$\mathcal{R}^2$-CNN: Fast Tiny Object Detection in Large-Scale Remote Sensing Images

Jiangmiao Pang, Cong Li, Jianping Shi, Zhihai Xu, Huajun Feng

***Abstract*— Recently, the convolutional neural network has brought impressive improvements for object detection. However, detecting tiny objects in large-scale remote sensing images still remains challenging. First, the extreme large input size makes the existing object detection solutions too slow for practical use. Second, the massive and complex backgrounds cause serious false alarms. Moreover, the ultratiny objects increase the difficulty of accurate detection. To tackle these problems, we propose a unified and self-reinforced network called remote sensing region-based convolutional neural network ($\mathcal{R}^2$-CNN), composing of backbone Tiny-Net, intermediate global attention block, and final classifier and detector. Tiny-Net is a lightweight residual structure, which enables fast and powerful features extraction from inputs. Global attention block is built upon Tiny-Net to inhibit false positives. Classifier is then used to predict the existence of target in each patch, and detector is followed to locate them accurately if available. The classifier and detector are mutually reinforced with end-to-end training, which further speed up the process and avoid false alarms. Effectiveness of $\mathcal{R}^2$-CNN is validated on hundreds of GF-1 images and GF-2 images that are 18 000 × 18 192 pixels, 2.0-m resolution, and 27 620 × 29 200 pixels, 0.8-m resolution, respectively. Specifically, we can process a GF-1 image in 29.4 s on Titian X just with single thread. According to our knowledge, no previous solution can detect the tiny object on such huge remote sensing images gracefully. We believe that it is a significant step toward practical real-time remote sensing systems.**

***Index Terms*— Object detection, remote sensing images, remote sensing region-based convolutional neural network ($\mathcal{R}^2$-CNN).**

I. Introduction

THANKS to the development of optical remote sensing imaging technology, high-resolution images can be easily obtained, which help us understand the earth better. Object detection, change detection, semantic segmentation, and other tasks become popular in the remote sensing area.

Han *et al.* [1], Long *et al.* [2], Bai *et al.* [3], Zhang *et al.* [4], and Lei *et al.* [5] propose different approaches for object detection in remote sensing images with the powerful feature extraction capability of deep convolutional neural networks.

Manuscript received May 27, 2018; revised October 5, 2018 and December 12, 2018; accepted January 28, 2019. This work was supported by the Basic Research under Grant ID JCKY2018110C081. *(Corresponding author: Huajun Feng.)*

J. Pang, Z. Xu, and H. Feng are with the State Key Laboratory of Modern Optical Instrumentation, Zhejiang University, Hangzhou 310027, China, and also with the College of Optical Science and Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: pjm@zju.edu.cn; xuzh@zju.edu.cn; fenghj@zju.edu.cn).

C. Li and J. Shi are with SenseTime Research, Beijing 100084, China (e-mail: licong@sensetime.com; shijianping@sensetime.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2019.2899955

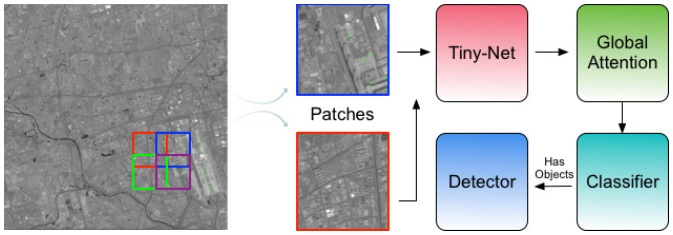

Fig. 1. $\mathcal{R}^2$-CNN is a unified network working in a patchwise manner. Pipeline is shown on the right.

However, those methods mainly focus on small region segments comparing to the original large inputs, e.g., usually over than 20 000 × 20 000 pixels. Therefore, they cannot scale up to handle such huge images gracefully. Zhang *et al.* [4] attempted to detect airports at first to speed up the process in large-scale images, but the training and testing images are all from the regions near to airports, which escape from the complex backgrounds. According to our experiments, it is not robust enough for practical applications.

Object detection in large-scale remote sensing images is pretty challenging. First, the scale of the input image is too large to reach the practical application. The computation time and memory consumption are increased quadratically, making it too slow and not runnable on current hardware. Second, massive and complex backgrounds that appear in a real scenario may introduce more false positives, for instance, desert region with random texture or urban area with massive building structure. Moreover, the performance drops drastically with tiny objects (such as 8–32 pixels), especially in the low-resolution images, which further increases the difficulty of tiny object detection in remote sensing images.

To tackle these problems, we propose a unified and self-reinforced convolutional neural network called remote sensing region-based convolutional neural network ($\mathcal{R}^2$-CNN), which is composed of the backbone Tiny-Net, intermediate global attention block, and final classifier and detector, enabling the entire network *efficient* in both computation and memory consumption, *robust* to false positives, and *strong* to detect tiny objects. Pipeline is shown in Fig 1.

First, as a unified and self-reinforced framework, $\mathcal{R}^2$-CNN first crops large-scale images with a much more smaller scale (such as 640 × 640 pixels) with 20% overlap to tackle the oversized input size. By processing the patches asynchronously, the limited memory is not a problem anymore. A convolutional backbone structure is then applied to inputs, which enables powerful features extraction. Based on the discriminative

features, a classifier first predicts the existence of detection target in the current patch, and a detector is followed to locate them accurately if available. The classifier and detector are mutually reinforced each other under the end-to-end training framework. There are two advantages of this self-reinforced architecture as follows.

1) Since, in large-scale remote sensing images, most crops do not contain valid target so that about 99% of the total patches do not need to pass the heavy detector branch. The light classifier branch can filter out a blank patch without heavier detector cost.
2) As most false positives commonly occur with massive backgrounds, benefited from the self-reinforced framework, the classifier can identify the difficult situation even when there is only one tiny object in the patch given the fine-grained features from the detector. On the other hand, the detector receives less false positive candidates since most of them are filtered out by the classifier. Even if the patches are distinguished incorrectly by a classifier, the detector can still rectify the results later.

