we replaced the MLGC module with the LightFormer block from deep to shallow in MLGC-VGG-16. v8–v18 are the layer names for the VGG-16, Ghost-VGG-16, MLGC-VGG-16, and LightFormer-VGG-16 (see the first column in Table 1). The best performance was achieved when we replaced MLGC modules of v8–v13 layers in MLGC-VGG-16 with the LightFormer block. The weights/FLOPs decreased from the MLGC-VGG-16 with 6.0 M/127 M to the LightFormer-VGG-16 with 2.80 M/97 M with similar accuracies. Referring to the inflection point in Figure 9, our method can achieve an accuracy of 93.8% with the FLOPs of 97 M.

In CNNs, the receptive field of the feature maps will increase as the networks get deeper. This can be beneficial to CNNs. For example, Dilated/Atrous Convolution [71,72] is proposed to increase the receptive field. Considering that the LightFormer block has captured long-range dependencies, can we remove some layers in LightFormer-VGG-16 to reduce the weights and FLOPs of the network? In Table 10, we removed layers from the deep to the shallow of the LightFormer-VGG-16 (layer v8–v13 with the LightFormer block). First, the accuracy of removing v11–v13 or v10–v13 is similar to v9–v13, so v11–v13 and v10–v13 are not shown in Table 10 for simplicity. Only one LightFormer block (v8) used in the LightFormer-VGG-16 can be obtained with a great performance. We continued to remove the layers before v8 in the network. When we removed v6–v7&v9–v13, a higher

performance was maintained at a lower computational expense. If the number of layers continues to be reduced, the accuracy will decrease. The experimental results show that the final architecture of LightFormer-VGG-16 is the last column in Table 1. LightFormer-VGG-16 uses four MLGC modules and one LightFormer block. It shows that the designed network only needs to put a LightFormer block on the deep layer of the network to achieve an outstanding performance. Referring to the inflection point in Figure 10, our method can achieve an accuracy of 93.9% with the FLOPs of 55 M.

Table 9. Weights, FLOPs, and accuracy of replacing the corresponding layer in MLGC-VGG-16 (g_1 = 2 and g_2 = 2) with the LightFormer Block.

Model	MLGC-VGG-16 Layer Name	Weights (M)	FLOPs (M)	Acc. (%)
LightFormer-VGG-16 ($g_1 = 2, g_2 = 2$)	-	6.0	127	93.6
	v11	5.4	125	93.7
	v12	5.4	125	93.6
	v13	5.4	125	93.7
	v11–v13	4.3	120	93.8
	v8–v13	2.8	97	93.8
	v7–v13	2.7	88	93.3
	v6–v13	2.5	79	92.9

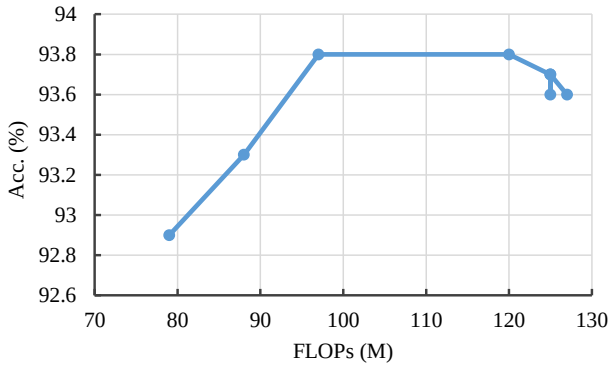

Figure 9. Accuracy vs. FLOPs of MLGC-VGG-16 ($g_1 = 2, g_2 = 2$) with the LightFormer Block.

Table 10. Removed layers in LightFormer-VGG-16 (v8–v13).

Model	Removed Layer Name	Weights (M)	FLOPs (M)	Acc. (%)
LightFormer-VGG-16 ($g_1 = 2, g_2 = 2$)	-	2.8	97	93.8
	v9–v13	1.2	83	93.8
	v7&v9–v13	1.0	69	93.8
	v6–v7&v9–v13	0.8	55	93.9
	v4&v6–v7&v9–v13	0.8	40	92.9
	v2&v4&v6–v7&v9–v13	0.7	26	92.1

