Few-Shot Object Detection on Remote Sensing Images

Xiang Li, Jingyu Deng, and Yi Fang, *Member, IEEE*

***Abstract*— In this article, we deal with the problem of object detection on remote sensing images. Previous researchers have developed numerous deep convolutional neural network (CNN)-based methods for object detection on remote sensing images, and they have reported remarkable achievements in detection performance and efficiency. However, current CNN-based methods often require a large number of annotated samples to train deep neural networks and tend to have limited generalization abilities for unseen object categories. In this article, we introduce a metalearning-based method for few-shot object detection on remote sensing images where only a few annotated samples are needed for the unseen object categories. More specifically, our model contains three main components: a metafeature extractor that learns to extract metafeature maps from input images, a feature reweighting module that learns class-specific reweighting vectors from the support images and use them to recalibrate the metafeature maps, and a bounding box prediction module that carries out object detection on the reweighted feature maps. We build our few-shot object detection model upon the YOLOv3 architecture and develop a multiscale object detection framework. Experiments on two benchmark data sets demonstrate that with only a few annotated samples, our model can still achieve a satisfying detection performance on remote sensing images, and the performance of our model is significantly better than the well-established baseline models.**

***Index Terms*— Few-shot learning, metalearning, object detection, remote sensing images, You-Only-Look-Once (YOLO).**

I. Introduction

OBJECT detection has been a long-standing problem in both remote sensing and computer vision fields. It is generally defined as identifying the location of target objects in the input image and recognizing the object categories. Automatic object detection has been widely used in many real-world applications, such as hazard detection, environmental monitoring, change detection, and urban planning [1], [2].

Manuscript received November 6, 2020; revised December 14, 2020; accepted January 10, 2021. Date of publication February 24, 2021; date of current version December 6, 2021. This work was supported in part by the ADEK under Grant AARE-18150, and in part by the NYU Abu Dhabi Institute under Grant AD131. *(Xiang Li and Jingyu Deng contributed equally to this work.) (Corresponding author: Yi Fang.)*

Xiang Li and Yi Fang are with the NYU Multimedia and Visual Computing Laboratory, New York University Abu Dhabi, Abu Dhabi 129188, UAE, also with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi 129188, UAE, and also with the Department of Electrical and Computer Engineering, New York University Tandon School of Engineering, Brooklyn, NY 11201 USA (e-mail: yfang@nyu.edu).

Jingyu Deng is with the NYU Multimedia and Visual Computing Laboratory, New York University Abu Dhabi, Abu Dhabi 129188, UAE, and also with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi 129188, UAE.

Digital Object Identifier 10.1109/TGRS.2021.3051383

In the past decades, object detection has been extensively studied, and a large number of methods have been developed for the detection of both artificial objects (e.g., vehicles, buildings, roads, and bridges) and natural objects (e.g., lakes, coasts, and forests) in remote sensing images. Existing object detection methods in remote sensing images (RSIs) can be roughly divided into four categories: 1) template matching-based methods; 2) knowledge-based methods; 3) object-based image analysis (OBIA)-based methods; and 4) machine learning-based methods [1]. Among these methods, the machine learning-based methods have powerful abilities for robust feature extraction and object classification and have been extensively studied by many recent approaches and achieved significant progress in solving this problem [3]–[6].

In recent years, among all machine learning-based methods for object detection, deep learning methods, especially convolutional neural networks (CNNs), have drawn immense research attention. Due to the powerful feature extraction abilities of CNN models, a huge number of CNN-based methods have been developed for object detection on both optical and remote sensing images. Notable methods include Faster R-CNN [7], You-Only-Look-Once (YOLO) [8], and single-shot-detector (SSD) [9]. In the remote sensing field, recent studies mostly build their methods using these prevalent deep learning-based architectures.

Despite the breakthroughs achieved by deep learning-based methods for object detection, these methods suffer from a common issue: a large-scale, diverse data set is required to train a deep neural network model. Any adjustments of the candidate identifiable classes are expensive for existing methods because collecting a new RSI data set with a large number of manual annotations is costly, and these methods need a lot of time to re-train their models on the newly collected data set. On the other hand, training a model with only a few samples from the new classes tends to cause the model to suffer from the overfitting problem, and its generalization abilities are severely reduced. Therefore, a special mechanism of learning robust detection from a few samples of the new classes is desired for object detection on RSIs.

In the past few years, few-shot learning has been extensively studied in the computer vision field to undertake the tasks of scene classification [10]–[12], image segmentation [13]–[15], object detection [16]–[19], and shape analysis [20], [21]. Few-shot learning aims at learning to learn transferable knowledge that can be well generalized to new classes and, therefore, performs image recognition (e.g., classification and

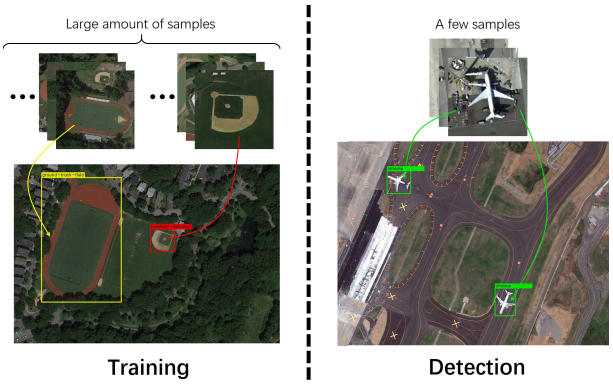


Fig. 1. Illustration of few-shot detection on remote sensing images. Our model is trained with large number of annotated samples from the seen classes and performs detection on unseen classes with only a few annotated samples.

segmentation) on new classes with only a few annotated examples. Existing few-shot object detection methods are designed for common object detection (e.g., bicycles, cars, and chairs) in optical images using a single-scale detection network. These common objects usually have small size variations, while, in remote sensing images, objects can have very different sizes, and the spatial resolution of RSIs can be quite different, which makes the problem even more challenging when only a few annotated samples are provided.

In this article, we introduce a metalearning-based method for few-shot object detection on remote sensing images. Under the few-shot scenario, our model aims to learn a detection model from the data set of seen classes that can conduct accurate object detection for unseen (novel) classes with only a few annotated samples. Fig. 1 illustrates the basic idea of few-shot object detection on remote sensing images. We build our method based upon a recently published article [19], which is designed for common object detection in optical images. To address the scale variations inherently present in remote sensing images, we extend [19] to a multiscale feature extraction and object detection framework. Concretely, a metafeature extractor is designed to learn to extract metafeature maps from input images. A feature reweighting module is designed to learn class-specific reweighting vectors that carry the necessary information for detecting objects of the same classes and use them to recalibrate the metafeature maps. The feature reweighting module is implemented by a deep neural network. A bounding-box prediction module carries out object detection on the reweighted feature maps. Our few-shot detection method includes two training stages: the metatraining stage and the meta-fine-tuning stage. In the metatraining stage, our model is trained on a large amount of data from the seen classes and learns metaknowledge for object detection. In the meta-fine-tuning stage, a few samples from the unseen classes (no overlapping with the seen classes) are used to fine-tune the model to make it adapted to the unseen classes while still maintaining the metaknowledge during the metatraining stage.

The main contributions of this article are summarized as follows.

1) In this article, we introduce the first metalearning-based method for few-shot object detection on remote sensing images. Our method is trained with large-scale data from some seen classes and can learn metaknowledge about object detection and, thus, generalize well to unseen classes with only a few labeled samples.
2) Our method contains three main components: a metafeature extraction network, a feature reweighting module, and a bounding box prediction module. All three modules are designed with multiscale architecture to enable multiscale object detection.
3) Experiments on two public benchmark data sets demonstrate the effectiveness of the proposed method for few-shot object detection on remote sensing images.

II. Related Work

A. Object Detection in Computer Vision

Object detection is a hot topic in the computer vision field with extensive studies, especially since the prosperity of deep learning methods. CNNs, as one of the commonly used deep learning models, have achieved great success for various vision tasks, include image classification [22], [23], semantic segmentation [24], [25], object detection [7], [26], image registration [27], [28], and point cloud and shape analysis [29], [30]. R-CNN [31] is one of the earliest and successful methods that adopt CNNs for object detection. In R-CNN, the authors replace the traditional handcrafted feature engineering with a CNN-based feature learning process, and the model experiences a significant performance boost. Following R-CNN, Fast R-CNN [32] performs feature extraction on the original input images and maps all region proposals onto the extracted feature map. A region-of-interest (RoI) pooling layer is proposed to transform feature representations of each ROI into a fixed-length vector. To facilitate the neural network design, the SVM classifier is replaced with a softmax classifier, and the bounding box regression process is included within the model instead of doing it afterward. Fast R-CNN improves detection efficiency by a large margin. Another important variant comes from Faster R-CNN [7]. To further overcome the computation burden from the region proposal generation process, Faster R-CNN introduces a region proposal network (RPN) to generate region proposals from the CNN network and enables weight sharing between the RPN network and the detection network. The following works, such as [33], [34], mostly base their method on the Faster R-CNN architecture. For example, Mask R-CNN [34] adopts feature pyramid network (FPN) [33] as the backbone network to produce multiscale feature maps and adds a mask prediction branch to detect the precise boundary of each instance.