Second, we specially designed a lightweight residual network called Tiny-Net to reduce the inference cost and preserve powerful features for object detection. Tiny-Net is motivated by [6] but is much more lightweight. On the other hand, Tiny-Net can be trained from scratch with a cycle training schedule because of fewer parameters, making that the framework does not be influenced by the limited training samplers and the domain gap between natural images and remote sensing images.

Third, to further inhibit the false positives, we also use feature pyramid pooling as a global attention block on the top of Tiny-Net. The feature maps are first pooled in different pyramid levels, such as 1×1, 2×2, and 4×4. Then, we recover the pooled features to their original scale with bilinear interpolation. The feature maps are fused additionally next. Feature maps get more context information, and the receptive field is also enlarged to the whole image. The detector is more discriminative with the help of more context information. We can find that the confidence of false positives drops obviously with this module, proving its effectiveness.

Finally, to make the framework strong to detect tiny objects, we comprehensively analyzed why the detected performance drops drastically with tiny objects and proposed a scale-invariant anchor strategy to tile anchors reasonably, especially for small objects based on the region proposal network (RPN) in [7]. On the other hand, we insert an efficient zoom-out and zoom-in architecture in Tiny-Net to enlarge the feature maps, which improve the recall of tiny objects obviously. Position-sensitive region of interests (RoI) pooling [8] is also used to share the computation from all detectors on the entire image and get more spatial information, which is faster and more accurate than the original RoI pooling in [7].

Our contributions can be summarized into four components.

1) We proposed a unified and self-reinforced framework called $\mathcal{R}^2$-CNN, which is *efficient* in computation and memory consumption, *robust* to false positives, and *strong* to detect tiny objects.
2) We proposed Tiny-Net, a lightweight residual network that can be trained from scratch and further improve the efficiency.
3) We insert a global attention block into $\mathcal{R}^2$-CNN to further inhibit the false positives.
4) We comprehensively analyze why the detected performance drops drastically with tiny objects and then make the framework strong to detect tiny objects.

The remainder of this paper is organized as follows. In Section II, we briefly introduce the state-of-the-art object detection methods and their applications on remote sensing systems. Then, we explain the details of our $\mathcal{R}^2$-CNN in Section III and show the experiments in Section IV. Finally, Section V concludes this paper with a discussion of the results.

II. Related Work

As a fundamental problem in a remote sensing area, object detection in remote sensing images has been extensively studied in recent years. Previous methods (such as scale-invariant feature transform [9] and histogram of oriented gradient (HoG) [10], [11]) use low-level or middle-level feature representations to detect objects. Recently, impressive improvements have achieved with convolutional neural networks. Cheng and Han [12] provide a review of the recent progress in those fields and propose two promising research directions, which are deep learning-based methods and weakly supervised learning-based methods.

Convolutional neural networks got a start from LeNet [13] and became popular with AlexNet [14]. Many impressive methods are proposed to promote the development of image recognition from then on, such as network-in-network [15], VGGNet [16], and GoogLeNet [17]. ResNet [6] is a milestone, which is using residual connections to train very deep convolutional models. It made a great improvement in image recognition. Object detectors, such as OverFeat [18] and region convolutional neural network (R-CNN) [19], made dramatic improvements in accuracy with those deep learning-based feature representations. OverFeat adopted a Conv-Net as a sliding window detector on an image pyramid. R-CNN adopted a region proposal-based method based on selective search [20] and then used a Conv-Net to classify the scale-normalized proposals. spatial pyramid pooling (SPP) [21] adopted R-CNN on feature maps extracted on a single image scale, which demonstrated that such region-based detectors could be applied much more efficiently. Fast R-CNN [22] and Faster R-CNN [7] made a unified object detector in a multitask manner. Region proposal networks are proposed to replace selective search. Dai *et al.* [8] proposed R-FCN, which uses position-sensitive RoI pooling to get a faster and better detector. While those region-based methods are too slow for practical use, a single-stage detector, such as YOLO [23] and SSD [24], is proposed to accelerate the processing speed but with a performance drop, especially in small objects.

Along with the rapid development with those mechanisms, small object detection seems much more difficult, and thus, researchers proposed many frameworks for small object detection specifically. Those methods mainly focus on how to implement a multiscale framework elegantly or using hard

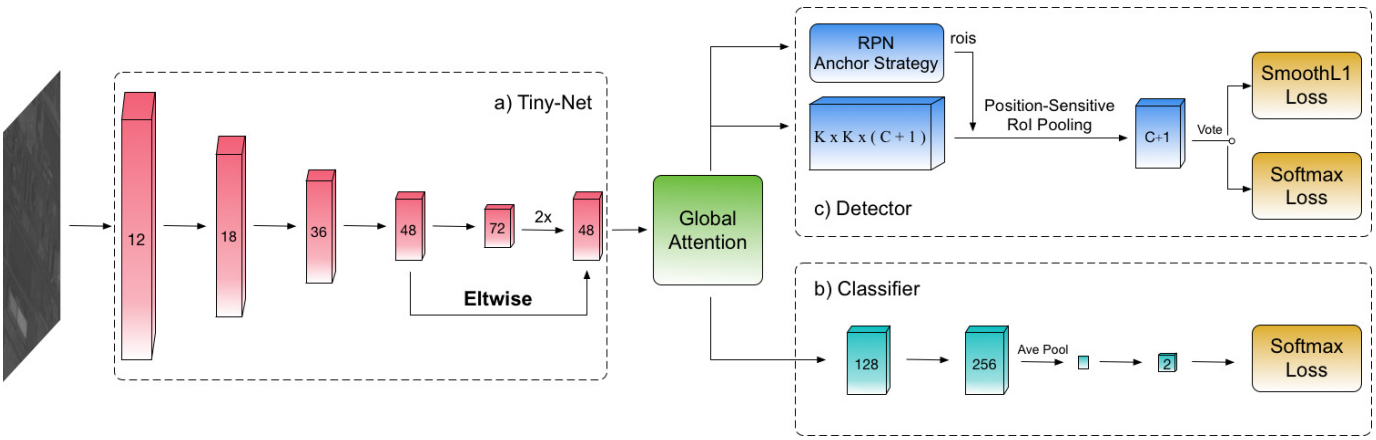

Fig. 2. Architecture of our $\mathcal{R}^2$-CNN. (a) Tiny-Net, a lightweight residual structure which enables fast and powerful features extraction from inputs. (b) Classifier, which can speed up the unified network and avoid false alarm raised by massive backgrounds. (c) Detector, which can locate target objects accurately if available. The classifier and detector are mutually reinforced each other under the end-to-end training framework. In addition, global attention block is built on top of Tiny-Net to inhibit false positives.