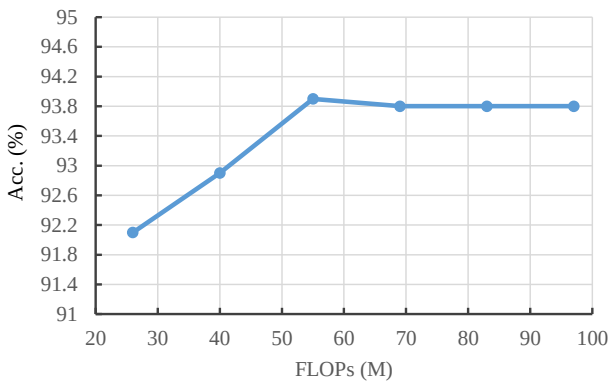

Figure 10. Accuracy vs. FLOPs of LightFormer-VGG-16 (v8–v13).

As seen from the experimental results in Table 11, whether position embedding is added has little influence on the model accuracy. The MLGC module already contains position information, so we removed the position embedding in the LightFormer block.

Table 11. Impact of position embedding on performance.

Model	Position Embedding	Weights (M)	FLOPs (M)	Acc. (%)
LightFormer-VGG-16 ($g_1 = 2, g_2 = 2$)	Yes	0.8	55	93.8
	No			93.9

We further demonstrated that our lightweight LightFormer-VGG-16 is also suitable for small datasets, which is very important for research in the field with limited datasets [45]. The experiments in Table 12 demonstrate that the LightFormer-VGG-16/0.5 can have as little as 0.2 M weights to achieve a better result (improve 1.9%) than the CCT-2/3 × 2 [45] with 0.3 M weights on CIFAR-10. The LightFormer-VGG-16/0.5 means that the number of channels for all output feature maps in each layer is reduced by half. Considering that multi-head attention in CCT-2/3 × 2 uses four heads, we attempted to use four heads in our method and obtained similar results to eight heads. In addition, the positional encoder had little effect. We reduced the amount of training data on CIFAR-10 for LightFormer-VGG-16/0.5 to measure whether the proposed method is a data-hungry [45] model. In Figure 11, we compare the accuracy of the model on the number of samples (500, 1000, 2000, 3000, 4000, or 5000) per class. Experiments show that our model is more robust and accurate on very small datasets compared to CCT-2/3 × 2.

Table 12. Impact of number of heads and position embedding on performance.

Model	Heads	Positional Embedding	Weights (M)	Acc. (%)
MobileNetV2/0.5[1]	-	-	0.7	84.8
CCT-2/3 × 2[2]	4	Yes	0.3	88.9
LightFormer-VGG-16/0.5 ($g_1 = 2, g_2 = 2$)	8	Yes	0.2	90.8
	4	Yes		90.7
	4	No		90.7

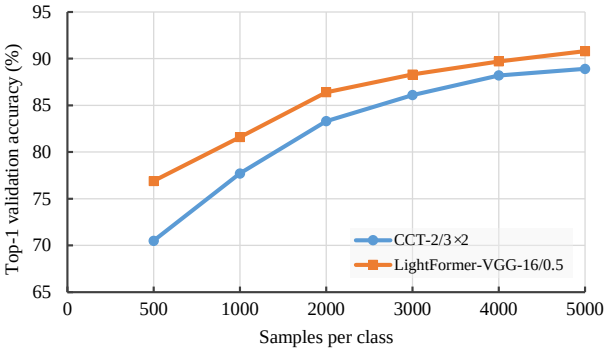

Figure 11. Results on CIFAR-10 with the reduced number of samples per class.

5. Conclusions

In this paper, we proposed a lightweight transformer network (LTNet) for RS scene classification. First, we presented a simple and efficient MLGC module with the ability of multi-level feature representation with a lower computational cost. In the MLGC module, not only is the redundancy of both input and output feature maps considered to reduce computing resources but the diversity of the output features is also maintained. Second, based on the strong ability of the feature representation of the MLGC module, we designed a lightweight transformer block named LightFormer without MLP layers. Finally, we built an efficient LTNet using the MLGC module and LightFormer block. In the LTNet, richer

local features extracted by the stacked MLGC modules are fed into one LightFormer block to capture long-range dependencies. Experiments on four common RS scene classification datasets demonstrated the efficacy of our LTNet.

