Hindawi
Complexity
Volume 2021, Article ID 6663851, 11 pages
https://doi.org/10.1155/2021/6663851

Research Article

Multiscale Feature Filtering Network for Image Recognition System in Unmanned Aerial Vehicle

Xianghua Ma, Zhenkun Yang, and Shining Chen

School of Electrical and Electronic Engineering, Shanghai Institute of Technology, Shanghai 201418, China

Correspondence should be addressed to Xianghua Ma; xhuam@sit.edu.cn

Received 19 November 2020; Revised 28 December 2020; Accepted 3 February 2021; Published 19 February 2021

Academic Editor: Rui Wang

For unmanned aerial vehicle (UAV), object detection at different scales is an important component for the visual recognition. Recent advances in convolutional neural networks (CNNs) have demonstrated that attention mechanism remarkably enhances multiscale representation of CNNs. However, most existing multiscale feature representation methods simply employ several attention blocks in the attention mechanism to adaptively recalibrate the feature response, which overlooks the context information at a multiscale level. To solve this problem, a multiscale feature filtering network (MFFNet) is proposed in this paper for image recognition system in the UAV. A novel building block, namely, multiscale feature filtering (MFF) module, is proposed for ResNet-like backbones and it allows feature-selective learning for multiscale context information across multiparallel branches. These branches employ multiple atrous convolutions at different scales, respectively, and further adaptively generate channel-wise feature responses by emphasizing channel-wise dependencies. Experimental results on CIFAR100 and Tiny ImageNet datasets reflect that the MFFNet achieves very competitive results in comparison with previous baseline models. Further ablation experiments verify that the MFFNet can achieve consistent performance gains in image classification and object detection tasks.

1. Introduction

To understand the environment, unmanned aerial vehicles (UAVs) need to integrate information from various sensors such as cameras, lidar, radar, and GPS. The information from the camera provides a straightforward way of visual perception, which supports further advanced thinking and reasoning for UAV. One of the important tasks in the visual perception of UAV, image recognition [1, 2] has always been a research hotspot. Convolutional neural networks (CNNs) have been widely used in solving visual cognition tasks, such as image classification [3, 4], object detection [5], and salient object detection [6]. Unlike traditional hand-crafted features (e.g., HOG [7]), features learned by CNNs based on data require minimal human involvement during training. Thus, most of the recent research on visual recognition is based on network engineering. It is becoming increasingly important to design better CNN architectures for visual recognition tasks.

Generally, for the design criterion of convolution networks, there are three important issues: depth, width, and cardinality. In 2015, Simonyan and Zisserman [8] designed an effective and very deep network by stacking blocks of the same shape, which achieved the state-of-the-art performance. However, as CNNs become increasingly deeper, gradient propagation becomes more difficult. In order to alleviate the problem of gradient disappearance caused by the increase of network depth, He et al. [9] proposed a deep residual learning approach, which referred to the input of the layer to learn the residual function. Experiments showed that this residual learning method can be easily optimized and can obtain higher accuracy by increasing the depth. Szegedy et al. [10] showed that width was another important factor to improve the performance of CNNs. Compared with shallower and less extensive networks, the main advantage of this method was that it can significantly improve the accuracy with a moderate increase in computing demand. ResNeXt [11] employed the potential of grouped convolutions and empirically showed that increasing cardinality was more effective than going deeper or wider as capacity increases. In 2016, Zagoruyko and Komodakis [12]

demonstrated that using more channels and a wider convolution can improve detection accuracy. Then, Huang et al. [13] proposed a dense convolutional network, which utilized direct connections between any two layers with the same feature map size to strengthen feature propagation. Ding et al. [14] designed a novel convolutional network, which used asymmetric convolutions to strengthen the square convolution filters.

Other network studies [15–18] exploited the potential of network from attention mechanism. For example, Hu et al. [15] designed a novel squeeze and excitation (SE) block that adaptively recalibrates channel-wise feature responses by emphasizing the interdependent channel maps. After that, Woo et al. [16] introduced a simple attention module called CBAM, which exploited both spatial and channel-wise attention to emphasize meaningful features along channel and spatial axes. Li et al. [17] proposed selective kernel networks (SKNets), which realized the adaptive receptive field sizes of neurons in a nonlinear approach. Furthermore, previous work [18] captured multiscale features through the additive effects of feature-selective and spatial attention. Targets appear at different scales in the image frame and are often occluded by clutter, which is a major challenge for image recognition algorithms in UAV applications. Therefore, multiscale feature representation is particularly critical for image recognition system in the UAV. However, most existing multiscale feature representation methods using attention mechanism simply employ several attention blocks to adaptively recalibrate the feature response, which overlooks the context information at a multiscale level.

Based on this analysis, in this paper, a multiscale feature filtering network (MFFNet) is proposed for image recognition system in the UAV. In MFFNet, we propose a novel building block, called multiscale feature filtering (MFF) module. Our key idea is to retain important information about smaller and insignificant objects by allowing feature-selective learning for multiscale context information across multiparallel branches. These branches employ multiple atrous convolutions at different scales, respectively, and further adaptively generate channel-wise feature responses by emphasizing channel-wise dependencies.

It is possible to construct an MFF network (MFFNet) by simply replacing the standard 3×3 filters in ResNet-like backbones with MFF modules. Besides, while the template for the MFF module is generic, the role it performs varies at different depths throughout the MFFNet. To compare the difference between the MFF module and standard 3×3 filter, we visualize the class activation mapping using Grad-CAM [19] and observe that the MFFNet-based CAM results tend to focus on the whole object more than other baseline networks. Experimental results on CIFAR [20], Tiny ImageNet [21], PASCAL VOC 2007 [22], MS COCO [23], and UAV123 [24] datasets show that our proposed method can achieve consistent performance gains in image recognition tasks.

The rest of the paper is organized as follows: Section 2 introduces our proposed MFFNet and presents the details of multiscale feature filtering (MFF) module. Section 3 shows experimental settings and analyses experimental results. Section 4 concludes this study and describes future work of this paper.

2. Method

In this section, the MFFNet, a novel backbone network for image recognition system in the UAV, is introduced. An overview of MFFNet is depicted in Figure 1. A MARNet contains four stages, and each stage contains multiple MFF units. Each MFF unit consists of a sequence of a 1×1 convolution, an MFF module, a 1×1 convolution, and a further skip layer. Figure 2 shows the schema of an MFF unit.

Furthermore, we present the details of multiscale feature filtering (MFF) module. The MFF module consists of three submodules: split module (SM), multiscale branch module (MBM), and fusion module (FM).