mining method which let the network pay more attention to small objects. Lin *et al.* [25] proposed feature pyramid networks that use the top-down architecture with lateral connections as an elegant multiscale feature warping method. Zhang *et al.* [26] proposed a scale-equitable face detection framework to handle different scales of faces well. Hu and Ramanan [27] showed that the context is crucial and defines the templates that make use of massively large receptive fields. Zhao *et al.* [28] proposed a pyramid scene parsing network that employs the context reasonable. Shrivastava *et al.* [29] proposed an online hard example mining method that can improve the performance of small objects obviously.

Many methods [1]–[5], [30]–[35] are proposed to improve the object detection accuracy in remote sensing images with convolutional neural networks. Those methods often use pretrained CNN models on large data sets to handle the limited remote sensing training data. Zhang *et al.* [31] used the trained CNN models to extract surrounding features. Those features were combined with features from HoG to get final representations and then applied gradient orientation to generate region proposals. Zhu *et al.* [36] used CNN features from multilevel layers for object detection, which handle the scale-invariance with single scale input. Jiang *et al.* [37] used a graph-based superpixel segmentation to generate proposals and then trained a CNN to classify these proposals into different classes. Cheng *et al.* [35] introduced a rotation-invariant operator to the existing CNN architectures and achieves a significant performance. Long *et al.* [2] proposed an unsupervised score-based bounding-box regression for accurate object localization in remote sensing images. Those methods mainly focus on small region segment compared to the original large remote sensing image input, usually over $10\,000 \times 10\,000$ pixels, and thus, they cannot scale up to handle such large input gracefully. Zhang *et al.* [4] attempted to detect airports in large-scale images first to reduce overall airplane detection time, but the training and testing images are all from the region near airports without arbitrary massive backgrounds. For practical use, object detection in large-scale remote sensing images is very important and necessary.

III. Proposed Method

The $\mathcal{R}^2$-CNN, shown in Fig 2, consists of the backbone Tiny-Net, intermediate global attention block, and final classifier and detector.

A. $\mathcal{R}^2$-CNN

$\mathcal{R}^2$-CNN is a unified and self-reinforced framework working in an end-to-end manner. Considering that the large input image size increases the computation time and memory consumption quadratically, large-scale remote sensing images (such as $20\,000 \times 20\,000$ pixels) are cropped with a much more smaller scale (such as 640×640 pixels) with 20% overlap. By processing the patches asynchronously, the limited memory is not a problem anymore.

A convolutional backbone structure is then applied to the inputs, which enables powerful features extraction. Based on those discriminative features, the classifier first predicts the existence of detection target in the current patch, and the detector is followed to locate them accurately if available. The classifier and detector are mutually reinforced each other under the end-to-end training framework. There are two advantages of this self-reinforced architecture as follows.

First, the light classifier branch can filter out a blank patch without heavier detector cost. Classifier's architecture is in Fig. 2(b), and we just use two CONV-BN-RELU blocks to extract features from the former features. Global average pooling and a 1×1 convolutional operator are then attached to it. Softmax loss is employed to guide the training of the classifier. Considering that most crops do not contain a valid target in remote sensing images, about 99% of the total patches do not need to pass the heavy detector branch.

Second, massive and complex backgrounds appear in a real scenario may introduce more false positive, for instance, desert region with random texture or urban area with massive building structure. The false positives are first inhibited by the mutual reinforcement from the classifier and the detector. On the one hand, the classifier can distinguish the difficult situation even when there is only one tiny object (such as 12×12 pixels) in the patch. We explain this promotion mainly

given the fine-grained feature extracted from the detector. On the other hand, the detector receives less false positive candidates since most of them are filtered out by the classifier. Even if the patches are distinguished incorrectly by the classifier, the detector can still rectify the results later.

There are three outputs from our network. One output m from classifier represents the probability of whether there are target objects in corresponding patch or not. Two outputs from detector represent the discrete probability $[p = (p_0, \ldots, p_k)]$ distribution of each RoI over $K + 1$ categories and bounding-box regression offsets, $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, for each of the K object classes, indexed by k, in the corresponding patch. We use the parameterization for t^k in [19], in which t^k represents a scale-invariant translation and log-space height/width shift relative to an object proposal. Each of the training patches is labeled with a binary ground truth n, and each RoI in detector is labeled with a ground-truth class u and a ground-truth bounding-box regression target v. We use a unified multitask loss L on each patch to joint classifier and detector

$$L(m, n, p, u, t^u, v) = L_{\text{cls}}(m, n) + \mu[n = 1](L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)) \quad (1)$$

in which $L_{\text{cls}}(p, u)$ and $L_{\text{cls}}(m, n)$ are softmax loss and L_{loc} is smooth-L1 Loss in [7]. The hyperparameter λ and μ in (1) controls the balance between the three task losses. All experiments use $\lambda = 1$ and $\mu = 1$. We only backpropagate detector's loss when there are detection targets in the corresponding patch during training time. The entire network is *efficient, robust*, and *strong*.