Author Contributions: Conceptualization, F.L. and X.H.; methodology, X.H.; software, X.H. and Y.C.; validation, F.L. and X.H.; formal analysis, P.L. and X.H.; investigation, L.L.; resources, P.C.; writing—original draft preparation, X.H. and Y.C.; writing—review and editing, X.H., Y.C. and F.L.; supervision, F.L.; funding acquisition, F.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (No. 62076192), the State Key Program of National Natural Science of China (No. 61836009), in part by the Program for Cheung Kong Scholars and Innovative Research Team in University (No. IRT_15R53), in part by the Fund for Foreign Scholars in University Research and Teaching Programs (the 111 Project) (No. B07048), in part by the Key Scientific Technological Innovation Research Project by Ministry of Education, and the CAAI Huawei MindSpore Open Fund.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xiao, Y.; Zhan, Q. A review of remote sensing applications in urban planning and management in China. In Proceedings of the 2009 Joint Urban Remote Sensing Event, Shanghai, China, 20–22 May 2009 ; IEEE: Piscataway Township, NJ, USA, 2009; pp. 1–5.
2. Martha, T.R.; Kerle, N.; Van Westen, C.J.; Jetten, V.; Kumar, K.V. Segment optimization and data-driven thresholding for knowledge-based landslide detection by object-based image analysis. *IEEE Trans. Geosci. Remote. Sens.* **2011**, *49*, 4928–4943. [CrossRef]
3. Stumpf, A.; Kerle, N. Object-oriented mapping of landslides using Random Forests. *Remote. Sens. Environ.* **2011**, *115*, 2564–2577. [CrossRef]
4. Cheng, G.; Guo, L.; Zhao, T.; Han, J.; Li, H.; Fang, J. Automatic landslide detection from remote-sensing imagery using a scene classification method based on BoVW and pLSA. *Int. J. Remote. Sens.* **2013**, *34*, 45–59. [CrossRef]
5. Tong, X.Y.; Xia, G.S.; Lu, Q.; Shen, H.; Li, S.; You, S.; Zhang, L. Land-cover classification with high-resolution remote sensing images using transferable deep models. *Remote. Sens. Environ.* **2020**, *237*, 111322. [CrossRef]
6. Li, Y.; Zhang, Y.; Tao, C.; Zhu, H. Content-based high-resolution remote sensing image retrieval via unsupervised feature learning and collaborative affinity metric fusion. *Remote Sens.* **2016**, *8*, 709. [CrossRef]
7. Du, Z.; Li, X.; Lu, X. Local structure learning in high resolution remote sensing image retrieval. *Neurocomputing* **2016**, *207*, 813–822. [CrossRef]
8. Duan, Y.; Liu, F.; Jiao, L.; Zhao, P.; Zhang, L. SAR image segmentation based on convolutional-wavelet neural network and Markov random field. *Pattern Recognit.* **2017**, *64*, 255–267. [CrossRef]
9. Jiao, L.; Zhang, S.; Li, L.; Liu, F.; Ma, W. A modified convolutional neural network for face sketch synthesis. *Pattern Recognit.* **2018**, *76*, 125–136. [CrossRef]
10. Li, L.; Ma, L.; Jiao, L.; Liu, F.; Sun, Q.; Zhao, J. Complex Contourlet-CNN for polarimetric SAR image classification. *Pattern Recognit.* **2020**, *100*, 107110. [CrossRef]
11. Wang, J.; Duan, Y.; Tao, X.; Xu, M.; Lu, J. Semantic perceptual image compression with a laplacian pyramid of convolutional networks. *IEEE Trans. Image Process.* **2021**, *30*, 4225–4237. [CrossRef]
12. Singh, P.; Mazumder, P.; Namboodiri, V.P. Context extraction module for deep convolutional neural networks. *Pattern Recognit.* **2022**, *122*, 108284. [CrossRef]
13. Cui, Y.; Liu, F.; Jiao, L.; Guo, Y.; Liang, X.; Li, L.; Yang, S.; Qian, X. Polarimetric multipath convolutional neural network for PolSAR image classification. *IEEE Trans. Geosci. Remote. Sens.* **2021**, *60*, 1–18. [CrossRef]
14. Nogueira, K.; Penatti, O.A.; Dos Santos, J.A. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognit.* **2017**, *61*, 539–556. [CrossRef]
15. Cheng, G.; Han, J.; Lu, X. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE* **2017**, *105*, 1865–1883. [CrossRef]
16. Bazi, Y.; Al Rahhal, M.M.; Alhichri, H.; Alajlan, N. Simple yet effective fine-tuning of deep CNNs using an auxiliary classification loss for remote sensing scene classification. *Remote Sens.* **2019**, *11*, 2908. [CrossRef]
17. Li, W.; Wang, Z.; Wang, Y.; Wu, J.; Wang, J.; Jia, Y.; Gui, G. Classification of high-spatial-resolution remote sensing scenes method using transfer learning and deep convolutional neural network. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2020**, *13*, 1986–1995. [CrossRef]
18. Lu, X.; Sun, H.; Zheng, X. A feature aggregation convolutional neural network for remote sensing scene classification. *IEEE Trans. Geosci. Remote. Sens.* **2019**, *57*, 7894–7906. [CrossRef]