The abovementioned approaches generally divide the detection process into two stages: region proposal generation and object detection from the region proposals. These methods are, therefore, often called two-stage object detectors. Another family of methods removes the region proposal generation process and directly conducts object detection on the input images. These methods are, therefore, often called one-stage object detectors. One of the most successfully one-stage object detectors is YOLO [8]. In the YOLO model, the input image is divided into grid cells, and each cell is responsible for

detecting a fixed number of objects. A deep CNN architecture is designed to learn high-level feature representations for each cell, and a successive of fully connected layers are used to predict the object categories and locations. YOLO is generally a lot faster than two-stage object detectors but with inferior detection performance. Subsequent variants, such as YOLOv2 [35] and YOLOv3 [36], improve the performance by using a more powerful backbone network and conduct object detection on multiple scales. More specifically, the YOLOv3 model adopts FPN [33] as the backbone network and, thus, enables more powerfully feature extraction and detection at different scales. Follow-on research efforts mostly improve the performance by using deconvolutional layers [37], multiscale detection pipeline [9], or focal loss [38].

B. Object Detection in RSIs

Existing methods for object detection on remote sensing images fall into four categories, template matching-based methods, knowledge-based methods, object-based image analysis (OBIA)-based methods, and machine learning-based methods [1]. The template matching-based methods use the stored templates, which are generated through handcrafting or training, to find the best matches at each possible location in the source image. Typical template matching-based methods include rigid template matching [39]–[41] and deformable template matching [42]. Knowledge-based methods treat the object detection problem as a hypothesis testing process by using preestablished knowledge and rules. Two kinds of well-known knowledge are geometric knowledge [43]–[46] and context knowledge [43], [47], [48]. OBIA-based methods start with segmenting images into homogeneous regions that represent a relatively homogeneous group of pixels and then perform region classification using region-level features from handcrafted feature engineering. The last family of methods, machine learning-based object detectors, contains two fundamental processes: handcrafted feature extraction and classification using machine learning-based algorithms. Machine learning-based methods have shown more powerful generalization abilities compared to the other three families of methods [1].

Among all machine learning-based methods, deep learning-based methods have drawn enormous research attention and are widely used in recent RSI object detection works. Unlike traditional machine learning-based methods that use handcrafted features, deep learning-based methods use deep neural networks to automatically learn robust features from input images. Following this research trajectory, early efforts adopt R-CNN architecture to detect geospatial objects on remote sensing images [49]–[56]. For example, Chen *et al.* [49] introduced a new rotation-invariant layer to the R-CNN architecture to enhance the performance for detection of objects with different orientations. Zhang *et al.* [57] introduced a hierarchical feature encoding network for robust object representation learning and demonstrate the effectiveness of their method for object detection on high-resolution remote sensing images. Following the great success of Faster R-CNN, numerous works have tried to extend the Faster R-CNN framework to the remote sensing community [58]–[61]. For example, Li *et al.* [53] developed a rotation-insensitive RPN by using multiangle anchors instead of the horizontal anchors used in conventional RPN networks. The proposed method can effectively detect geospatial objects of arbitrary orientations.

Following the great success of one-stage based methods for object detection on natural images, researchers also developed various regression-based methods for object detection on remote sensing images [26], [51], [61], [62]. For example, [61] extends the SSD model to conduct real-time vehicle detection on remote sensing images. Reference [62] replaces the horizontal anchors with oriented anchors in the SSD [9] framework and, thus, enables the model to detect objects with orientation angles. Subsequent methods further enhance the performance of geospatial object detection on remote sensing images by using hard example mining [51], multifeature fusion [63], transfer learning [64], nonmaximum suppression [65], and so on.

C. Few-Shot Detection

Few-shot learning aims at learning to learn transferable knowledge that can be generalized to new classes and, therefore, performs image recognition (e.g., classification, detection, and segmentation) on new classes where only a few annotated samples are given. In recent years, few-shot detection has received increased attention recently in the computer vision field. Chen *et al.* [66] proposed to fine-tune a pretrained model, such as Faster R-CNN [7] and SSD [9], on a few given examples and transfers it into a few-shot object detector. In [67], the authors enrich the training examples with additional unannotated data in a semisupervised setting and obtain performances comparable to weakly supervised methods with a large amount of training data. Karlinsky *et al.* [17] introduced a metric learning subnet to replace the classification head of a standard detection architecture [33] and achieve satisfying detection performance with a few training samples. Kang *et al.* [19] introduced a reweighting module to produce a group of reweighting vectors from a few supporting samples, one for each class, to reweight the metafeature extracted from the DarkNet-19 network. With the reweighted metafeatures, a bounding box prediction module is adapted to produce the detection results. However, the DarkNet-19 only produces a single metafeature map for each input image, leading to poor performance when detecting objects with large size variations. In contrast to [19] that only conducts object detection on a single-scale feature map, our proposed method extracts hierarchy feature maps with different scales from an FPN-like structure and improves the performance by performing multiscale object detection in the few-shot scenario.

D. Few-Shot Learning and Transfer Learning

Few-shot learning is one of the applications of metalearning in the supervised domain. It shares a lot of similarities in terms of reusability with another commonly used technique called transfer learning. Metalearning, also called learn to learn, is generally defined as the machine learning theory in

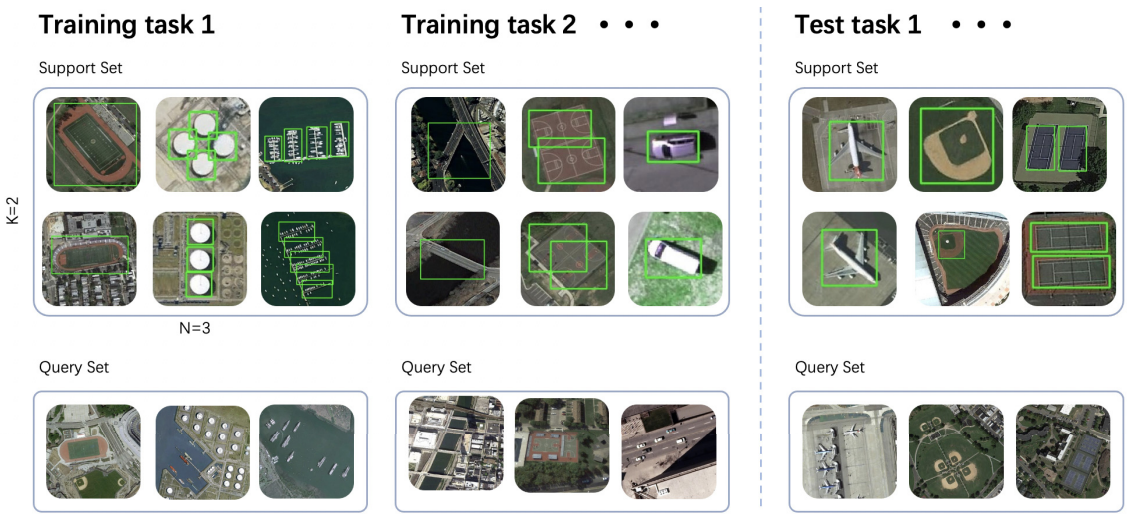


Fig. 2. Illustration of a three-way-two-shot setting. For each task, the support set consists of two annotated images for each of the three object categories.

which some algorithms are designed to learn new concepts and skills fast with a few training examples. In contrast, transfer learning works by transferring the knowledge learned from one problem to a different but related problem. By comparison, metalearning is more about learning prior information from a branch of tasks that can be used to quickly learn new models for some new tasks, whereas transfer learning uses the model trained for a source task as the initialization for some new target tasks that are relatively similar to the source task. Thus, although they can both be used for the task to task transferring, their focuses are quite different: one tries to learn prior knowledge and fast adaption for new tasks, and the other simply reuses an already optimized model, or part of it.

III. Method

A. Method Overview

We first clarify the settings for the few-shot object detection problem. The problem of few-shot object detection aims at learning a detection model from the data set of seen classes that can conduct object detection on images from unseen classes with only a few annotated samples. There are adequate samples for model training for each seen class, while each unseen class has only a few annotated samples. A few-shot object detection model should be able to learn metaknowledge from the data set of seen classes and well transfer it to the unseen classes.

This few-shot object detection setting is very common in real-world scenarios. One may need to develop a new object detection model, while collecting a large scale data set for the target classes is time-consuming. A good starting point would be deploying a detection model pretrained on some existing large-scale object detection data sets (e.g., DIOR [2]). However, these data sets only cover a limited number of object categories, while one may only focus on several specific object categories that may not happen to be included in these data sets. This calls for the need for few-shot-based object detection models in the remote sensing field.

To achieve few-shot learning, a common solution is to build the detection model using metalearning techniques. In a few-shot detection scenario, a metalearning algorithm learns to learn metaknowledge from a large number of detection tasks sampled from the seen classes and, thus, can generalize well to unseen classes. Each of the sampled tasks is called an episode. Each episode $\mathcal{E}$ is constructed from a set of support images $\mathcal{S}$ (with annotations) and a set of query images $\mathcal{Q}$. For each episode, the support images can be regarded as the training samples and are used for learning how to solve this task, while the query images can be regarded as the test samples and are used for evaluating the performance on this task. Fig. 2 gives an illustration of the few-shot detection setting.