B. Tiny-Net

Recently, CNN-based methods often use VGG [16] or ResNets [6] as feature extractors. Those models are pretrained on ImageNet [38], a large-scale hierarchical image database with millions of images, to deal with the limited training samples and get a much more quicker convergence. However, there are still many disadvantages when using those pretrained models. First, those models are too heavy to reach real-time efficiency. Second, those models are designed specifically for image classification, making that the feature resolution may be not enough for object detection. Finally, considering the heavy parameters, training scratch is pretty difficult, especially with limited training samplers. When applying the pretrained models to remote sensing frameworks, the domain gap between natural images and remote sensing images may make the models suboptimal.

The architecture of Tiny-Net is shown in Table I. The 3×3 block is a residual block in ResNet [6] except conv-1. We do not apply the downsample operator in conv-1 to enable the feature maps more discriminative for tiny object detection, which is different from ImageNet pretrain models such as VGG [16] and ResNets [6]. The parameters of Tiny-Net are far less than ResNets. Thanks to this lightweight architecture, Tiny-Net can be trained from scratch and converge well just with a cycle training schedule, which iteratively updates the step learning rate twice or more. Under this condition, Tiny-Net will not be influenced by the domain gap between the natural images and the remote sensing images.

TABLE I
ARCHITECTURE OF OUR TINY-NET. EACH 3×3 BLOCK ARE RESIDUAL BLOCK IN RESNET [6] EXCEPT CONV-1

layer name	output size	architecture
conv-1	640×640	$[3 \times 3, 12] \times 2$
conv-2	320×320	$\begin{bmatrix} 3 \times 3, 18 \\ 3 \times 3, 18 \end{bmatrix} \times 2$
conv-3	160×160	$\begin{bmatrix} 3 \times 3, 36 \\ 3 \times 3, 36 \end{bmatrix} \times 2$
conv-4	80×80	$\begin{bmatrix} 3 \times 3, 48 \\ 3 \times 3, 48 \end{bmatrix} \times 3$
conv-5	40×40	$\begin{bmatrix} 3 \times 3, 72 \\ 3 \times 3, 72 \end{bmatrix} \times 2$

Benefited from those characters, Tiny-Net can reduce inference cost and preserve powerful features for tiny object detection in remote sensing images, which further improved the efficiency of $\mathcal{R}^2$-CNN.

C. Global Attention

Thanks to the unified classifier and detector, numerous blank patches are filtered by classifier, making that false positives reduce obviously. However, the problem still exists because of the limited receptive field. When you catch sight of two objects with similar appearance, you may not be sure what they are without context information. For example, when you see the top image in Fig. 3(a), you may confuse in this question: what exactly are they? However, when you see two images in the bottom image, you can easily distinguish them out. As discussed in [39], the CNN has two types of receptive fields: the theoretical receptive field and the effective receptive field. The theoretical receptive field indicates the input region that can affect the value of this unit theoretically. However, not every pixel in the theoretical receptive field contributes equally to the final output. Only a subset of the area has an effective influence on the output value, which is called effective receptive field. The effective receptive field is smaller than the theoretical receptive field, as shown in Fig. 3(b). The limited effective receptive field leads the final feature map to obtain little context information, thus leading to more false positives.

Inspired by this phenomenon, we use feature pyramid pooling as a global attention block on the top of Tiny-Net. The architecture is shown in Fig. 3(c). The feature maps are first pooled in different pyramid levels, such as 1×1, 2×2, and 4×4. Then, we recover the pooled features to their original scale with bilinear interpolation. The feature maps are fused additionally next. Feature maps can get more context information, and the receptive field will also be enlarged to the whole image. The global attention module fuses the features from different pyramid scales and leads the detector to pay more attention to the whole image. The detector is