2.1. MFFNet Architecture. MFF modules can be integrated into a standard architecture, such as ResNet [9], by replacing every 3×3 layer with MFF modules. Here, MFF modules are used with MFF units. By making this change to each such module in the MFF unit, an MFFNet network can be constructed. Further variants that integrate MFF modules with ResNeXt [11], DenseNet [13], ShuffleNetV2 [25], and MobileNetV2 [26] can be constructed by following similar schemes. Like ResNet-50 and ResNeXt-50, MFFNet-50 and MFFNeXt-50 can be constructed by simply stacking a set of MFF units. MFFNet-50 can be obtained by changing the number of MFF units per stage. MFFNeXt-50 can be obtained from MFFNet-50 by changing the bottleneck width [12] and cardinality [11] of the MFF units. The cardinality, c, is the number of groups within a filter, whereas the bottleneck width, d, is the number of channels in a layer.

Table 1 shows the MFFNet-50 and MFFNeXt-50 architectures with four phases, using $\{3, 4, 6, 3\}$ MFF units. The filter sizes and feature dimensionalities of a residual block are shown inside the brackets. The number of stacked blocks for each stage is shown outside the brackets. "B = 3" denotes an MFF module with three branches, and "$c = 32$" suggests grouped convolutions with 32 groups.

2.2. Multiscale Feature Filtering Module. The structure of the MFF module is illustrated in Figure 3. First, in MFF module, given an input feature map, to obtain fine-grained multiscale information, the SM divides the input feature map into multiple feature map subsets. Second, to capture the objects at different scales, the MBM employs multiple atrous convolutions with different rates. Meanwhile, these branches use atrous convolutions instead of standard convolutions to reduce the model's parameters. Besides, the MBM further selectively generates channel-wise feature responses by emphasizing channel-wise dependencies. Once channel-wise feature responses with different scales are captured, the transformed features are connected by skip structures to enhance feature propagation. Third, a channel concatenation operator is applied to fuse previously captured information from different branches.

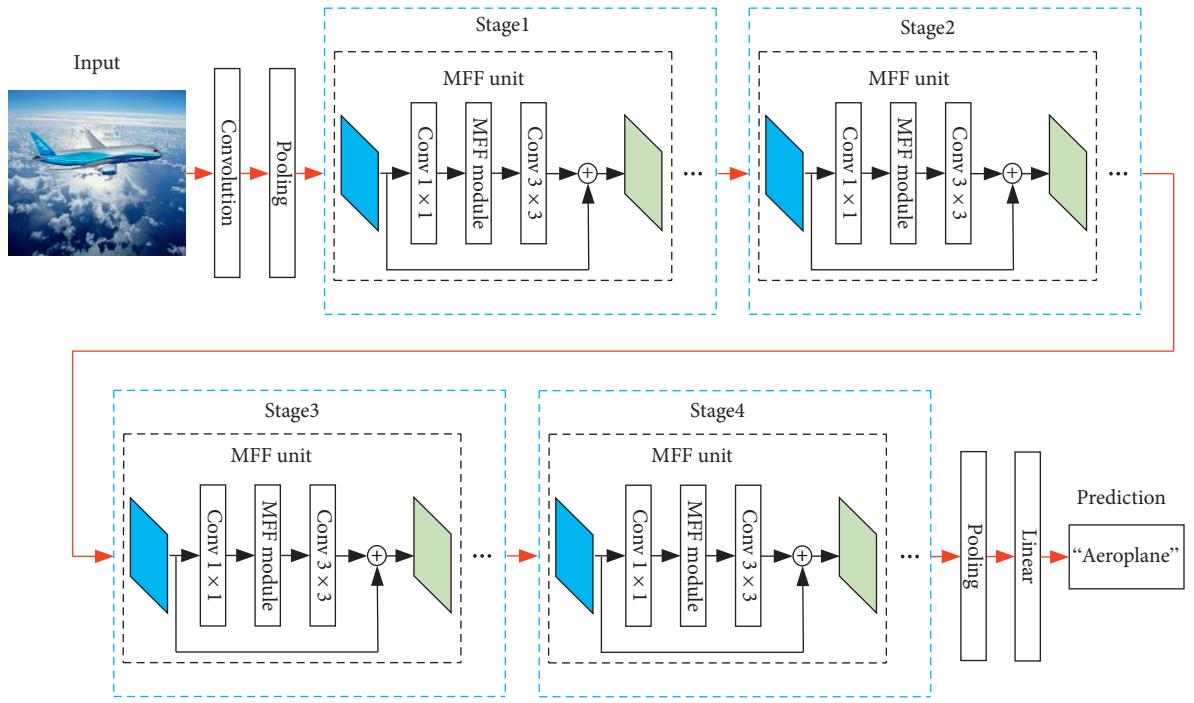

Figure 1: An MFFNet with four stages.

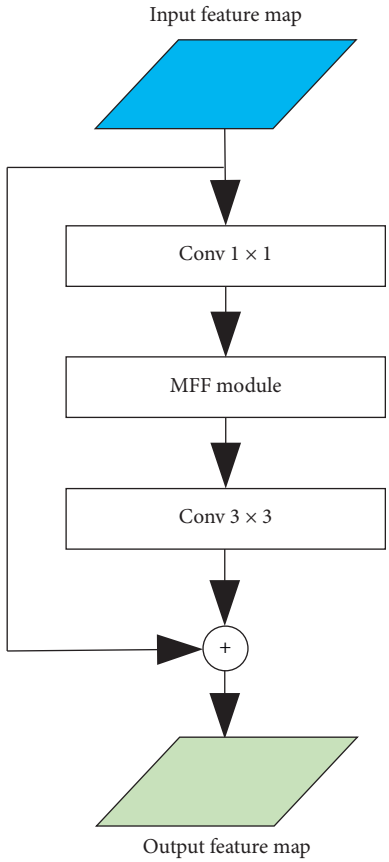

Figure 2: An MFF unit.

2.2.1. Split Module. As shown in Figure 1, in split module (SM), for any given input feature map $X \in \mathbb{R}^{H\times W\times C'}$, where $X = [X^1, X^2, \ldots, X^{C'}]$, to obtain fine-grained multiscale information, the SM first equally splits X into n feature map subsets, such as the three feature map subsets shown in Figure 1, namely, $X_1 \in \mathbb{R}^{H\times W\times C}$, $X_2 \in \mathbb{R}^{H\times W\times C}$, and $X_3 \in \mathbb{R}^{H\times W\times C}$, where $C = (C'/3)$. H, W, and C denote the height, width, and number of channels of the feature map, respectively.

2.2.2. Multiscale Branch Module. The multiscale branch module (MBM) consists of three branches, namely, A-branch, B-branch, and C-branch. Moreover, each branch contains a feature filtering module (FFM). The structure of the feature filtering module (FFM) is depicted in Figure 4.

$$m^c = F_{gm}(U^c) = \max\left[\sum_{i=1}^{H}\sum_{j=1}^{W} U^c(i,j)\right], \tag{1}$$

In FFM, we selectively generate channel-wise feature responses by emphasizing channel-wise dependencies. Specifically, for the preprocessed feature map $U \in \mathbb{R}^{H\times W\times C}$, firstly an FFM uses global average pooling and global max pooling to generate two different channel-wise statistics as $n \in \mathbb{R}^C$ and $m \in \mathbb{R}^C$. The global average pooling and global max pooling operations are denoted as F_{ga} and F_{gm}. Specifically, the c-th element of n and m is calculated as

$$n^c = F_{ga}(U^c) = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} U^c(i,j), \tag{2}$$

Table 1: The third and fourth columns show the architecture of MFFNet-50 and MFFNeXt-50, respectively.