We follow [19] to construct episodes from the data set of both seen and unseen categories. Given an N-way-K-shot detection task, each support set $\mathcal{S}$ consists of K annotated images for each of the N object categories. We denote the support set as $\mathcal{S} = \{(I_{\text{ck}}, M_{\text{ck}})\}$, where I_{ck} denotes the input image, $I_{\text{ck}} \in \mathbb{R}^{h\times w\times 3}$, $c = 1, 2, \ldots, N, k = 1, 2, \ldots K$, and M_{ck} denotes the corresponding bounding box annotations. The query set $\mathcal{Q}$ contains N_q images from the same set of class $\mathcal{C}$ as the support set. During metatraining, we randomly sample as many episodes from the data set of seen classes to train our model to learn metaknowledge of how to detect objects in the query images given the information lied in support images. Each of the sampled episodes/tasks can be completely nonoverlapping. After metatraining, we fine-tune our model on the unseen classes with a few samples to make our model well adapted to the unseen classes.

Fig. 3 illustrates the pipeline of the proposed method. Our few-shot object detection model (FSODM) is designed to leverage the metaknowledge from the data set of seen classes. Specifically, a metafeature extractor module is first developed to learn metafeatures at three different scales from input query images. Then, a feature reweighting module takes as an input N support images with labels, one for each class, and outputs three groups of N reweighting vectors, one for each scale. These reweighting vectors are used to recalibrate the metafeatures of the same scale through a channelwise multiplication. With the reweighting module, the metainformation from the support samples is extracted and used to amplify

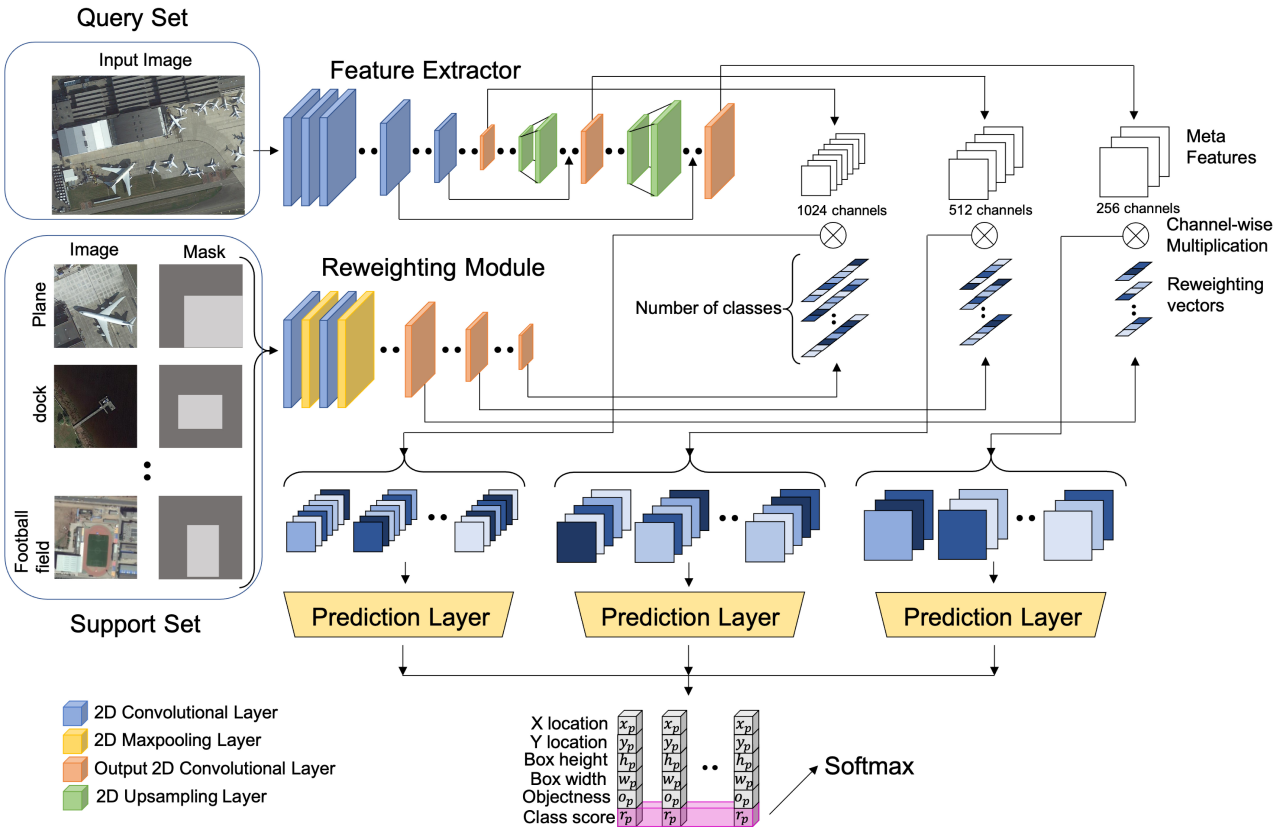


Fig. 3. Pipeline of the proposed method for few-shot object detection on remote sensing images. Our method consists of three main components, a metafeature extractor, a reweighting module, and a bounding box prediction module. The feature extractor network takes a query image as input and produces metafeature maps at three different scales. The reweighting module takes as an input K support images with labels for each of the N classes and outputs three groups of N reweighting vectors. These reweighting vectors are used to recalibrate the metafeature maps of the same scale through a channelwise multiplication. The reweighted feature maps are then fed into three independent bounding box detection modules to predict the bounding box locations and sizes (x_p, y_p, w_p, and h_p), the objectness scores (o_p), and the classification scores (r_p) at three different scales.

those metafeatures that are informative for detecting novel objects in the query images. The reweighted metafeatures are then fed into three independent bounding box detection modules to predict the objectness scores (o), the bounding box locations, and sizes (x, y, w, h) and class scores (c) at three different scales.

B. Metafeature Extractor

Our metafeature extractor network is designed to extract robust feature representations from input query images. Unlike [19] that only extracts single-scale metafeatures, objects in remote sensing images can have quite different sizes. Therefore, a multiscale feature extraction network is desired. In this article, our feature extractor network is designed based on DarkNet-53 [36] and FPN [33]. The detailed network architecture can be found in [36]. For each input query image, our metafeature extractor network produces metafeatures at three different scales. Let $I^q \in \mathcal{Q}$ ($q \in \{1, 2, \ldots, N_q\}$) be one the input query image, and the generated metafeatures after the feature extractor network can be formulated as

$$F_i^q = \mathcal{F}_\theta(I) \in \mathbb{R}^{h_i \times w_i \times m_i} \tag{1}$$

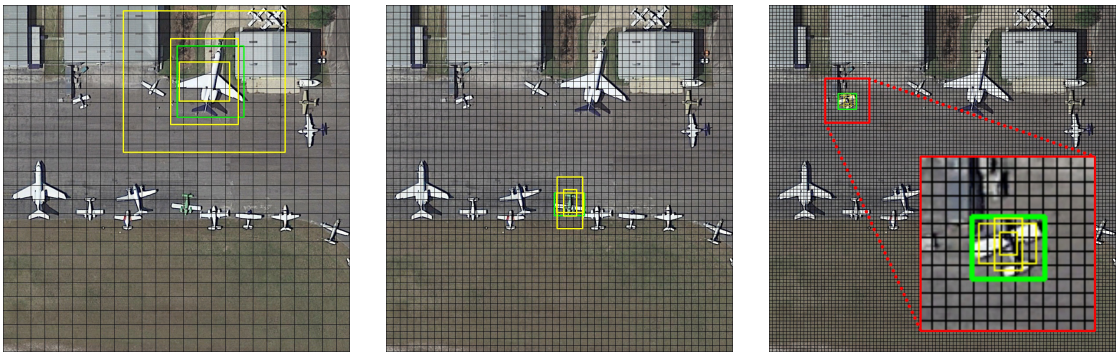
where $\mathcal{F}_\theta$ denotes the feature encoding network with parameters θ, i denotes the scale level, $i \in \{1, 2, 3\}$, and h_i, w_i, and m_i denote the sizes of feature maps at scale i.

In this article, we choose feature maps at the scales of 1/32×, 1/16×, and 1/8×, i.e., the output feature maps will have sizes of $(h/32 \times w/32 \times 1024)$, $(h/16 \times w/16 \times 512)$, and $(h/8 \times w/8 \times 256)$.

C. Feature Reweighting Module

Our feature reweighting module is designed to extract metaknowledge from the support images and to guide object detection in query images. To achieve this goal, a lightweight CNN is formulated to map each support image to a set of reweighting vectors, one for each scale. These reweighting vectors will be used to adjust the contribution of metafeatures and highlight metafeatures significant for novel object detection.

Assuming the support samples are from N object categories, our feature reweighting module receives inputs of $N \times K$ support images and their annotations. For each of the support classes c, where $c \in \{1, 2, \ldots, N\}$, K support image $\{I_{\text{ck}}\}_{k=1}^K$, along with its corresponding bounding box annotations $\{M_{\text{ck}}\}_{k=1}^K$, will be randomly chosen from the support set. Our feature reweight module first extracts per-object features using a feature encoding network $\mathcal{G}_\phi$ (with parameters ϕ) and then averages the feature vectors by object classes, resulting in class-specific representations $V_{ic} = \text{average}\{\mathcal{G}_\phi(I_{\text{ck}}, M_{\text{ck}})\}_{k=1}^K$,



(a) (b) (c)

Fig. 4. Anchor boxes on three feature maps at different scales. Green boxes represent ground-truth bounding boxes, and yellow boxes represent anchor boxes at different scales. The size of the input image is 800 × 800. (a)–(c) Anchor settings of its small (25 × 25), middle (50 × 50), and large (100 × 100) feature maps.