19. Sun, H.; Li, S.; Zheng, X.; Lu, X. Remote sensing scene classification by gated bidirectional network. *IEEE Trans. Geosci. Remote. Sens.* **2019**, *58*, 82–96. [CrossRef]
20. He, N.; Fang, L.; Li, S.; Plaza, A.; Plaza, J. Remote sensing scene classification using multilayer stacked covariance pooling. *IEEE Trans. Geosci. Remote. Sens.* **2018**, *56*, 6899–6910. [CrossRef]
21. Liu, Y.; Liu, Y.; Ding, L. Scene classification based on two-stage deep feature fusion. *IEEE Geosci. Remote. Sens. Lett.* **2017**, *15*, 183–186. [CrossRef]
22. Xue, W.; Dai, X.; Liu, L. Remote sensing scene classification based on multi-structure deep features fusion. *IEEE Access* **2020**, *8*, 28746–28755. [CrossRef]
23. Wang, X.; Wang, S.; Ning, C.; Zhou, H. Enhanced feature pyramid network with deep semantic embedding for remote sensing scene classification. *IEEE Trans. Geosci. Remote. Sens.* **2021**, *59*, 7918–7932. [CrossRef]
24. Wang, Q.; Liu, S.; Chanussot, J.; Li, X. Scene classification with recurrent attention of VHR remote sensing images. *IEEE Trans. Geosci. Remote. Sens.* **2019**, *57*, 1155–1167. [CrossRef]
25. Tang, X.; Ma, Q.; Zhang, X.; Liu, F.; Ma, J.; Jiao, L. Attention consistent network for remote sensing scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2021**, *14*, 2030–2045. [CrossRef]
26. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS), Long Beach, CA, USA, 4–9 December 2017.
27. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Vienna, Austria, 4 May 2021.
28. 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), Montreal, BC, Canada, 11–17 October 2021; pp. 22–31.
29. Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.H.; Tay, F.E.; Feng, J.; Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 558–567.
30. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2020; pp. 213–229.
31. Sun, Z.; Cao, S.; Yang, Y.; Kitani, K.M. Rethinking transformer-based set prediction for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 3611–3620.
32. Chen, H.; Wang, Y.; Guo, T.; Xu, C.; Deng, Y.; Liu, Z.; Ma, S.; Xu, C.; Xu, C.; Gao, W. Pre-trained image processing transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 12299–12310.
33. Wang, Y.; Xu, Z.; Wang, X.; Shen, C.; Cheng, B.; Shen, H.; Xia, H. End-to-end video instance segmentation with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 8741–8750.
34. Zheng, S.; Lu, J.; Zhao, H.; Zhu, X.; Luo, Z.; Wang, Y.; Fu, Y.; Feng, J.; Xiang, T.; Torr, P.H.; et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 6881–6890.
35. Li, S.; Liu, F.; Jiao, L. Self-Training Multi-Sequence Learning with Transformer for Weakly Supervised Video Anomaly Detection. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), Virtual Event, 22 February–1 March 2022 .
36. Bazi, Y.; Bashmal, L.; Rahhal, M.M.A.; Dayil, R.A.; Ajlan, N.A. Vision transformers for remote sensing image classification. *Remote Sens.* **2021**, *13*, 516. [CrossRef]
37. Ma, J.; Li, M.; Tang, X.; Zhang, X.; Liu, F.; Jiao, L. Homo–heterogenous transformer learning framework for RS scene classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.* **2022**, *15*, 2223–2239. [CrossRef]
38. Srinivas, A.; Lin, T.Y.; Parmar, N.; Shlens, J.; Abbeel, P.; Vaswani, A. Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 16519–16529.
39. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861
40. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
41. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
42. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 6848–6856.

43. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
44. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 1580–1589.
45. Hassani, A.; Walton, S.; Shah, N.; Abuduweili, A.; Li, J.; Shi, H. Escaping the Big Data Paradigm with Compact Transformers. *arXiv* **2021**, arXiv:2104.05704.
46. Graham, B.; El-Nouby, A.; Touvron, H.; Stock, P.; Joulin, A.; Jégou, H.; Douze, M. LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 12259–12269.
47. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; IEEE: Piscataway Township, NJ, USA, 2009; pp. 248–255.
48. Yang, Y.; Newsam, S. Bag-of-visual-words and spatial extensions for land-use classification. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; pp. 270–279.
49. Xia, G.S.; Hu, J.; Hu, F.; Shi, B.; Bai, X.; Zhong, Y.; Zhang, L.; Lu, X. AID: A benchmark data set for performance evaluation of aerial scene classification. *IEEE Trans. Geosci. Remote. Sens.* **2017**, *55*, 3965–3981. [CrossRef]
50. Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; Jégou, H. Training data-efficient image transformers & distillation through attention. In Proceedings of the International Conference on Machine Learning (ICML), Virtual Event, 18–24 July 2021; pp. 10347–10357.
51. Chen, C.F.R.; Fan, Q.; Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 357–366.
52. Dai, Z.; Cai, B.; Lin, Y.; Chen, J. Up-detr: Unsupervised pre-training for object detection with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 20–25 June 2021; pp. 1601–1610.
53. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual Event, 3–7 May 2021.
54. Mehta, S.; Rastegari, M. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In Proceedings of the 10th International Conference on Learning Representations (ICLR), Virtual Event, 25–29 April 2022.
55. He, Z.; Yuan, Z.; An, P.; Zhao, J.; Du, B. MFB-LANN: A lightweight and updatable myocardial infarction diagnosis system based on convolutional neural networks and active learning. *Comput. Methods Programs Biomed.* **2021**, *210*, 106379. [CrossRef]
56. Jiang, X.; Wang, N.; Xin, J.; Xia, X.; Yang, X.; Gao, X. Learning lightweight super-resolution networks with weight pruning. *Neural Networks* **2021**, *144*, 21–32. [CrossRef]
57. Qian, X.; Liu, F.; Jiao, L.; Zhang, X.; Guo, Y.; Liu, X.; Cui, Y. Ridgelet-Nets With Speckle Reduction Regularization for SAR Image Scene Classification. *IEEE Trans. Geosci. Remote. Sens.* **2021**, *59*, 9290–9306. [CrossRef]
58. Ma, H.; Yang, S.; Feng, D.; Jiao, L.; Zhang, L. Progressive Mimic Learning: A new perspective to train lightweight CNN models. *Neurocomputing* **2021**, *456*, 220–231. [CrossRef]
59. Ioannou, Y.; Robertson, D.; Cipolla, R.; Criminisi, A. Deep roots: Improving cnn efficiency with hierarchical filter groups. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1231–1240.
60. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
61. Krizhevsky, A.; Hinton, G. *Learning Multiple Layers of Features from Tiny Images*; Technical Report; University of Toronto: Toronto, ON, Canada, 2009 .
62. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
63. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. In Proceedings of the Advances in Neural Information Processing Systems 32 (NIPS), Vancouver, BC, Canada, 8–14 December 2019.
64. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems 25 (NIPS), Lake Tahoe, NV, USA, 3–6 December 2012 .
65. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
66. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning (ICML). PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
67. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.

68. Tang, Y.; Han, K.; Guo, J.; Xu, C.; Xu, C.; Wang, Y. GhostNetv2: enhance cheap operation with long-range attention. *Adv. Neural Inf. Process. Syst. (NIPS)* **2022**, *35*, 9969–9982.
69. Luo, J.H.; Wu, J.; Lin, W. Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5058–5066.
70. Wang, Y.; Xu, C.; Xu, C.; Xu, C.; Tao, D. Learning versatile filters for efficient convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems 31 (NIPS), 3–8 December 2018, Montréal, QC, Canada.
71. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Semantic image segmentation with deep convolutional nets and fully connected crfs. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
72. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 834–848. [CrossRef]