Layers	Output size	MFFNet-50	MFFNeXt-50 (32 × 4d)
Convolution	112 × 112	Conv 7 × 7, 64, stride 2	
Pooling	56 × 56	Max pool 3 × 3, stride 2	
Stage 1	56 × 56	$\begin{bmatrix} 1\times1, & 64 \\ \mathrm{MFF}[b=3], & 64 \\ 1\times1, & 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 128 \\ \mathrm{MFF}[b=3,c=32], & 128 \\ 1\times1, & 256 \end{bmatrix}\times3$
Stage 2	28 × 28	$\begin{bmatrix} 1\times1, & 128 \\ \mathrm{MFF}[b=3], & 128 \\ 1\times1, & 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, & 256 \\ \mathrm{MFF}[b=3,c=32], & 256 \\ 1\times1, & 512 \end{bmatrix}\times4$
Stage 3	14 × 14	$\begin{bmatrix} 1\times1, & 256 \\ \mathrm{MFF}[b=3], & 256 \\ 1\times1, & 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, & 512 \\ \mathrm{MFF}[b=3,c=32], & 512 \\ 1\times1, & 1024 \end{bmatrix}\times6$
Stage 4	7 × 7	$\begin{bmatrix} 1\times1, & 512 \\ \mathrm{MFF}[b=3], & 512 \\ 1\times1, & 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 1024 \\ \mathrm{MFF}[b=3,c=32], & 1024 \\ 1\times1, & 2048 \end{bmatrix}\times3$
Classification layer	1 × 1	Global average pool 1000 d fully connected, softmax	

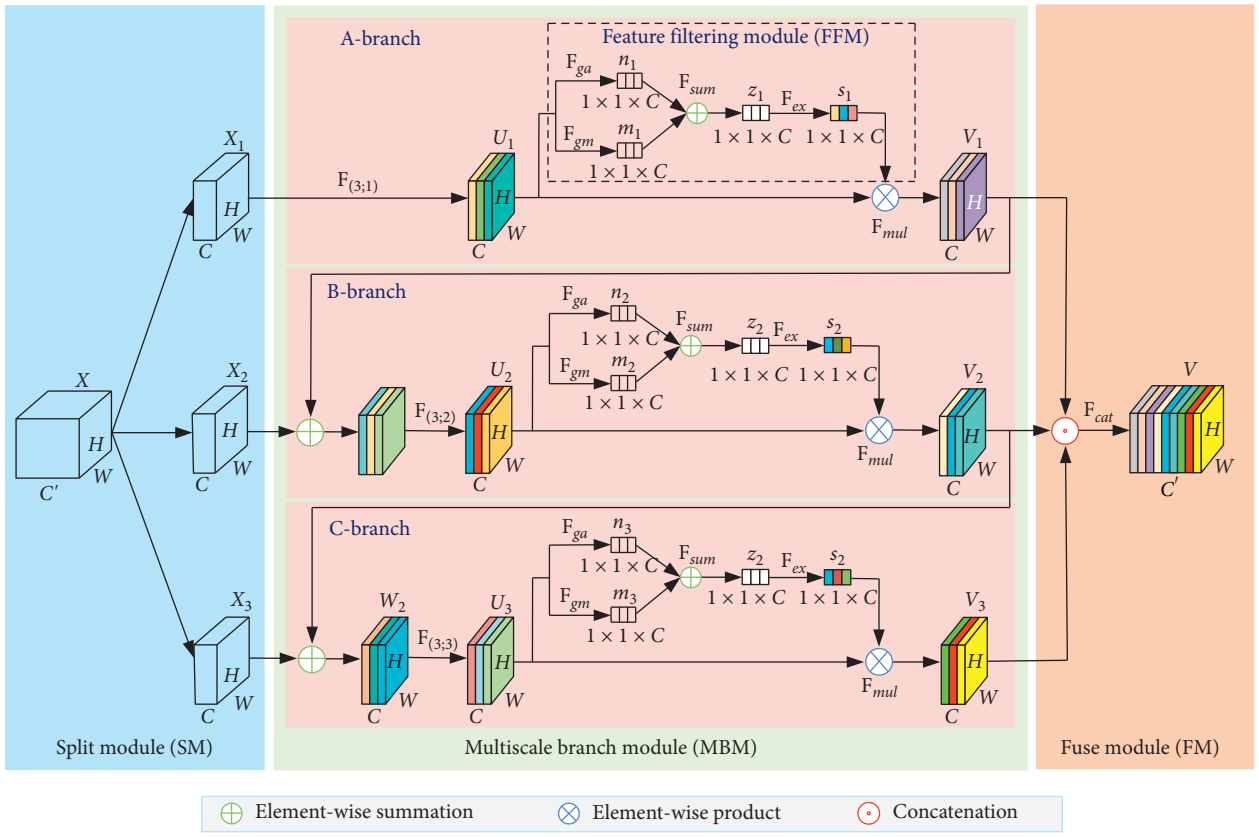

Figure 3: The structure of the multiscale feature filtering (MFF) module.

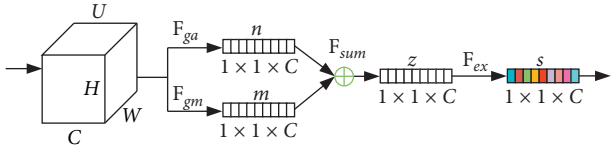

Figure 4: The structure of the feature filtering module (FFM).

where $U = [U^1, U^2, \ldots, U^c]$. U^c denotes the c-th feature map channel in the feature map U. In addition, $U^c(i, j)$ refers to (i, j) – th pixel in U^c.

Then, to fuse the transformed feature information from global average pooling and global max pooling, an element-wise summation is used to obtain finer global channel-wise statistics as $z \in \mathbb{R}^C$.

$$z = F_{\text{sum}}(n, m) = n \oplus m, \tag{3}$$

where F_{sum} indicates the element-wise summation operation between channel-wise statistics n and m. Furthermore, in order to make use of the previously fused feature information, the previously global channel-wise statistics z is forwarded to a F_{ex} function, which is composed of one dimensionality-reduction layer with parameters M_1 and reduction ratio l, dimensionality-increasing layer with parameters M_2, sigmoid activation function, and ReLU activation function. The final output of the FFM is computed as

$$s = F_{ex}(z) = \sigma\left(M_2 \delta\left(M_1 z\right)\right), \tag{4}$$

where σ and δ are the sigmoid and ReLU activation function, respectively. $M_1 \in \mathbb{R}^{(C/l)\times C}$ and $M_2 \in \mathbb{R}^{C\times(C/l)}$, where $l = 16$.