TABLE I
NETWORK ARCHITECTURE OF THE REWEIGHTING MODULE

Index	Type	Filters	Size	Output
1	Convolutional	32	3×3/1	512×512
2	Max-pooling		2×2/2	256×256
3	Convolutional	64	3×3/1	256×256
4	Max-pooling		2×2/2	128×128
5	Convolutional	128	3×3/1	128×128
6	Max-pooling		2×2/2	64×64
7	Convolutional	256	3×3/1	64×64
8	Max-pooling		2×2/2	32×32
9	Convolutional	256	3×3/1	32×32
10	GlobalMax		32×32/1	1×1
11	Route	8		
12	Convolutional	512	3×3/1	32×32
13	Max-pooling		2×2/2	16×16
14	Convolutional	512	3×3/1	16×16
15	GlobalMax		16×16/1	1×1
16	Route	13		
17	Convolutional	1024	3×3/1	16×16
18	Max-pooling		2×2/2	8×8
19	Convolutional	1024	3×3/1	8×8
20	GlobalMax		8×8/1	1×1

where $V_{ic} \in \mathbb{R}^{m_i}$, $i \in \{1, 2, 3\}$. The reweighting vector V_{ic} will be used to reweight the metafeatures and highlight the informative one at scale i and class c.

Table I shows the network architecture of our feature reweighting module $\mathcal{G}$. "Convolutional" denotes 2-D convolutional layer. "Filters" is the number of convolutional filters. "Size" indicates the spatial size and stride of a convolutional kernel in the form of "kernel height × kernel width/stride"; "Max-pooling" denotes max-pooling Layer; "GlobalMax" denotes global max-pooling layer where the kernel size equals the input size; and "Route" is a layer that used to control forward route. "Route" layer will take the output of layer in the "Filter" column as the input of the next layer. For example, "Route 8" in Layer 11 means taking the output of Layer 8 as the input of the next layer, i.e., Layer 12. The output reweighting vectors are taken from every global max-pooling layer (marked with an underline in Table I), and each reweighting vector has the same dimension as the corresponding metafeature.

After obtaining the metafeatures F_i^q and the reweighting vectors V_{ic}, we compute the class-specific reweighted feature maps $\widehat{F}_{ic}$ by

$$\widehat{F}_{ic}^q = F_i^q \otimes V_{ic}, \quad i = 1, 2, 3 \text{ and } c = 1, 2, \ldots, N \tag{2}$$

where $\otimes$ is channelwise multiplication that is realized through 1 × 1 convolution with the reweighting vectors V_{ic} as the convolution kernels.

As one can see, after channelwise multiplication, there will be three groups of reweighted feature maps, one for each scale. In each group, our feature reweighting module produces N reweighted feature maps. Each reweighted feature map is responsible for detecting objects for one of the N classes.

D. Bounding Box Prediction

Our bounding prediction module ($\mathcal{M}_\psi$) takes as input the reweighted feature maps and produces the object categories and bounding box locations. Following the setting of YOLOv3 [36], at each cell of the reweighted feature maps, we predict three bounding boxes for each of the reweighted features maps. To achieve this goal, we generate a set of anchor boxes at each pixel location on the reweighted feature maps. Fig. 4 illustrates the anchor box settings at three different scales. For the first set of feature maps with scale level i equals 1, the sizes of the anchor boxes are set to (116 × 90), (156 × 198), and (373 × 326). For the second set of feature maps with scale level i equals 2, the sizes of the anchor boxes are set to (30 × 61), (62 × 45), and (59 × 119) for the middle feature map. For the third set of feature maps with scale level i equals 3, the sizes of the anchor boxes are set to (10 × 13), (16 × 30), and (33 × 23). In our experiments, we set the aspect ratio of different anchor boxes to around 1:2, 1:1, and 2:1 and search the optimal base size in a predefined range.

For each anchor box in the reweighted feature maps, our bounding box prediction module produces a 6-D output, as displayed in Fig. 3. Among the output, the first four elements are used for object location prediction, and the left two elements are the objectness score o_p and the classification score R_p. Fig. 5 shows the output representation of each bounding box. Assuming the coordinates of a predicted bounding box are b_x, b_y, b_w, and b_h, where b_x and b_y are the coordinates of its center and b_w and b_h are the width and height of the bounding box. Instead of directly regressing the bounding box locations,

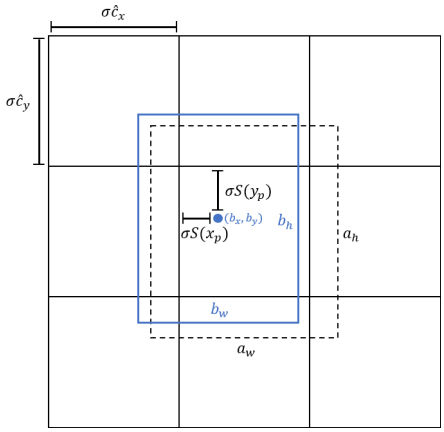


Fig. 5. Illustration of anchor box and predicted bounding box representations. The solid line grid represents the cells of the feature map, the dashed line rectangle represents an anchor box, and the blue line rectangle represents the predicted bounding box.

our bounding box prediction module predicts four offset values x_p, y_p, w_p, and h_p, and the coordinates of the predicted box can be computed through

$$\begin{aligned} b_x &= \alpha(\sigma(x_p) + \widehat{c}_x) \\ b_y &= \alpha(\sigma(y_p) + \widehat{c}_y) \\ b_w &= a_w e^{w_p} \\ b_h &= a_h e^{h_p} \end{aligned} \tag{3}$$

where $\widehat{c}_x$ and $\widehat{c}_y$ are cell offsets from the top left corner to the cell that makes the prediction; a_w and a_h are the width and height of the corresponding anchor box; $\sigma(\cdot)$ is the sigmoid function; and α is a scale transformation coefficient equivalents to the ratio between the input image side length and the feature map side length.

The objectness score (o_p) implies the possibility of the existences of an object, which can be computed as $\mathcal{P}_o = \sigma(o_p)$, where $\mathcal{P}_o$ is the possibility and $\sigma(\cdot)$ is the sigmoid function. Considering we have one set of reweighted feature maps for each class, each predicted bounding box only needs one score for a class prediction instead of the total number of categories (N). The classification score r_p indicates the possibility that the detected object belongs to each one of the N classes. Taking the classification scores generated from the same anchor boxes locations with the same anchor sizes as a group, there will be N classification scores belonging to the same anchor boxes of the input image. Naming these N predicted boxes as $\{r_p^c\}$ $(c = 1, 2, \ldots, N)$, a softmax function is applied on the probability vector to normalize these probability values. The final classification score for each class c can be formulated as

$$\mathcal{R}_c = \frac{e^{r_p^c}}{\sum_{c=1}^{N} e^{r_p^c}}. \tag{4}$$

$\mathcal{R}_c$ is the final classification possibility of class c, and $\sum_{c=1}^{N} \mathcal{R}_c = 1$. Objectness possibility and classification possibility together can help to judge whether an object is detected and which class the object belongs to.

E. Loss Function

The loss function of our object detection model contains two parts: object localization loss and object classification loss. For object localization, we use the mean-square-error loss to penalize the misalignment between the predicted bounding boxes and the ground-truth ones. Given the predicted bounding boxes coordinates coord_p and ground-truth bounding boxes coordinates coord_t, the object localization loss is calculated as

$$\mathcal{L}_{\text{loc}} = \frac{1}{N_{\text{pos}}} \sum_{\text{pos}} \sum_{l} \left(\text{coord}_t^l - \text{coord}_p^l\right)^2 \tag{5}$$

where l denotes a coordinate enumerator, it can be chosen from $\{w, y, w, h\}$, i.e., the four coordinate representations of a specific bounding box. *pos* indicates all positive anchors. Only losses of positive anchors are used in localization loss calculation, and localization losses of those negative anchor boxes are ignored. We identify an anchor box as positive if the IoU between this anchor box and a certain ground-truth bounding box is larger than a given threshold (e.g., 0.7). Also, we identify an anchor box as negative if the IoU between this anchor box and all ground-truth bounding boxes are less than a given threshold (e.g., 0.3). We also identify an anchor box as positive if the IoU between this anchor box and a certain ground-truth bounding box is greater than the IoUs between all other anchor boxes and this ground-truth bounding box.