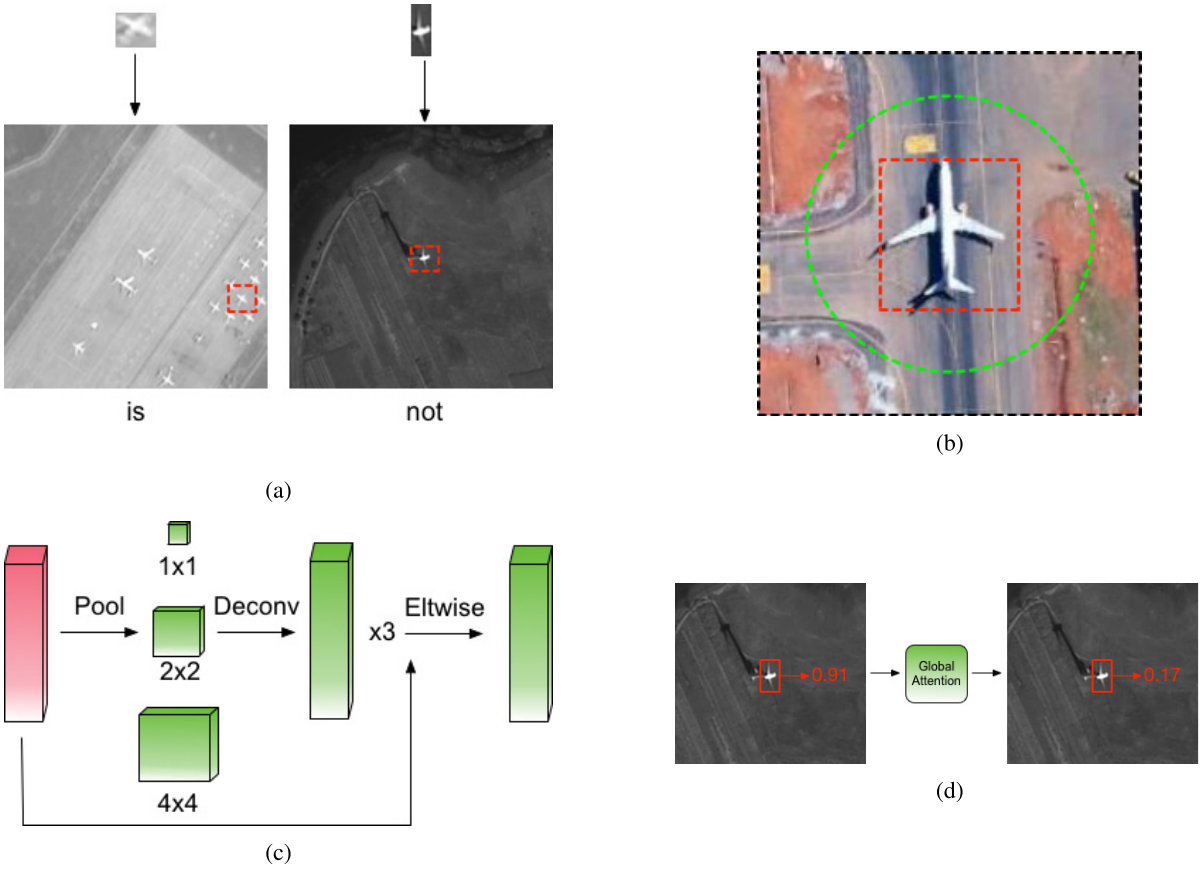

Fig. 3. (a) Whether they are airplanes or not? (b) Black area: theoretical receptive field. Green area: effective receptive field. Red area: bounding box. (c) Global attention block. (d) False positive's confidence drop obviously with global attention.

more discriminative with the help of more context information. We can found that the confidence of false positives drops obviously with this module, as shown in Fig. 3(d), proving its effectiveness.

D. Detector

Our $\mathcal{R}^2$-CNN is strong to detect tiny objects. The architecture of the detection branch is shown in Fig. 2(c). The state-of-the-art object detectors are mainly based on the RPN in [7], which use anchors to generate object proposals. Anchors are a set of predefined boxes with multiple scales and aspect ratios tiled regularly on the image plane. However, anchor-based detectors drop the performance drastically on objects with tiny sizes, such as less than 16×16 pixels, and those tiny objects are the majority in remote sensing images, such as airplanes, ships, and cars. To tackle this problem, we first investigate why this is the case and propose a scale-invariant anchor strategy to tile anchors reasonably, especially for small objects. On the other hand, we insert an efficient zoom-out and zoom-in architecture in Tiny-Net to enlarge the feature map without margin cost, which improves the recall of tiny objects obviously. Position-sensitive RoI pooling is also used to get more spatial information. Through these ways, we get the excellent results on tiny object detection.

1) Why This Is the Case?: Like in Fig. 4(a), the stride size of the lowest anchor-associated layer is too large (e.g., 16 pixels or 32 pixels), and the features loss along with the downsampling of pooling layer. Therefore, tiny objects have been highly squeezed on these layers and have a few features for detection. An airplane may be only 1×1 pixels in the final feature map. On the other hand, the anchor scales are discrete (i.e., $16, 32, 64, \ldots 2^k$), but object scales are continuous. During training, an anchor will be assigned to a ground-truth box if its intersection over union (IoU) with this box is the highest than other anchors or its IoU is higher than a threshold T_h. When the object's scale is near to anchor scales, they will be attached more anchors and thus easier to be located. The face detector single shot scale-invariant face detector (S^3FD) [26], which uses SSD [24] architecture, explained this phenomenon appropriately, and we infer their statistics in Fig. 4(b). It shows the number of matched anchors at different face scales under $16, 32, 64, \ldots 2^k$ anchor scales. If an object has a scale over average line, it will be matched enough anchors. However, tiny objects are matched a few anchors, leading to performance drop drastically.

2) Our Method: To tile anchors more reasonably, we analyze the bounding boxes' scale distribution of our training data set, which is shown in Fig. 4(d). We can see that tiny objects are majority in it. Instead of choosing anchors by hand, we run k-means clustering on the training set to automatically find good priors. Our training set's bounding-box scales are $X : (x1, x2, \ldots, x_n)$. There are k anchor scales we want, and the center scales are $(\mu_1, \mu_2, \ldots, \mu_k)$. Scales are clustered