Employing large atrous rate enlarges the model's receptive field, so that object coding can be performed at multiple scales. As shown in Figure 5, A-branch, B-branch, and C-branch employ three atrous convolutions with different atrous rates r, where $r = \{1, 2, 3\}$. In addition, these branches use atrous convolutions with different rates instead of standard convolutions to reduce the model's parameters.

For any atrous convolution layer, the learned set of convolution filters $G = [G^1, G^2, \ldots, G^c]$, where $G^c \in \mathbb{R}^{K\times K\times C}$ refers to the parameters of the corresponding c-th convolution filter. Let $I \in \mathbb{R}^{H\times W\times C}$ denote the input of the atrous convolution layer. $U = [U^1, U^2, \ldots, U^c]$ are the output of the atrous convolution layer. For the c-th filter at such a layer, the corresponding output feature map channel is

$$U^c = F_{(K;r)}(G^c, I), \tag{5}$$

where $F_{(K;r)}$ denotes an atrous convolution layer with filter size $K \times K$ and atrous rate r.

In A-branch, for the input feature map subset X_1 obtained from the split module, an atrous convolution layer with filter size 3×3 and atrous rate $r = 1$ is conducted to generate the output feature map U_1 of a specific scale. For the c-th filter at such a layer, $K = 3$ and $r = 1$ are put into equation (5) to obtain the c-th output feature map channel.

$$U_1^c = F_{3\times 1}(G(1)^c, X_1), \tag{6}$$

where $F_{(3;1)}$ denotes an atrous convolution layer with filter size 3×3 and atrous rate $r = 1$. $G(1)$ denotes the learned set of $F_{(3;1)}$. $G(1) = [G(1)^1, G(1)^2, \ldots, G(1)^c]$ and $X_1 = [X_1^1, X_1^2, \ldots, X_1^c]$.

Further, in order to take advantage of the information aggregated in the feature filtering module (FFM), the feature map U_1 is sent to the FFM. The output of the FFM in A-branch is denoted as s_1. The final output V_1 of the A-branch is obtained by rescaling the feature map U_1 with an element-wise multiplication operation.

$$V_1 = F_{\text{mul}}(U_1, s_1) = U_1 \otimes s_1, \tag{7}$$

where F_{mul} indicates the element-wise multiplication operation.

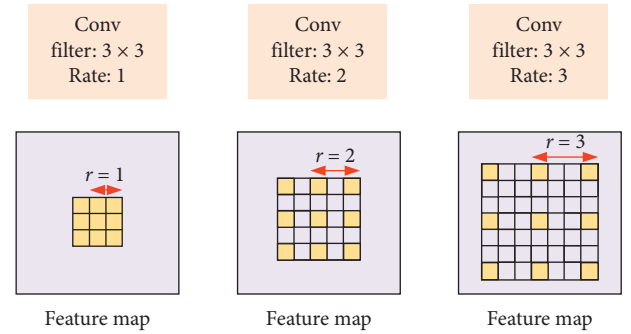

Figure 5: Atrous convolution with filter size 3×3 and different rates. A-branch: atrous convolution with filter size 3×3 and rate $(r) = 1$. B-branch: atrous convolution with filter size 3×3 and rate $(r) = 2$. C-branch: atrous convolution with filter size 3×3 and rate $(r) = 3$.

In B-branch, to enhance feature propagation, firstly we fuse the output V_1 of the A-branch and the feature map subset X_2 obtained from the split module by using an element-wise summation operation. Thus, the fusion output feature map W_1 of V_1 and X_2 is computed as

$$W_1 = F_{\text{sum}}(V_1, X_2) = V_1 \oplus X_2. \tag{8}$$

Then, an atrous convolution layer with filter size 3×3 and atrous rate $r = 2$ is conducted to generate the output feature map U_2. For the c-th filter at such a layer, $K = 3$ and $r = 2$ are put into equation (1) to obtain the c-th output feature map channel.

$$U_2^c = F_{(3;2)}(G(2)^c, W_1), \tag{9}$$

where $F_{(3;2)}$ denotes an atrous convolution layer with filter size 3×3 and atrous rate $r = 2$. $G(2)$ denotes the learned set of $F_{(3;2)}$. $G(2) = [G(2)^1, G(2)^2, \ldots, G(2)^c]$ and $W_1 = [W_1^1, W_1^2, \ldots, W_1^c]$.

Similar to the A-branch, in order to take advantage of the information aggregated in the feature filtering module (FFM), the feature map U_2 is sent to the FFM. The output of the FFM in B-branch is denoted as s_2. The final output V_2 of the B-branch is obtained by rescaling the feature map U_2 with an element-wise multiplication operation.

$$V_2 = F_{\text{mul}}(U_2, s_2) = U_2 \otimes s_2. \tag{10}$$

For the C-branch, similar to the B-branch, firstly we fuse the output V_2 of the C-branch and the feature map subset X_3 obtained from the split module by using an element-wise summation operation. Thus, the fusion output feature map W_2 of V_2 and X_3 is computed as

$$W_2 = F_{\text{sum}}(V_2, X_3) = V_2 \oplus X_3. \tag{11}$$

Then, an atrous convolution layer with filter size 3×3 and atrous rate $r = 3$ is conducted to generate the output feature map U_3. For the c-th filter at such a layer, $K = 3$ and $r = 3$ are put into equation (1) to obtain the c-th output feature map channel.

$$U_3^c = F_{(3;3)}\left(G(3)^c, W_2\right), \tag{12}$$

where $F_{(3;3)}$ denotes an atrous convolution layer with filter size 3×3 and atrous rate $r = 3$. $G(3)$ denotes the learned set of $F_{(3;3)}$. $G(3) = [G(3)^1, G(3)^2, \ldots, G(3)^c]$ and $W_2 = [W_2^1, W_2^2, \ldots, W_2^c]$.

The final output V_3 of the C-branch is obtained by rescaling the feature map U_3 with an element-wise multiplication operation.

$$V_3 = F_{\text{mul}}\left(U_3, s_3\right) = U_3 \otimes s_3, \tag{13}$$

where s_3 is the output of the FFM in the C-branch.

2.3. Fusion Module. As shown in Figure 1, in order to take advantage of the feature information aggregated in the multiscale branch module (MBM), the outputs of A-branch, B-branch, and C-branch are forwarded to the fusion module (FM), which is implemented by a concatenation function. The output of FM is V, which can be calculated by

$$V = F_{\text{cat}}\left(V_1, V_2, V_3\right) = V_1 \odot V_2 \odot V_3, \tag{14}$$

where F_{cat} denotes the concatenation operation between feature maps.