The loss function for the objectness score $\mathcal{L}_{\text{obj}}$ is a binary cross-entropy loss, calculated as

$$\begin{aligned} \mathcal{L}_o &= \frac{1}{N_{\text{pos}}} \sum_{\text{pos}} -[\mathcal{P}_t \cdot \log \mathcal{P}_o + (1 - \mathcal{P}_t) \cdot \log(1 - \mathcal{P}_o)] \\ &= \frac{1}{N_{\text{pos}}} \sum_{\text{pos}} -\log \mathcal{P}_o \\ \mathcal{L}_{\text{noobj}} &= \frac{1}{N_{\text{neg}}} \sum_{\text{neg}} -[\mathcal{P}_t \cdot \log \mathcal{P}_o + (1 - \mathcal{P}_t) \cdot \log(1 - \mathcal{P}_o)] \\ &= \frac{1}{N_{\text{neg}}} \sum_{\text{neg}} -\log(1 - \mathcal{P}_o) \\ \mathcal{L}_{\text{obj}} &= w_{\text{obj}} \cdot \mathcal{L}_o + w_{\text{noobj}} \cdot \mathcal{L}_{\text{noobj}} \end{aligned} \tag{6}$$

where $\mathcal{P}_o$ denotes the predicted objectness possibility mentioned above; $\mathcal{P}_t$ denotes the true possibility that is one when it is a positive box and is zero when negative; and w_{obj} and w_{noobj} are weights of objectness loss and none-objectness loss. Considering there are usually more negative boxes than positive boxes, w_{obj} and w_{noobj} are used to balance these two loss terms.

For object classification, we use cross-entropy loss to enforce the predicted classes to be aligned with the ground-truth ones, calculated as

$$\mathcal{L}_{\text{cls}} = \frac{1}{N_{\text{pos}}} \sum_{\text{pos}} \sum_{c} -y_c \log \mathcal{R}_c \tag{7}$$

where y_c is the ground-truth classification label. Because we already use an objectness score to decide whether the predicted box contains an object or not, so the background class is

ignored during the classification loss calculation. The overall objective loss function is formulated as

$$\mathcal{L} = \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{obj}} + \mathcal{L}_{\text{cls}}. \tag{8}$$

F. Training and Inference

In our few-shot detection model, the training and inference processes are conducted with episode data. To facilitate model training in the few-shot detection scenario, we reorganize the training data set into two sets: the query set ($\mathcal{Q}$) and the support set ($\mathcal{S}$). The support set is a training data set regrouped by object classes. As explained in III-C, each query image is associated with a group of support images from all classes. Therefore, we separate training images into N groups according to object categories. After regrouping, a bounding box mask is generated for each support image. The mask is generated by setting the pixel value to 1 when the pixel is located within the ground-truth bounding box and 0 otherwise. Therefore, a support set can be formulated as

$$\mathcal{S} = \{(I_{\text{ck}}, M_{\text{ck}})\}, \quad c = 1, 2, \ldots N, \ k = 1, 2, \ldots, K. \tag{9}$$

Similarly, the query set contains a set of query images and their annotations

$$\mathcal{Q} = \{(I^q, A^q)\}_{q=1}^{N_q}. \tag{10}$$

We, therefore, formulate an episode $\mathcal{T}$ by

$$\begin{aligned} \mathcal{T} &= \mathcal{Q} \cup \mathcal{S} \\ &= \{(I^q, A^q)\} \cup \{(I_{\text{ck}}, M_{\text{ck}})\}. \end{aligned} \tag{11}$$

I^q and $\{(I_{\text{ck}}, M_{\text{ck}})\}$ are fed into the metafeature extractor network and feature reweighting module, respectively, while A^q is used as the ground-truth supervision during the training process.

Under the few-shot detection scenario, we need to leave some object classes in the data set for model evaluation. All classes in the data set are divided into seen classes and unseen classes. Seen classes require as many samples as possible to train a robust model, while unseen classes are viewed as a new detection task with only a few annotated samples.

The training process is divided into two steps. The first step is training on the seen classes to learn metaknowledge from the data set of seen classes. This process is also called meta-training. This step generally requires a large amount of training data, takes a relatively long time, and usually is not necessary to repeat in the following utilization. The second step is meta-fine-tuning on the unseen classes with a few samples to make our model well adapted to the unseen classes. This meta-fine-tuning process is fast and is going to be done whenever a new class is added. Both the metatraining and fine-tuning processes are conducted with episode data. The overall training and testing processes are illustrated in Algorithm 1. Our code will be released at https://github.com/lixiang-ucas/FSODM.

Algorithm 1 Training and Testing Process

```
Construct training set D_train from seen classes and testing set D_test from unseen classes.
Initialize the network parameters θ, φ, ψ in the feature extractor network, feature reweighting module, and bounding box prediction module.
for each training episode (S, Q) ∈ D_train do
    Model meta-training.
end for
for each training episode (S, Q) ∈ D_test do
    Model meta-finetuning.
end for
for each testing episode (S, Q) ∈ D_test do
    Extract feature maps for query images using Meta Feature Extractor.
    Generate class-specific reweighting vectors and compute the reweighted feature maps.
    Generate predicted bounding boxes using the bounding box prediction modules.
end for
```

IV. Experiments and Results

In this section, we evaluate the performance of our model for the few-shot object detection on two public benchmark RSI data sets and compare our method with both conventional object detectors and few-shot detectors to show the superiority of our model.

A. Data Set

NWPU VHR-10 is a very high resolution (VHR) remote sensing image data set released by [68]. This data set contains 800 RSIs collected from Google Earth and the ISPRS Vaihingen data set [69]. One hundred and fifty "negative samples" without target objects and 650 "positive samples" with at least one object are annotated manually. There are in total ten object categories in this data set: airplane, baseball diamond, basketball, bridge, court, ground track field, harbor, ship, storage tank, tennis court, and vehicle.

DIOR is a large-scale benchmark data set for object detection on RSIs, released by [2]. Images in the DIOR data set are collected from Google Earth with 23 463 images and 192 472 instances of 20 classes. The object classes include airplane, airport, baseball field, basketball court, bridge, chimney, dam, expressway service area, expressway toll station, harbor, golf course, ground track field, overpass, ship, stadium, storage tank, tennis court, train station, vehicle, and windmill. All images are of the size 800×800 pixels, and the spatial resolutions range from 0.5 to 30 m. In the DIOR data set, the object sizes vary widely.

B. Experimental Settings

To evaluate the detection performance of our FSODM model under the few-shot scenario, we divide each data set into two parts: one is constructed from the seen classes, the other is constructed from the unseen classes. For the NWPU VHR-10 data set, three classes (airplane, baseball diamond, and

TABLE II
FEW-SHOT DETECTION PERFORMANCE (MAP) ON THE UNSEEN CLASSES OF THE NWPU VHR-10 DATA SET. THE TOP PART SHOWS CONVENTIONAL OBJECT DETECTORS, AND THE BOTTOM PART SHOWS FEW-SHOT LEARNING-BASED METHODS

Method	shot	airplane	baseball diamond	tennis court	mean
Faster R-CNN(ResNet101) [7]	10	0.12	0.57	0.12	0.24
	20	0.24	0.62	0.15	0.34
Faster R-CNN(VGG16) [7]	10	0.03	0.15	0.02	0.06
	20	0.18	0.27	0.01	0.15
YOLOv3 [36]	10	0.14	0.26	0.01	0.14
	20	0.30	0.50	0.03	0.28
RepMet [17]	3	0.19	0.36	0.12	0.23
	5	0.20	0.36	0.14	0.23
	10	0.22	0.39	0.18	0.26
YOLO-Low-shot [19]	3	0.13	0.12	0.11	0.12
	5	0.24	0.39	0.11	0.24
	10	0.20	0.74	0.26	0.40
FSODM (Ours)	3	0.15	0.57	0.25	0.32
	5	0.58	0.84	0.16	0.53
	10	0.60	0.88	0.48	0.65

tennis court) are used as unseen classes and the others as seen classes. For the DIOR data set, five classes (airplane, baseball field, tennis court, train station, and windmill) are chosen as unseen classes and the others as seen classes. In our experiments, we randomly choose the seen/unseen classes. It should be noted that different seen/unseen splits can have different final detection performance, but, in this article, we aim to introduce the first few-shot learning-based method for object detection on remote sensing images but not to achieve state-of-the-art performance. Note that one image can contain several object instances; the number of shots in our experiments means the number of object instances, not the number of images.

Moreover, we apply a multiscale training technique process to enhance detection performance. The scale range of input images varies in (384, 416, 448, 480, 512, 544, 576, 608, and 640), and all input images are square. We note that, in the DIOR data set, the original images are much larger than the desired input scales. Therefore, those large images are cropped into a series of patches with 1024×1024 pixels and a stride of 512 pixels (for the DIOR data set, this step is ignored). For the objects that get truncated in this process, we ignore these truncated object instances that have an overlapping of less than 70% with the original object instances.

C. Comparing Methods

We compare our FSODM model with the prevalent one-stage object detector YOLOv3 [36] and two-stage object detector Faster R-CNN [7] (Faster R-CNN includes ResNet101 and VGG16 two type of feature extraction networks). We train these conventional object detectors using transfer learning techniques, where the training process of these conventional object detectors consists of two steps: pretraining and model fine-tuning. In the pretraining stage, we remove all objects belonging to the unseen classes from the training data and train a conventional none few-shot model; in the fine-tuning stage, we train the model with a few annotated samples from the unseen classes. Note that these conventional detectors use complicated data augmentation strategies to enhance their performance; in our experiments, we do not implement these strategies in order to have a fair comparison with our method. There are only a few methods focused on the problem of few-shot detection, so we only include two current state-of-the-art few-shot detectors of YOLO-Low-Shot [19] and RepMet [17]. For [19], the experimental settings are the same as our method: training on the same set from the seen classes and fine-tuning on the same set from the unseen classes.