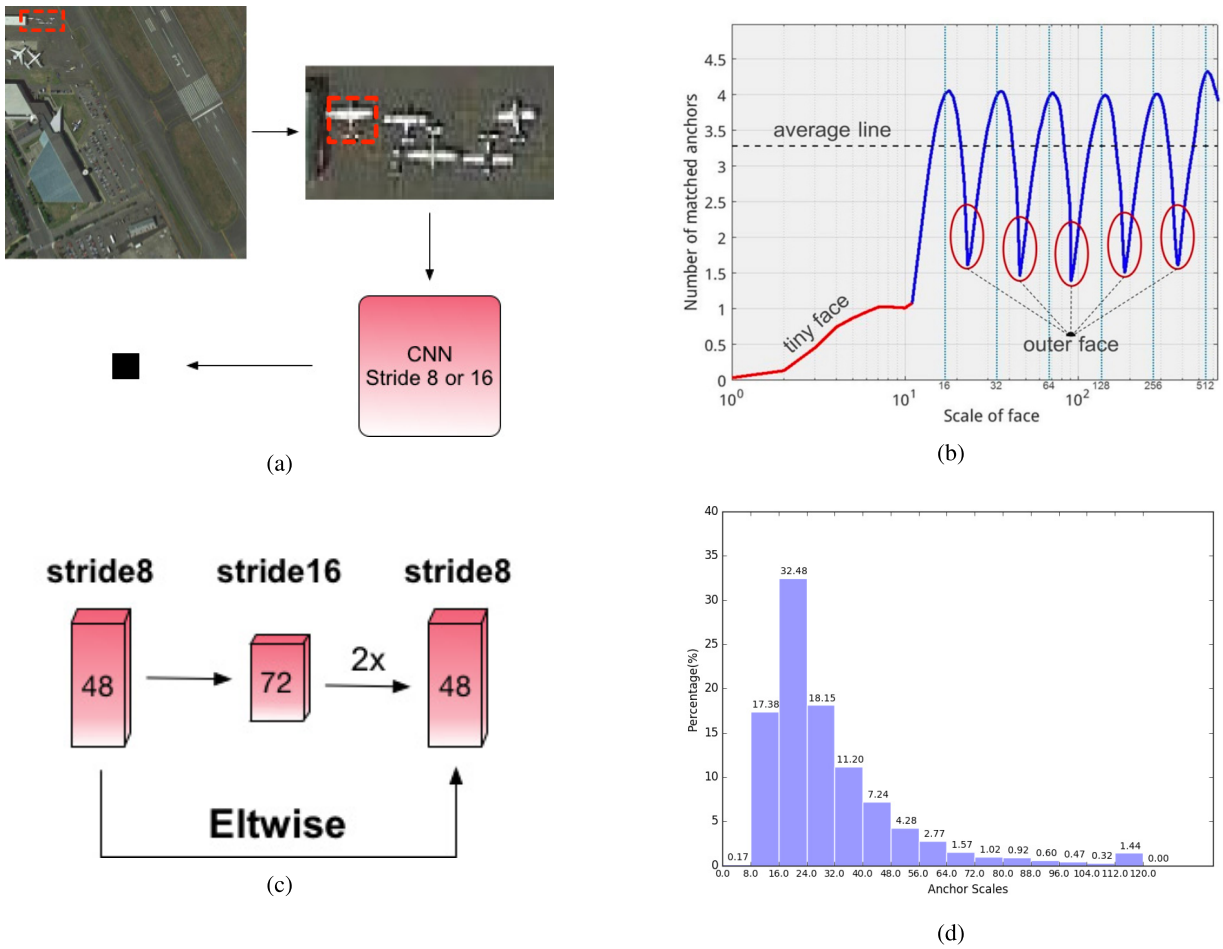

Fig. 4. (a) Few features: small objects have a few features at detection layer. (b) Anchor matching analysis: the figure is from S^3FD [26], and tiny and outer objects match too little anchors. (c) Our skip-connection architecture for reducing the anchor stride by enlarging the feature map. (d) Data distribution: bounding-box scales of our training data set.

in $(s1, s2, \ldots, s_k)$. Through minimizing

$$\arg\min \sum_{i=1}^{k} \sum_{x \in s_i} ||x - \mu_i||^2 \tag{2}$$

we can get the clustered anchor scales.

To extract beneficial features for tiny object detection, one way is to reduce the anchor stride by enlarging the feature map using a zoom-out and zoom-in architecture. Like shown in Fig. 4(c), we first zoom out the feature map with a residual block, and thus, its anchor stride is 16 pixels. Then, we employed to recover the feature maps that are recovered to their original scale using a zoom-in operator. In addition, we use a skip connection between the stride-8 layer and the upsampled layer. We found that the stride-16 layer can extract more high-level features, which is beneficial to object detection. The skip connection can fuse low-level features and high-level features, making final feature maps more discriminative.

Considering the complicated backgrounds of remote sensing images, we use position-sensitive RoI pooling [8] in our detector instead of RoI pooling. RoI pooling applies costly per-region subnetwork hundreds of times. If there are 1000 proposals, the detector will be tested 1000 times wastefully. In contrast to this operator, position-sensitive RoI pooling is fully convolutional with almost all computation shared on the entire image. Much computation is saved with this operator. On the other hand, position-sensitive RoI pooling can address the dilemma between translation invariance in image classification and translation variance in object detection. More spatial information is extracted, thus leading to a better performance.

IV. Experiments

In this section, we first present the implementations of our $\mathcal{R}^2$-CNN, such as data sets, evaluation metric, and parameter settings. The results of our network and comparative experiments are then discussed.

A. Implementations

1) Data Sets: Due to the lack of standard data sets of large-scale remote sensing images, we collected 1169 GF-1 images and 318 GF-2 images that are 18 000 × 18 192 pixels, 2.0-m resolution, and 27 620 × 29 200 pixels, 0.8-m resolution, respectively. In addition, we collected 38 472 pieces of 640 × 640 images, which contains target objects from the publicly available Google Earth service to supplement the poor positive patches in GF images. All the images are cropped in

TABLE II

$\mathcal{R}^2$-CNN'S RESULTS IN *GF1-Test-Dev*. RESULTS ARE SHOWN IN UNIFIED/ UNUNIFIED/FULLY DETECTION TRAINING AND TESTING

Score Thre	TP	FP	Recall	Precision
0.05	613 / 593 / 616	186 / 264 / 8126	99.35 / 96.11 / 99.84	76.72 / 69.19 / 6.97
0.5	607 / 591 / 612	46 / 93 / 561	98.38 / 95.79 / 99.19	92.96 / 86.40 / 52.17
0.8	593 / 577 / 596	15 / 41 / 264	96.11 / 93.52 / 96.60	97.53 / 93.36 / 69.30
0.85	591 / 573 / 592	10 / 29 / 189	95.78 / 92.87 / 95.95	98.33 / 95.18 / 75.80
0.9	579 / 568 / 586	7 / 18 / 131	93.84 / 92.06 / 94.98	98.80 / 96.93 / 81.73
0.95	556 / 542 / 561	3 / 7 / 37	90.11 / 87.84 / 90.92	99.46 / 98730 / 93.81

the patches of 640×640 pixels with 20% overlap. In particular, if we have a maximum object scale d and a cropped scale D, we recommend an overlap d/D. For example, as the scales in Fig. 4(d), the maximum scale of the objects in our data set is about 128 and the cropped scale is 640; thus, a 20% overlap is applied to not only prevent the object from being truncated off but also augment the data sets. To help the convergence of the network, we control the proportion of positive patches and negative patches to be 1 : 3 to obtain a balanced training set. The negative patches are selected using hard example boosting to enhance the training process.

We collect 102 GF-1 images as *GF1-test-dev* and 40 GF-2 images as *GF2-test-dev* to evaluate the ability of $\mathcal{R}^2$-CNN. To evaluate our model more exhaustively, we collected 4633 images (640×640 pixels) from Google Earth as *Rgb-test-dev* and 1000 images (640×640 pixels) from GF-1 and GF-2 as *Gray-test-dev*, which help us evaluate the ability of the detector.