3. Experimental Results and Analysis

In this section, we describe experiments that study the effectiveness of MFF modules for a range of tasks, datasets, and model architectures. Besides, all models are implemented by using the PyTorch framework.

For image classification tasks, we evaluate all models on the CIFAR-100 and Tiny ImageNet datasets. The objects in the CIFAR-100 and Tiny ImageNet datasets have features of different scales, which can effectively verify the effectiveness of our proposed MFFNet in the UAV. For benchmarking, we evaluate the single-crop top-1 error rate and adopt the same data augmentation scheme used in [9, 27]. Moreover, we train the network using stochastic gradient descent with momentum 0.9, weight decay 0.0001, and a mini-batch size of 32 on 1 RTX 2080Ti GPU. For the CIFAR-100 and Tiny ImageNet datasets, every model is trained for 200 epochs. We start with a learning rate of 0.1, which is divided by 10 at 60, 120, and 160 epochs, respectively.

For object detection tasks, all models are trained in the PASCAL VOC 2007 and MS COCO datasets with 1 RTX 2080Ti GPU and the mini-batch size is 2 images. We use a weight decay of 0.0001 and a momentum of 0.9. In addition, all models are trained for 80k iterations with a learning rate of 0.002 and then for 30k iterations with 0.0001. Other implementation details are as in [28]. Besides, in order to verify the effectiveness of our proposed method, we further test the MFFNet on the UAV123 dataset, which is captured from a low-altitude aerial perspective.

3.1. Experiments on Tiny ImageNet. We evaluate our method on the Tiny ImageNet dataset, which contains 100k training images, 10k validation images, and 10k test images in 200 classes. Each class has 500 training images, 50 validation images, and 50 test images. An input image is 224×224 pixels randomly cropped from a resized image. We use ResNet-50, ResNet-101, and ResNeXt-50 as the representatives for the residual model architecture. In addition, we compare the results with those from the SENet and SKNet model architectures, which are based on attention mechanisms. We compare the single-crop top-1 error rate of each baseline and its MFFNet counterpart on the Tiny ImageNet dataset. As shown in Table 2, MFFNeXt-50 achieves significant performance gains over ResNeXt-50, with a reduction of 3.82% in the error rate. Compared with ResNet-50, MFFNet-50 is better by 1.04%. Meanwhile SENet-29 ($16c \times 32d$) achieves 33.67% error and MFFNeXt-50 ($32c \times 4d$) achieves 32.76% error. MFFNeXt-50 is better than SKNet-29 ($16c \times 32d$) by 2.27%. Besides, MFFNeXt-50 ($32c \times 4d$) achieves a top-1 error rate of 32.59%, although SENet-29 ($16c \times 32d$) needs 26.88% more parameters.

Table 2: Single 224×224 cropped top-1 error rate (%) and complexity comparisons on the Tiny ImageNet validation set. #P is the number of parameters. Atrous rate $r = \{1, 2, 3\}$.

Models	#P	Top-1 error rate (%)
ResNet-50	24.59	35.97
ResNet-101	43.24	35.15
ResNeXt-50 ($32c \times 4d$)	24.21	34.91
SENet-29 ($16c \times 32d$)	34.78	33.87
SKNet-29 ($16c \times 32d$)	27.45	33.55
MFFNet-50	25.82	35.03
MFFNet-101	44.86	33.84
MFFNeXt-50 ($32c \times 4d$)	25.43	32.59

The top-1 testing error rate versus number of epochs for the different architectures is shown in Figure 6. SKNet-29 ($16c \times 32d$) needs 27.45 M parameters, whereas MFFNeXt-50 ($32c \times 4d$) needs only 25.43 M trainable parameters and achieves a higher accuracy. The results show that MFF modules consistently improve the performance of state-of-the-art CNNs.

3.2. Experiments on CIFAR-100. To further evaluate the performance of the MFFNet, we conduct experiments on CIFAR-100. This dataset consists of 60k 32×32 color images drawn from 100 classes. There are 50k training images and 10k testing images. The 100 classes in CIFAR-100 are grouped into 20 superclasses. Each image has a fine label and a coarse label. We use implementations of ShuffleNetV2 1 ×, MobileNetV2 1 ×, ResNet-50, ResNeXt-50, ResNet-101, and DenseNet-BC-121 ($k = 12$) as the representative models. Similar to ShuffleNet [3], ShuffleNetV2 1 × and MobileNetV2 1 × mean scaling the number of filters by 1 time.

Table 3 shows more results of single-crop testing on CIFAR-100. Note that while ResNet-50 achieves a 21.55% error rate, MFFNet-50 achieves a 21.06% error rate. Moreover, MFFNeXt-50 ($32c \times 4d$) outperforms ResNeXt-50 ($32c \times 4d$) by achieving 20.03% top-1 error. For lightweight models, we compare ShuffleNetV2 1 × with MFF and MobileNetV2 1 × with MFF to the original ShuffleNetV2 1 × and MobileNetV2 1 ×, ShuffleNetV2 1 × with MFF and

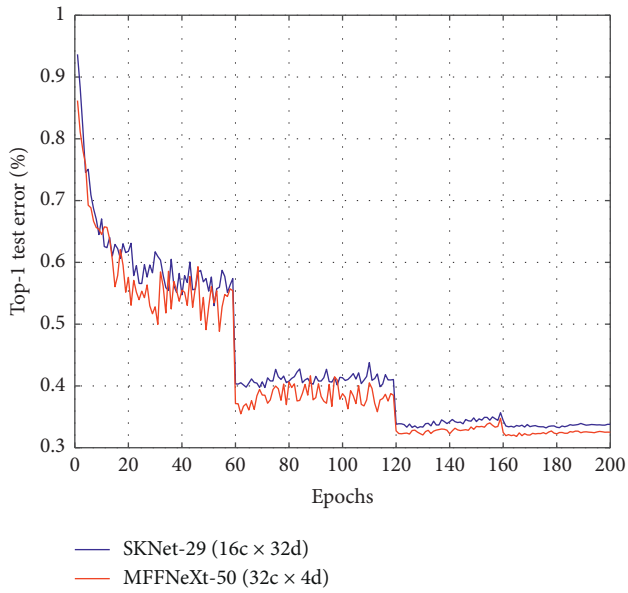

Figure 6: Top-1 testing error rate versus number of epochs for Tiny ImageNet. The numbers of parameters for SKNet-29 (16c × 32d) and MFFNeXt-50 (32c × 4d) are 27.45 M and 25.43 M, respectively.