The mean average precision (mAP) is used to evaluate object detection performance. We follow the PASCAL VOC2007 benchmark [70] to calculate the mAP, which takes the average of 11 precision values when recall increases from 0 to 1 with a step of 0.1.

D. Results on NWPU VHR-10

Table II lists the few-shot object detection performance of our FSODM method and the comparison methods on the unseen classes of the NWPU VHR-10 data set. In this table, we show the performance under different numbers of shots (i.e., annotated samples in unseen classes). As shown in Table II, our proposed FSODM model achieves significantly better performance than all comparing methods. More specifically, compared to the current state-of-the-art few-shot object detector [19], our method obtains an mAP 166.6% higher in the three-shot setting, 120.8% higher in the five-shot setting, and 62.5% higher in the ten-shot setting. The conventional none few-shot-based methods (Faster RCNN and YOLOv3) obtain a lot worse performance than the two few-shot-based methods. Even in the ten-shot setting, Faster RCNN w ResNet101 only gets an mAP of 0.24, which is worse than our FSODM model in the three-shot setting with an mAP of 0.32. Moreover, as shown in Table II, with increases in the number of annotated samples in the unseen classes, the detection performance of our FSODM model increases quickly.

From Table II, one can also see that both our FSODM model and the comparison methods obtain better performance on the 'baseball diamond' category. This is because baseball

TABLE III

FEW-SHOT DETECTION PERFORMANCE (MAP) ON THE UNSEEN CLASSES OF THE DIOR DATA SET. THE TOP PART SHOWS CONVENTIONAL OBJECT DETECTORS, AND THE BOTTOM PART SHOWS FEW-SHOT LEARNING-BASED METHODS

Method	shot	airplane	baseball field	tennis court	train station	wind mill	mean
Faster R-CNN(ResNet101) [7]	10	0.09	0.30	0.31	0.06	0.09	0.17
	20	0.12	0.49	0.39	0.16	0.10	0.25
	30	0.12	0.50	0.41	0.19	0.17	0.28
Faster R-CNN(VGG16) [7]	10	0.01	0.07	0.09	0.00	0.00	0.03
	20	0.09	0.27	0.11	0.05	0.00	0.10
	30	0.09	0.31	0.18	0.03	0.02	0.13
YOLOv3 [36]	10	0.02	0.32	0.29	0.01	0.04	0.14
	20	0.07	0.36	0.40	0.05	0.12	0.20
	30	0.09	0.45	0.42	0.08	0.21	0.25
RepMet [17]	5	0.09	0.19	0.11	0.01	0.01	0.08
	10	0.13	0.33	0.24	0.01	0.01	0.14
	20	0.14	0.34	0.29	0.03	0.03	0.16
YOLO-Low-Shot [19]	5	0.09	0.33	0.47	0.09	0.13	0.22
	10	0.15	0.45	0.54	0.07	0.18	0.28
	20	0.19	0.52	0.55	0.18	0.26	0.34
FSODM (Ours)	5	0.09	0.27	0.57	0.11	0.19	0.25
	10	0.16	0.46	0.60	0.14	0.24	0.32
	20	0.22	0.50	0.66	0.16	0.29	0.36

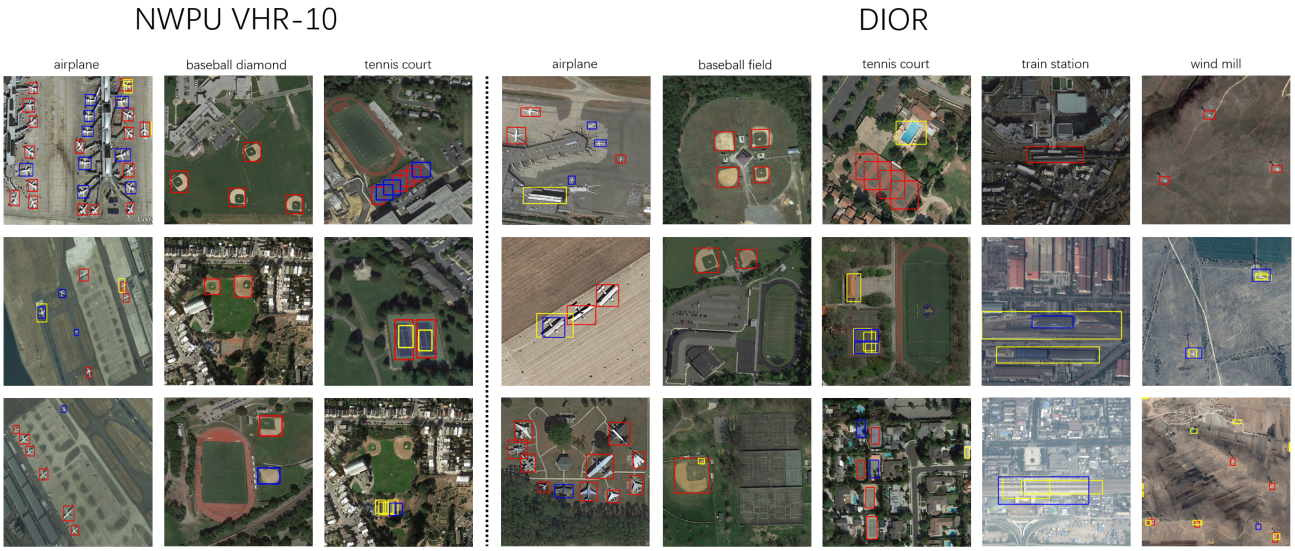


Fig. 6. Selected examples of our few-shot detection results. (Left) Detection results on the unseen classes of the NWPU VHR data set in a ten-shot setting. (Right) Detection results on the unseen classes of the DIOR data set in a 20-shot setting. Red, yellow, and blue boxes indicate true positive, false positive, and false negative detection, respectively.

diamonds have smaller size variations, which reduces the recognition challenges for a detection model.

E. Results on DIOR

Considering the DIOR data set is a large scale data set with large variations in object structures and sizes, a larger number of annotated samples are used for the unseen classes. Specifically, for conventional object detector (Faster RCNN and YOLOv3), we conduct experiments with 10, 20, and 30 annotated samples for each of the unseen classes. Table III shows the quantitative results of our method and the comparing methods on the unseen classes of the DIOR data set.

As shown in Table III, our FSODM model achieves better performance than other two few-shot-based methods presented in [17] and [19]. All these three few-shot-based methods achieve a lot better performance than the none few-shot-based methods (Faster R-CNN and YOLOv3), even with fewer samples. Moreover, with the increase in the number of annotated samples in unseen classes, the detection performance improves consistently for all methods. Table III also shows that the "baseball field" and "tennis court" categories reach better detection performance. This is probably because these two object categories have smaller in-category variations.

Fig. 6 shows some examples of the few-shot detection results of our FSODM model on the NWPU VHR-10 data set and the DIOR data set. As shown in Fig. 6, our model can successfully detect most of the objects in all the unseen classes of the NWPU VHR-10 and the DIOR data sets. Most of the failure cases come from the missing or false detection of small objects. Moreover, with only a few annotated samples of the unseen classes, our model fails to accurately localize "train stations" with large sizes or appearance variations.

TABLE IV
DETECTION PERFORMANCE (MAP) ON THE SEEN CLASSES OF THE NWPU VHR-10 DATA SET

Class	FSODM (Ours)	YOLO-Low-shot [19]	YOLOv3 [36]	Faster R-CNN (ResNet101) [7]	Faster R-CNN (VGG16) [7]
ship	0.72	0.77	0.71	0.88	0.69
storage tank	0.71	0.80	0.68	0.49	0.23
basketball court	0.72	0.51	0.62	0.56	0.43
ground track field	0.91	0.94	0.94	1.00	0.87
harbor	0.87	0.86	0.84	0.66	0.42
bridge	0.76	0.77	0.80	0.57	0.14
vehicle	0.76	0.68	0.77	0.74	0.33
mean	0.78	0.76	0.77	0.70	0.44

TABLE V
DETECTION PERFORMANCE (MAP) ON THE SEEN CLASSES OF THE DIOR DATA SET

Class	FSODM (Ours)	YOLO-Low-shot [19]	YOLOv3 [36]	Faster R-CNN (ResNet101) [7]	Faster R-CNN (VGG16) [7]
airport	0.63	0.59	0.59	0.73	0.40
basketball court	0.80	0.74	0.83	0.69	0.46
bridge	0.32	0.29	0.28	0.26	0.17
chimney	0.72	0.70	0.68	0.72	0.70
dam	0.45	0.52	0.39	0.57	0.35
expressway service area	0.63	0.63	0.68	0.59	0.48
expressway toll station	0.60	0.48	0.57	0.45	0.33
golf course	0.61	0.61	0.63	0.68	0.53
ground track field	0.61	0.54	0.70	0.65	0.41
harbor	0.43	0.52	0.43	0.31	0.16
overpass	0.46	0.49	0.43	0.45	0.37
ship	0.50	0.33	0.64	0.10	0.09
stadium	0.45	0.52	0.43	0.67	0.45
storage tank	0.43	0.26	0.46	0.21	0.16
vehicle	0.39	0.29	0.41	0.19	0.16
mean	0.54	0.50	0.54	0.48	0.35

V. DISCUSSION

A. Detection Performance on Seen Classes

A good few-shot object detection model should not only perform well on the unseen classes with few annotated samples but also should not sacrifice the performance on seen classes, which means that it should also perform the conventional none few-shot-based models when data are abundant.