2) Evaluation Metric: Considering the few target objects in large-scale remote sensing images and the requirement for practical engineering applications, we evaluate our $\mathcal{R}^2$-CNN with *recall* and *precision* of different score thresholds. The correct number of detections is true positives TP, and the number of spurious detections of the same object is false positives FP. The number of ground-truth instances is NP. The precision and recall are given in the following:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{NP}. \tag{4}$$

We also show the instance number in detail for more intuitional and convincing. Considering numerous negative scene of classifier, the *accuracy* is generally over 99.0%, making this metric meaningless. Therefore, we use *recall*, *precision*, and instance number to evaluate our classifier. We use mean average precision (*mAP*) and *Max-Recall* with a score threshold of 0.05 as in PASCAL-VOC [40] to evaluate our detector, which are defined as

$$mAP = \frac{1}{Q} \sum_{q=1}^{Q} AP(q) \tag{5}$$

$$AP = \frac{\sum_{k=1}^{n} (P(k) \times r(k))}{\mid R(q) \mid} \tag{6}$$

where Q is the number of categories, $\mid R(q) \mid$ is the number of images relevant to the category q, k is the rank in the sequence of retrieved images, n is the total number of retrieved images, $P(k)$ is the precision at cutoff k in the list, and $r(k)$ is an indicator function whose value is 1 if the image at rank k is relevant and is 0 otherwise.

3) Parameter Settings: We adopt synchronized stochastic gradient descent training on 8 GPUs with synchronized batch normalization. A minibatch involves 1 image per GPU and 512 proposals per image for detector training. We use a momentum of 0.9 and a weight decay of 0.0005. We use a learning rate of 0.001 for 80k minibatches and 0.0001, 0.00001 for the next 80k and 40k minibatches. The learning rate and training epochs are iterated twice as a cycle schedule, because the network is trained from scratch. We randomly initialize all layers by drawing weights from a zero-mean Gaussian distribution with a standard deviation of 0.01. The anchor's scale in RPN we used in our final model is $[1, 2, 4, 10, 16, 30]$ with a stride of 8, and the anchor's ratio is 1 considering the airplane's shape. Other implementation details are the same as in [6] and [7]. We also use multiscale training with scale $[513, 609, 721, 801, 913]$ and online hard example mining [29] for hard example boosting. We stretch remote sensing images from uint16 to uint8 which is divided by a factor of 4. We also stretch the images' histogram with a factor in $[0, 0.02]$, which can not only inhibit the noise in remote sensing images but also as a data augmentation method. We augment the data online with rotation and flipping randomly. Our implementation uses Caffe [41].

B. Evaluation of $\mathcal{R}^2$-CNN

We comprehensively evaluate our method on the *GF1-test-dev* and *GF2-test-dev*. The results are shown in Tables II and III. It is *efficient* that we can process a $18\,000 \times 18\,192$ GF-1 image in 29.4 s on Titian X just with single thread. It is *robust* that with a score threshold of 0.85, there are only 10 false positives in *GF1-test-dev* and 5 false positives in *GF2-test-dev*. It is *strong* that we can get 95.78 *recall* and 98.33 *precision* on *GF1-test-dev* also with a score threshold of 0.85, showing its potential for practical application. We compared our $\mathcal{R}^2$-CNN with ununified (train and test separately) network and fully detection network. The results show that when we joint detector with classifier, they can promote each other to get the best results. When the network is ununified, the classifier cannot obtain

TABLE III

$\mathcal{R}^2$-CNN'S RESULTS IN *GF2-Test-Dev*. RESULTS ARE SHOWN IN UNIFIED/UNUNIFIED/FULLY DETECTION TRAINING AND TESTING

Score Thre	TP	FP	Recall	Precision
0.05	108 / 79 / 113	74 / 173 / 5746	93.91 / 68.69 / 98.26	59.34 / 31.35 / 1.93
0.5	107 / 77 / 111	14 / 34 / 217	93.04 / 66.95 / 96.52	88.43 / 69.36 / 33.84
0.8	105 / 74 / 108	8 / 21 / 69	91.30 / 64.34 / 93.91	92.92 / 77.89 / 61.01
0.85	104 / 74 / 106	5 / 17 / 47	90.43 / 64.34 / 92.17	95.41 / 81.32 / 69.28
0.9	100 / 72 / 105	2 / 13 / 31	86.96 / 62.60 / 91.30	98.04 / 84.71 / 77.20
0.95	96 / 69 / 100	1 / 7 / 13	83.48 / 60.00 / 86.96	98.97 / 90.79 / 88.49

TABLE IV

RESULTS OF CLASSIFIER ON *GF1-Test-Dev*. $\vdash$ MEANS FINE-TUNING FROM IMAGENET PRETRAINED MODEL. $\dashv$ MEANS TRAINING FROM STRETCH. $-$ MEANS TRAINING WITHOUT DETECTOR

Model	TP	FP	Recall	Precision	Time Cost
ResNet-50$^{\vdash-}$ [6]	134	34	87.01	79.76	48.21 ms
ResNet-50$^{\dashv-}$ [6]	128	29	83.12	81.53	48.21 ms
Ours^{-}	105	179	68.18	36.97	16.63 ms
$\mathcal{R}^2$-CNN	148	45	96.10	76.68	16.63 ms

TABLE V

RESULTS OF CLASSIFIER ON *GF2-Test-Dev*. $\vdash$ MEANS FINE-TUNING FROM IMAGENET PRETRAINED MODEL. $\dashv$ MEANS TRAINING FROM STRETCH. $-$ MEANS TRAINING WITHOUT DETECTOR

Model	TP	FP	Recall	Precision	Time Cost
ResNet-50$^{\vdash-}$ [6]	61	2	68.54	96.83	48.21 ms
ResNet-50$^{\dashv-}$ [6]	82	204	92.13	28.67	48.21 ms
Ours^{-}	55	67	61.80	43.65	16.63 ms
$\mathcal{R}^2$-CNN	82	38	92.13	68.33	16.63 ms

enough discrimination without the feature extracted by the detector, leading *recall* and *precision* drop obviously. When detecting the objects all using detector, too many false positives appeared and the efficiency is lower considering the heavy detector. Our network achieves the superior results on performance and speed, showing its *efficient, robust*, and *strong*, corresponding visualized results are shown in Fig. 6.