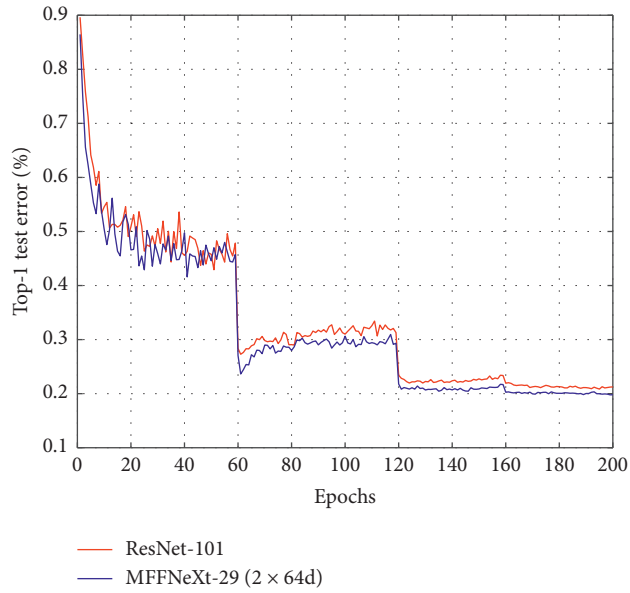

Figure 7: Top-1 testing error rate versus number of epochs for CIFAR-100. The numbers of parameters for ResNet-101 and MFFNeXt-29 (2c × 64) are 42.70 M and 9.37 M, respectively.

Table 3: Single 224 × 224 cropped top-1 error (%) and complexity comparisons on CIFAR-100. Atrous rate $r = \{1, 2, 3\}$.

Models	#P	Top-1 error rate (%)
ShuffleNetV2 1 ×	1.36	30.58
MobileNetV2 1 ×	2.37	30.21
ResNet-50	23.68	21.55
ResNet-101	42.70	21.25
ResNeXt-50 (32c × 4d)	23.14	20.59
ResNeXt-29 (2c × 64d)	9.22	22.63
ResNeXt-29 (4c × 64)	36.58	21.29
DenseNet-BC-121 ($k = 12$)	7.05	22.93
ShuffleNetV2 1 × + MFF	1.41	29.62
MobileNetV2 1 × + MFF	2.53	29.27
MFFNet-50	24.86	21.06
MFFNet-101	44.23	20.31
MFFNeXt-50 (32c × 4d)	24.61	20.03
MFFNeXt-29 (2c × 64d)	9.37	19.78
MFFNeXt-29 (4c × 64d)	37.85	17.36
DenseNet-BC-121 ($k = 12$) + MFF	7.59	21.54

MobileNetV2 1 × with MFF outperform original ShuffleNetV2 1 × and MobileNetV2 1 × by 0.96% and 0.94%, respectively.

In addition, for densely connected models, we choose a DenseNet-BC network with 121 layers. DenseNet-BC-121 ($k = 12$) with MFF achieves a performance gain of 1.29% over DenseNet-BC-121 ($k = 12$). The top-1 testing error rate versus number of epochs for the different architectures is shown in Figure 7. We can clearly see that MFFNeXt-29 (2c × 64d) outperforms ResNet-101 by achieving 19.78% top-1 error, although ResNet-101 needs more parameters.

3.3. Ablation Studies on CIFAR-100. To further validate the effectiveness of the MFFNet, we undertake ablation studies on the CIFAR-100 dataset. We first evaluate the trade-off between cardinality c and bottleneck width d. Next, in MFF module, we investigate the impact of changes in the complexity on performance by combining different atrous rates r.

3.3.1. Cardinality versus Width. To study the effects of the cardinality c and the width of the bottleneck d, we start from the three-branch case and fix the setting atrous rates $r = \{1, 2, 3\}$. We first evaluate the trade-off between cardinality c and bottleneck width d. Table 4 shows the results. MFFNeXt-29 (2c × 64d) has a top-1 error of 19.78%, which is 2.78% lower than that for MFFNeXt-29 (1c × 64d). We can see that as the cardinality c increases from 1 to 4 for constant bottleneck width, the error rate falls. In addition, as the bottleneck width d increases from 24 to 64 for constant cardinality c, the error rate again decreases.

We also note that increasing cardinality c can achieve much better results than going wider. For instance, MFFNeXt-29 (2c × 40d) performs better than MFFNeXt-29 (1c × 64d), even though it has 66.33% fewer parameters. MFFNeXt-29 (2c × 64d) needs 9.37 M parameters, whereas MFFNeXt-29 (4c × 40d) needs only 9.21 M trainable parameters and achieves a higher accuracy.

3.3.2. Combinations of Different Atrous Rates. Next, we investigate combinations of different atrous rates. The atrous rate r is used to control the receptive field size. MFFNet uses 3 × 3 filters with different atrous rates r. To limit the search

Table 4: Single 224 × 224 cropped top-1 error (%) and complexity comparisons on CIFAR-100. Atrous rate $r = \{1, 2, 3\}$.

Models	#P	Top-1 error rate (%)
MFFNeXt-29 (1c × 24d)	0.43	27.61
MFFNeXt-29 (1c × 40d)	1.05	24.73
MFFNeXt-29 (1c × 64d)	3.98	22.56
MFFNeXt-29 (2c × 24d)	0.76	24.03
MFFNeXt-29 (2c × 40d)	1.34	20.62
MFFNeXt-29 (2c × 64d)	9.37	19.78
MFFNeXt-29 (4c × 24d)	3.67	20.45
MFFNeXt-29 (4c × 40d)	9.21	19.01

Table 5: Results for MFFNeXt-29 (2c × 64d) for combinations of different atrous rates.

Models	$r = 1$	$r = 2$	$r = 3$	$r = 4$	#P	Top-1 error rate (%)
MFFNeXt-29 (2c × 64d)	Y	Y	N	N	6.63	20.77
MFFNeXt-29 (2c × 64d)	Y	N	Y	N	6.63	20.24
MFFNeXt-29 (2c × 64d)	Y	N	N	Y	6.63	20.01
MFFNeXt-29 (2c × 64d)	Y	Y	Y	N	9.37	19.78
MFFNeXt-29 (2c × 64d)	Y	Y	N	Y	9.37	19.49
MFFNeXt-29 (2c × 64d)	Y	N	Y	Y	9.37	19.75
MFFNeXt-29 (2c × 64d)	Y	Y	Y	Y	12.11	19.83

Table 6: Results for MobileNetV2 1 × with MFF for combinations of different atrous rates.