Tables IV and V show the performances of our FSODM model and the comparison methods on the seen classes of the NWPU VHR-10 and DIOR data sets. From Table IV, one can see that all three methods achieve similar performances on the seen classes, with slight differences in the mAP values. On the DIOR data set, our method performs better than another few-shot-based method [19]. This demonstrates that our proposed method can better maintain the performance on the seen classes under the few-shot detection scenario. The performance of our few-shot-based method achieves the same mAP value as the conventional YOLOv3 detection when a large amount of data is provided.

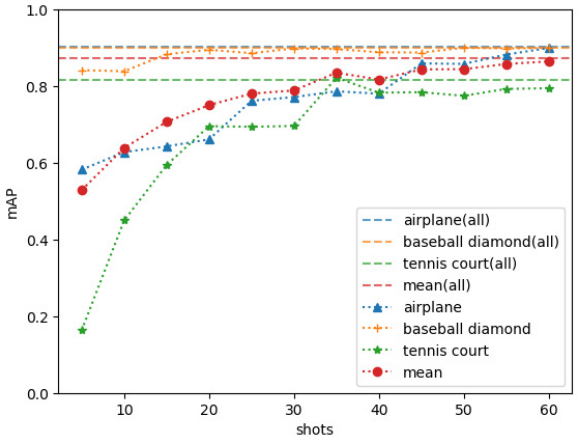
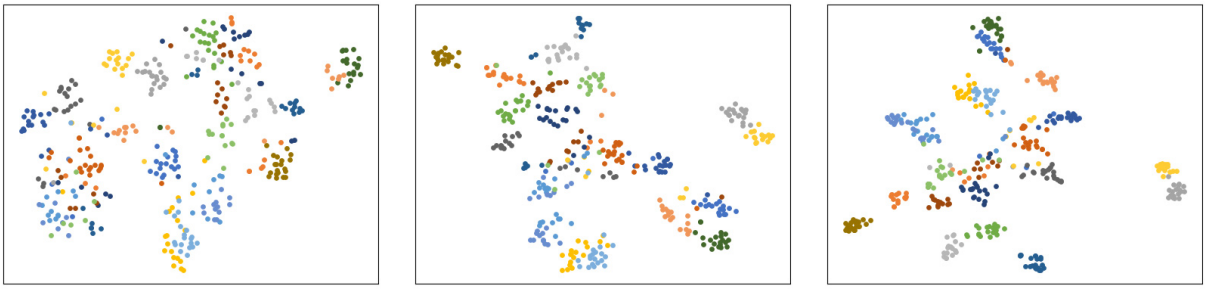


Fig. 7. Detection performance with different shots on the NWPU VHR-10 data set. Horizontal dash lines indicate the baseline performances generated using all the training samples in the data set (1025 samples for the airplane category, 519 samples for the baseball diamond category, and 643 samples for the tennis court category).

B. Number of Shots

We investigate the performance of our few-shot object detection model under different numbers of shots on the unseen classes. To show the advantage of our few-shot based model, we conduct comparing experiments with all training samples from the novel categories of the NWPU VHR-10 data set and use YOLOv3 as the baseline model. For our FSODM model, we conduct experiments with a larger few-shot range (from five shots to 60 shots). As shown in Fig. 7, our model with only 60 (about 8%) training samples from the novel categories can achieve almost the same detection performance as the baseline model that uses all the training samples. We attribute this to the fact that our FSODM model can learn metaknowledge from the seen classes and effectively apply it for detection on the unseen classes. Moreover, in the baseball



(a) (b) (c)

Fig. 8. t-SNE [71] visualization of reweighting vectors. The reweighting vectors are generated from 400 support images randomly picked from the DIOR data set (20 images of each category). (a)–(c) Visualizations of reweighting vectors with dimensions of 256, 512, and 1024, respectively.

TABLE VI
DETECTION SPEED OF THE COMPARING METHODS

Method	FPS
Faster R-CNN (ResNet101) [7]	6.17
Faster R-CNN (VGG16) [7]	8.39
YOLOv3 [36]	43.95
RepMet [17]	1.75
YOLO-Low-Shot [19]	120.23
FSODM (Ours)	52.91

diamond class, our model with only 20 annotated samples achieves almost the same performance as the baseline model that uses all the training samples. This is probably because baseball diamonds have smaller in-category variations and are easily identified by their structures from a few annotated samples. In contrast, although the airplane class has almost the same detection performance as the baseball diamond class using the baseline model, the few-shot detection performance is significantly worse. This is because objects in the airplane category have larger structural and size variations, as shown in Fig. 6. This is a challenge that impedes our model from achieving a satisfying performance with only a few samples (less than 60) even though our few-shot-based model can successfully obtain a comparable performance as the baseline model when enough annotated samples (60 shots) are given.

C. Reweighting Vectors

In our approach, the reweighting vectors are extracted by the reweighting module and significantly support the final detection performance. To explore the relationship between these reweighting vectors, we use t-Distributed Stochastic Neighbor Embedding (t-SNE) [71] to reduce their dimensions and visualize them on the coordinate axis. T-SNE is a dimensionality reduction technique that can pass the inner relationship between high dimension vectors to low-dimensional vectors. It keeps close vectors in high-dimensional space close in low-dimensional space and remote vectors in high-dimensional space remote in low-dimensional space.

Fig. 8 shows some examples of visualized reweighting vectors. In the figure, reweighting vectors from the same categories tend to aggregate together, which suggests the learned reweighting vectors successfully characterize the object class information from the original support masks. In addition, the clustering results in Fig. 8(c) are obviously better than the results in Fig. 8(a) and (b). The reason is that the more elements a reweighting vector has, the more information it carries. Therefore, reweighting vectors with higher dimensions tends to be more capable of representing the object information from support samples.

D. Detection Speed

In this section, we explore the detection speed of our FSODM model. We compare the detection speed of our model with Faster RCNN, YOLOv3, RepMet, and YOLO-Low-shot. Experiments are carried out on the NWPU data set in a three-way-ten-shot setting. We report the inference speed on a Tesla P100 GPU with the batch size set to 1. Results are listed in Table VI. From Table VI, one can see that our FSODM model obtains a detection speed comparable with YOLOv3 and is a lot faster than the two Faster R-CNN methods and another few-shot-based method RepMet. YOLO-Low-Shot is about 2× faster than our method because it uses a single-scale detection framework, while our FSODM model adopts a multiscale detection pipeline and can get better performance.

VI. CONCLUSION

This article introduces a new metalearning-based method for few-shot object detection on remote sensing images that are among the first method to challenge this area of research. We first formulate the few-shot object detection problem on remote sensing images. Then, we introduce our proposed method that includes three main components: a metafeature extractor, a feature reweighting module, and a bounding box prediction module. Each module is designed in a multiscale architecture to enable multiscale object detection. Our method is trained with large-scale data from some seen classes and can learn metaknowledge from seen classes and generalizes well to unseen classes with only a few samples. Experiments on two public benchmark data sets demonstrate the powerful ability of our method for detecting objects from unseen classes through a few annotated samples. This work is the very first step in the few-shot detection in the remote sensing field, and we will further improve it and keep exploring in this field.

REFERENCES

[1] G. Cheng and J. Han, "A survey on object detection in optical remote sensing images," *ISPRS J. Photogramm. Remote Sens.*, vol. 117, pp. 11–28, Jul. 2016.

[2] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS J. Photogramm. Remote Sens.*, vol. 159, pp. 296–307, Jan. 2020.

[3] X. Bai, H. Zhang, and J. Zhou, "VHR object detection based on structural feature extraction and query expansion," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 10, pp. 6508–6520, Oct. 2014.

[4] F. Bi, B. Zhu, L. Gao, and M. Bian, "A visual search inspired computational model for ship detection in optical satellite images," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 4, pp. 749–753, Jul. 2012.

[5] X. Huang and L. Zhang, "Road centreline extraction from high-resolution imagery based on multiscale structural features and support vector machines," *Int. J. Remote Sens.*, vol. 30, no. 8, pp. 1977–1987, Apr. 2009.

[6] M. Volpi, F. D. Morsier, G. Camps-Valls, M. Kanevski, and D. Tuia, "Multi-sensor change detection based on nonlinear canonical correlations," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2013, pp. 1944–1947.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[9] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[10] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3630–3638.

[11] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4077–4087.

[12] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4367–4375.

[13] C. Zhang, G. Lin, F. Liu, R. Yao, and C. Shen, "CANet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5217–5226.

[14] K. Wang, J. H. Liew, Y. Zou, D. Zhou, and J. Feng, "PANet: Few-shot image semantic segmentation with prototype alignment," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9197–9206.

[15] T. Hu, P. Yang, Z. Chiliang, G. Yu, Y. Mu, and C. Snoek, "Attention-based multi-context guiding for few-shot semantic segmentation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 8441–8448.

[16] M. Dixit, R. Kwitt, M. Niethammer, and N. Vasconcelos, "AGA: Attribute-guided augmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7455–7463.

[17] L. Karlinsky *et al.*, "RepMet: Representative-based metric learning for classification and few-shot object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5197–5206.