Models	$r = 1$	$r = 2$	$r = 3$	$r = 4$	#P (M)	Top-1 error rate (%)
MobileNetV2 1 × + MFF	Y	Y	N	N	2.44	29.65
MobileNetV2 1 × + MFF	Y	N	Y	N	2.44	29.59
MobileNetV2 1 × + MFF	Y	N	N	Y	2.44	30.25
MobileNetV2 1 × + MFF	Y	Y	Y	N	2.53	29.27
MobileNetV2 1 × + MFF	Y	Y	N	Y	2.53	29.79
MobileNetV2 1 × + MFF	Y	N	Y	Y	2.53	30.21
MobileNetV2 1 × + MFF	Y	Y	Y	Y	2.61	30.18

space, we use only four different atrous rates, $r = 1$, 2, 3, and 4. To study their effects, we change the other three branches for the 3×3 filter with $r = 1$ in the first filter branch of the MFF modules. Tables 5 and 6 show the top-1 error rate for MFFNeXt-29 (2c × 64d) and MobileNetV2 1 × with MFF. We can make three major observations as follows:

(1)First, when the number of branches in an MFF module $b = 2$, the top-1 error rate for MFFNeXt-29 (2c × 64d) gradually decreases as the atrous rate in the second branch increases. Moreover, MFFNeXt-29 (2c × 64d) achieves the lowest top-1 error for $r = \{1, 4\}$. In contrast, the top-1 error rate of MobileNetV2 1 × with MFF gradually increased as the atrous rate in the second branch increased.

(2) Second, when the number of branches in an MFF module $b = 3$, MFFNeXt-29 (2c × 64d) achieves the lowest top-1 error rate for $r = \{1, 2, 4\}$. However, MobileNetV2 1 × with MFF has the lowest top-1 error rate for $r = \{1, 2, 3\}$.

(3) Third, when the number of branches in an MFF module $b = 4$, MFFNeXt-29 (2c × 64d) and MobileNetV2 1 × with MFF do not achieve the lowest top-1 error rate. For example, MFFNeXt-29 (2c × 64d) with $b = 3$ achieves higher accuracy, although MFFNeXt-29 (2c × 64d) with $b = 4$ needs 22.63% more parameters. MobileNetV2 1 × with MFF for $r = \{1, 2, 3\}$ outperforms MobileNetV2 1 × with MFF for $r = \{1, 2, 3, 4\}$ by above 0.91% accuracy.

3.3.3. Class Activation Mapping. To intuitively understand the multiscale representation ability of MFFNet, we visualize the class activation mapping (CAM) using Grad-CAM for different networks. Grad-CAM uses gradients to calculate the importance of the spatial locations in convolutional layers.

Figure 8 compares the CAM for representative backbone networks. The areas that have a larger impact on the classification are covered with lighter colors. We can clearly see that the MFFNet-based CAM results tend to focus on the whole object more than ResNet.

3.3.4. Object Detection. The PASCAL VOC 2007 and MS COCO datasets are in 20 and 80 object categories, respectively. The PASCAL VOC 2007 dataset has about 5k trainval images and 5k test images. We use the 5k trainval images and 5k test images for training and 5k test images for validation. The MS COCO dataset has 80k images for training, 40k for validation, and 20k for testing. We used the 80k training set plus a 35k validation subset for training and a 5k validation subset for validation. We adopt Faster-RCNN [28] as our detection method and evaluate the average precision (AP)

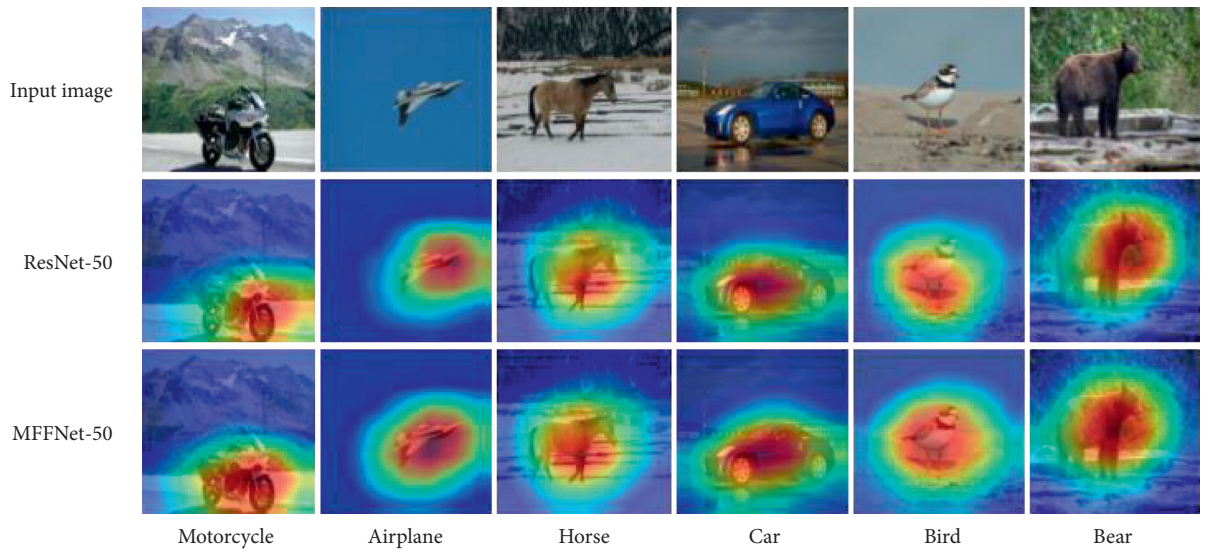

Figure 8: Grad-CAM visualization results, using ResNet-50 and our proposed MFFNet-50 with atrous rates $r = \{1, 2, 4\}$ as the backbone networks.

Table 7: Object detection. The detector is Faster-RCNN. Atrous rates $r = \{1, 2, 4\}$.

Backbone	Dataset	AP (%)	Inference (ms)
ResNet-101	VOC 2007	71.4	73.9
MFFNet-101 (ours)	VOC 2007	72.8	75.5
ResNet-101	COCO	25.6	135.3
MFFNet-101 (ours)	COCO	26.9	138.5