[18] T. Wang, X. Zhang, L. Yuan, and J. Feng, "Few-shot adaptive faster R-CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7173–7182.

[19] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8420–8429.

[20] Z. Chen, K. Yin, M. Fisher, S. Chaudhuri, and H. Zhang, "BAE-NET: Branched autoencoder for shape co-segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8490–8499.

[21] L. Wang, X. Li, and Y. Fang, "Few-shot learning of part-specific probability space for 3D shape segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4504–4513.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[24] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[25] Q. Wang, J. Gao, and X. Li, "Weakly supervised adversarial domain adaptation for semantic segmentation in urban scenes," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4376–4386, Sep. 2019.

[26] Y. Hu, X. Li, N. Zhou, L. Yang, L. Peng, and S. Xiao, "A sample update-based convolutional neural network framework for object detection in large-area remote sensing images," *IEEE Geosci. Remote Sens. Lett.*, vol. 16, no. 6, pp. 947–951, Jun. 2019.

[27] I. Rocco, R. Arandjelovic, and J. Sivic, "Convolutional neural network architecture for geometric matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6148–6157.

[28] J. Chen, L. Wang, X. Li, and Y. Fang, "Arbicon-Net: Arbitrary continuous geometric transformation networks for image registration," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 3410–3420.

[29] X. Li, C. Wen, L. Wang, and Y. Fang, "Topology-constrained shape correspondence," *IEEE Trans. Vis. Comput. Graphics*, early access, May 11, 2020, doi: 10.1109/TVCG.2020.2994013.

[30] X. Li, L. Wang, M. Wang, C. Wen, and Y. Fang, "DANCE-NET: Density-aware convolution networks with context encoding for airborne LiDAR point cloud classification," *ISPRS J. Photogramm. Remote Sens.*, vol. 166, pp. 128–139, Aug. 2020.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

[32] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.

[33] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

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

[35] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[36] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[37] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional single shot detector," 2017, *arXiv:1701.06659*. [Online]. Available: http://arxiv.org/abs/1701.06659

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

[39] D. Chaudhuri, N. K. Kushwaha, and A. Samal, "Semi-automated road detection from high resolution satellite images by directional morphological enhancement and segmentation techniques," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 5, no. 5, pp. 1538–1544, Oct. 2012.

[40] D. M. McKeown and J. L. Denlinger, "Cooperative methods for road tracking in aerial imagery," in *Proc. CVPR, Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1988, pp. 662–672.

[41] J. Zhou, W. F. Bischof, and T. Caelli, "Road tracking in aerial images based on human–computer interaction and Bayesian filtering," *ISPRS J. Photogramm. Remote Sens.*, vol. 61, no. 2, pp. 108–124, Nov. 2006.

[42] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 67–92, Jan. 1973.

[43] A. Huertas and R. Nevatia, "Detecting buildings in aerial images," *Comput. Vis., Graph., Image Process.*, vol. 41, no. 2, pp. 131–152, Feb. 1988.

[44] J. C. McGlone and J. A. Shufelt, "Projective and object space geometry 982 for monocular building extraction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 54–61.

[45] J. C. Trinder and Y. Wang, "Automatic road extraction from aerial images," *Digit. Signal Process.*, vol. 8, no. 4, pp. 215–224, Oct. 1998.

[46] U. Weidner and W. Förstner, "Towards automatic building extraction from high-resolution digital elevation models," *ISPRS J. Photogramm. Remote Sens.*, vol. 50, no. 4, pp. 38–49, Aug. 1995.

[47] H. G. Akcay and S. Aksoy, "Building detection using directional spatial constraints," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, Jul. 2010, pp. 1932–1935.

[48] R. B. Irvin and D. M. McKeown, "Methods for exploiting the relationship between buildings and their shadows in aerial imagery," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 6, pp. 1564–1575, Nov./Dec. 1989.

[49] G. Cheng, P. Zhou, and J. Han, "Learning rotation-invariant convolutional neural networks for object detection in VHR optical remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 12, pp. 7405–7415, Dec. 2016.

[50] Z. Deng, H. Sun, S. Zhou, J. Zhao, and H. Zou, "Toward fast and accurate vehicle detection in aerial images using coupled region-based convolutional neural networks," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 10, no. 8, pp. 3652–3664, Aug. 2017.
[51] T. Tang, S. Zhou, Z. Deng, H. Zou, and L. Lei, "Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining," *Sensors*, vol. 17, no. 2, p. 336, Feb. 2017.
[52] Y. Yang, Y. Zhuang, F. Bi, H. Shi, and Y. Xie, "M-FCN: Effective fully convolutional network-based airplane detection framework," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 8, pp. 1293–1297, Aug. 2017.
[53] K. Li, G. Cheng, S. Bu, and X. You, "Rotation-insensitive and context-augmented object detection in remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 4, pp. 2337–2348, Apr. 2018.
[54] Y. Zhong, X. Han, and L. Zhang, "Multi-class geospatial object detection based on a position-sensitive balancing framework for high spatial resolution remote sensing imagery," *ISPRS J. Photogramm. Remote Sens.*, vol. 138, pp. 281–294, Apr. 2018.
[55] W. Guo, W. Yang, H. Zhang, and G. Hua, "Geospatial object detection in high resolution satellite images based on multi-scale convolutional neural network," *Remote Sens.*, vol. 10, no. 1, p. 131, Jan. 2018.
[56] J. Yang, Y. Zhu, B. Jiang, L. Gao, L. Xiao, and Z. Zheng, "Aircraft detection in remote sensing images based on a deep residual network and super-vector coding," *Remote Sens. Lett.*, vol. 9, no. 3, pp. 228–236, Mar. 2018.
[57] Y. Zhang, Y. Yuan, Y. Feng, and X. Lu, "Hierarchical and robust convolutional neural network for very high-resolution remote sensing object detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5535–5548, Aug. 2019.
[58] Z. Zou and Z. Shi, "Ship detection in spaceborne optical image with SVD networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 5832–5845, Oct. 2016.
[59] H. Lin, Z. Shi, and Z. Zou, "Fully convolutional network with task partitioning for inshore ship detection in optical remote sensing images," *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1665–1669, Oct. 2017.
[60] W. Liu, L. Ma, and H. Chen, "Arbitrary-oriented ship detection framework in optical remote-sensing images," *IEEE Geosci. Remote Sens. Lett.*, vol. 15, no. 6, pp. 937–941, Jun. 2018.
[61] T. Tang, S. Zhou, Z. Deng, L. Lei, and H. Zou, "Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks," *Remote Sens.*, vol. 9, no. 11, p. 1170, Nov. 2017.
[62] L. Liu, Z. Pan, and B. Lei, "Learning a rotation invariant detector with rotatable bounding box," 2017, *arXiv:1711.09405*. [Online]. Available: http://arxiv.org/abs/1711.09405
[63] J. Zhong, T. Lei, and G. Yao, "Robust vehicle detection in aerial images based on cascaded convolutional neural networks," *Sensors*, vol. 17, no. 12, p. 2720, Nov. 2017.
[64] X. Han, Y. Zhong, and L. Zhang, "An efficient and robust integrated geospatial object detection framework for high spatial resolution remote sensing imagery," *Remote Sens.*, vol. 9, no. 7, p. 666, Jun. 2017.
[65] Z. Xu, X. Xu, L. Wang, R. Yang, and F. Pu, "Deformable ConvNet with aspect ratio constrained NMS for object detection in remote sensing imagery," *Remote Sens.*, vol. 9, no. 12, p. 1312, Dec. 2017.
[66] H. Chen, Y. Wang, G. Wang, and Y. Qiao, "LSTD: A low-shot transfer detector for object detection," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–8.
[67] X. Dong, L. Zheng, F. Ma, Y. Yang, and D. Meng, "Few-example object detection with model communication," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1641–1654, Jul. 2019.
[68] G. Cheng, J. Han, P. Zhou, and L. Guo, "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *ISPRS J. Photogramm. Remote Sens.*, vol. 98, pp. 119–132, Dec. 2014.
[69] J. Niemeyer, F. Rottensteiner, and U. Soergel, "Contextual classification of lidar data and building object detection in urban areas," *ISPRS J. Photogramm. Remote Sens.*, vol. 87, pp. 152–165, Jan. 2014.
[70] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisser-Man, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.
[71] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

Xiang Li received the B.S. degree in remote sensing science and technology from Wuhan University, Wuhan, China, in 2014, and the Ph.D. degree from the Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, Beijing, China, in 2019.

He is a Post-Doctoral Associate with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE. His research interests include deep learning, computer vision, and remote sensing image recognition.

Jingyu Deng received the B.S. degree in IoT engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2018, and the M.S. degree in computer engineering from the NYU Tandon School of Engineering, New York, NY, USA, in 2020.

His research interests include deep learning and computer vision.

Yi Fang (Member, IEEE) received the B.S. and M.S. degrees in biomedical engineering from Xi'an Jiaotong University, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in mechanical engineering from Purdue University, West Lafayette, IN, USA, in 2011.

He is an Assistant Professor with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, UAE. His research interests include 3-D computer vision and pattern recognition, large-scale visual computing, deep visual computing, deep cross-domain, and cross-modality multimedia analysis, and computational structural biology.