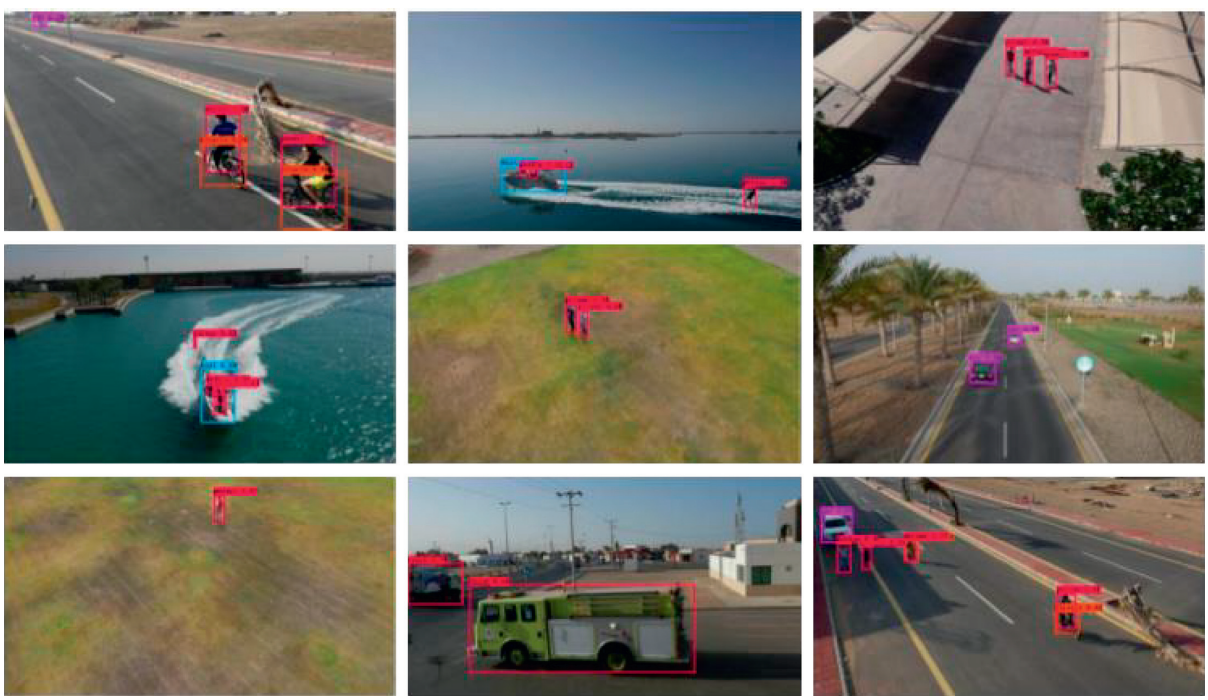

Figure 9: Detection examples generated by MFFNet-101 as backbone networks on UAV123 dataset. MARNet-101 is trained on the VOC 2007 trainval + test.

for PASCAL VOC 2007 and MS COCO. Moreover, ResNet-101 and MFFNet-101 are used as our backbone networks.

On the PASCAL VOC 2007 dataset, MFFNet-101 outperforms ResNet-101 by 1.4% on AP. On the MS COCO dataset, we improve ResNet-101 by 1.3%. Table 7 shows that MFFNet-101 has a little longer inference latency than ResNet-101 but is more accurate. For instance, on the PASCAL VOC 2007 dataset, we improve the ResNet-101 baseline by 1.4% for AP for only 1.6 ms of additional inference latency. On the MS COCO dataset, MFFNet-101 has an AP of 26.9%, which is 1.3% higher than the ResNet-101 baseline of 25.6% for only 3.2 ms of additional inference latency. These results demonstrate the general performance improvement of using MFF modules in object detection. Figure 9 shows detection examples generated by our proposed MFFNet-100 as backbone networks on UAV123 dataset. It can be seen that our method is able to detect target objects successfully regardless of their shapes, sizes, orientations, and appearances.

4. Conclusions

To address the multiscale recognition problem in the UAV visual perception, this paper establishes a new convolutional network architecture (MFFNet). In MFFNet, the MFF module is designed by employing multiple atrous convolutions at different rates with feature-selective learning ability. The MFF module is implemented via three operations: split module (SM), multiscale branch module (MBM), and fusion module (FM). In addition, MFF module can selectively generate channel-wise feature responses by emphasizing channel-wise dependencies. We further explore the effect of atrous rate on the multiscale representation ability of CNNs. Image classification results on CIFAR-100 and Tiny ImageNet datasets demonstrate that our proposed method achieves very competitive results on various benchmarks. Grad-CAM visualization results demonstrate that the MFFNet-based CAM results tend to focus on the whole object more than other baseline networks. That is, the MFFNet has a stronger multiscale representation ability, which can achieve better recognition accuracy in the UAV. Experimental results on PASCAL VOC 2007, MS COCO, and UAV123 datasets show that our proposed method achieves consistent performance gains in object detection, which is beneficial to expanding the application of UAV. We will further explore the effect of multiscale representation on image recognition results in future work.

Data Availability

The detailed mechanism model and model parameters of MFFNet are given in the article. The results are computed on the PyCharm software with the model and given parameters, while the relevant results are also given in the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Major Science and Technology Projects of China (Grant no. 2019ZX04026001).

References

[1] A. Zeggada, S. Benbraika, F. Melgani, and Z. Mokhtari, "Multilabel conditional random field classification for UAV images," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 399–403, 2018.

[2] A. Zeggada, F. Melgani, and Y. Bazi, "A deep learning approach to UAV image multilabeling," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 694–698, 2017.

[3] A. Krizhevsky, I. Sutskever, N. V. Lake Tahoe, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Processing Advances in Neural Information Processing Systems*, pp. 1097–1105, MIT press, London, UK, 2012.

[4] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, Salt Lake City, UT, USA, 2018.

[5] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, Honolulu, HI, USA, July 2017.

[6] G. Li and Y. Yu, "Contrast-oriented deep neural networks for salient object detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12, pp. 6038–6051, 2018.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 886–893, San Diego, CA, USA, June 2005.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015*, pp. 1409–1556, San Diego, CA, USA, May 2015.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[10] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, Boston, MA, USA, June 2015.

[11] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, Honolulu, HI, USA, July 2017.

[12] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, https://arxiv.org/abs/1605.07146.

[13] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, Honolulu, HI, USA, July 2017.

[14] X. Ding, Y. Guo, G. Ding, and J. Han, "ACNet: strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks," in *Proceedings of the 2019 IEEE/CVF*

International Conference on Computer Vision (ICCV), pp. 1911–1920, Seoul, South Korea, October 2019.

[15] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, Salt Lake City, UT, USA, June 2018.

[16] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: convolutional block attention module," https://arxiv.org/abs/1807.06521.

[17] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the IEEE conference on computer vision and pattern Recognition (CVPR)*, pp. 510–519, Seattle, WA, USA, July 2019.

[18] X. Ma, Z. Yang, and Z. Yu, "FSRFNet: feature-selective and spatial receptive fields networks," *Applied Sciences*, vol. 9, no. 19, p. 3954, 2019.

[19] R. R. Selvaraju, M. Cogswell, A. Das et al., "Grad-CAM: visual explanations from deep networks via gradient-based localization," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, Venice, Italy, October 2017.

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical Report, University of Toronto, Toronto, Canada, 2009.

[21] L. Yao and J. Miller, "Tiny ImageNet classification with convolutional neural networks," *CS 231N*, vol. 2, p. 8, 2015.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[23] T.-Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," in *Proceedings of the European Conference on Computer Vision*, pp. 740–755, Springer, Zurich, Switzerland, September 2014.

[24] M. Mueller, N. Smith, and B. Ghanem, "A benchmark and simulator for UAV tracking," in *Proceedings of the European Conference on Computer Vision*, pp. 445–461, Amsterdam, Netherlands, October 2016.

[25] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet v2: practical guidelines for efficient CNN architecture design," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 116–131, Munich, Germany, September 2018.

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNet v2: inverted residuals and linear bottlenecks," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, Salt Lake City, UT, USA, June 2018.

[27] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, Las Vegas, NV, USA, June 2016.

[28] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.