1) Efficient: Tiny-Net enables the inference time of network less than ResNet-50 or other large models and preserves powerful features for object detection, and the details are shown in Tables IV and V. The classifier with Tiny-Net costs 16.63 ms with 640 × 640 inputs on Titian X, and the detector costs 45.21 ms with the same setting. The detector is three times slower than classifier. In our *GF1-test-dev*, there are only 154 patches that have target objects. The unified architecture makes that 99.9% of the total patches do not need to pass the heavy detector branch. Total costs of our network and fully detection network are shown in Table VI. We can process a 18 000 × 18 192 GF-1 image in 29.4 s on Titian X just with single thread. Though there is still a long way to build a real-time detection system on large-scale remote sensing images, our network tackles the problem well with the proposed methods.

TABLE VI

TIME COSTS OF DIFFERENT METHODS WITH SINGLE THREAD

Benchmark	$\mathcal{R}^2$-CNN	Detection
GF1-test-dev	29.4 s	64.7 s
GF2-test-dev	66.2 s	163.7 s

TABLE VII

PERFORMANCE WITH OR WITHOUT GLOBAL ATTENTION BLOCK

Score Thre	TP	FP	Recall	Precision
0.05	613 / 611	186 / 216	99.35 / 99.03	76.72 / 73.88
0.5	607 / 609	46 / 101	98.38 / 98.70	92.96 / 85.77
0.8	593 / 600	15 / 63	96.11 / 97.24	97.53 / 90.49
0.85	591 / 597	10 / 49	95.78 / 96.76	98.33 / 92.41
0.9	579 / 582	7 / 27	93.84 / 94.33	98.80 /95.56
0.95	556 / 559	3 / 11	90.11 / 90.60	99.46 / 98.07

2) Robust: The unified classifier can identify the difficult situation even when there is only one tiny objects in the patch, given the fine-grained feature extracted by the detector, and the detector receives less false positive candidates since most of them are filtered out by the classifier. The results are shown in Tables IV and V. Because of the large memory needed by ResNet-50 with an anchor stride of 8, we cannot evaluate our $\mathcal{R}^2$-CNN with ResNet-50. Thus, the setting of ResNet-50 is the same as in [6] without detector. *GF1-test-dev* is cropped in 131 148 patches with 640 × 640 pixels and 20% overlap, and there are also 123 120 patches from *GF2-test-dev*. The *recall* and *precision* only consider the positive patches. The results of our $\mathcal{R}^2$-CNN get the best results compared with others. The classifier drops performance drastically without detector, and the recall of our $\mathcal{R}^2$-CNN is higher than ResNet-50, showing the effectiveness of the features extracted by the detector. Though there are more false positives in our model, the detector can rectify the results later. We have attempted to train ResNet-50 from scratch to break the domain gap between natural images and remote sensing images but get bad results. We argue that this is mainly because of the numerous parameters of ResNet-50 but with the small amount of training sets. Though there are hundreds of thousand remote sensing images for us, only a few of them can be used during training time to handle the positive–negative imbalance problem.

TABLE VIII
RESULTS OF DETECTORS ON *Rgb-test-dev*. − MEANS TRAINING WITHOUT CLASSIFIER. + MEANS TRAINING WITH IMAGENET PRETRAINED MODEL

Metric	stride-16	No Conv-5	ResNet-50$^+$ [8]	Faster R-CNN [7]	Anchor-4	Anchor-5	Anchor-8	Ours$^-$	$\mathcal{R}^2$-CNN
mAP (%)	84.73	93.74	83.86	94.61	94.95	95.12	96.43	95.81	96.04
Max Recall (%)	87.21	95.40	86.34	95.62	96.38	96.74	97.88	97.12	97.26
Time cost (ms)	31.93	41.81	58.26	49.26	39.58	42.53	50.07	45.21	45.21

TABLE IX
RESULTS OF DETECTORS ON *Gray-Test-Dev*. − MEANS TRAINING WITHOUT CLASSIFIER. + MEANS TRAINING WITH IMAGENET PRETRAINED MODEL

Metric	stride-16	No Conv-5	ResNet-50$^+$ [8]	Faster R-CNN [7]	Anchor-4	Anchor-5	Anchor-8	Ours$^-$	$\mathcal{R}^2$-CNN
mAP (%)	83.68	95.16	82.53	94.90	94.56	95.35	97.47	97.01	97.25
Max Recall (%)	85.19	96.15	84.07	96.04	95.58	96.49	98.23	98.13	98.03
Time cost (ms)	31.93	41.81	58.26	49.26	39.58	42.53	50.07	45.21	45.21

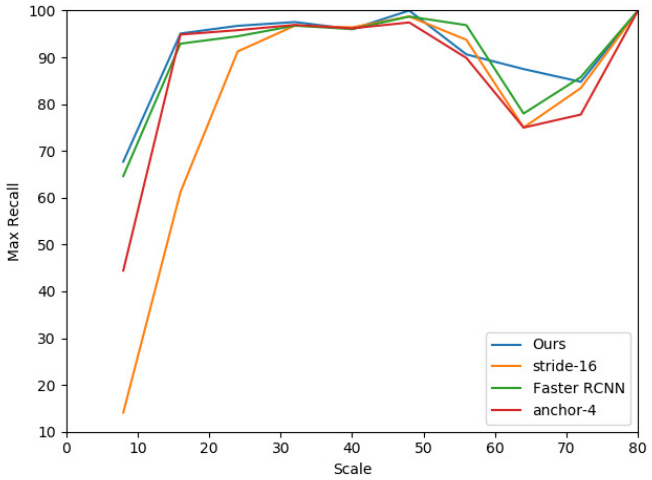

Fig. 5. *Max-Recall* of different scales in *Gray-test-dev*.

Besides, we evaluate the effectiveness of global attention block on *GF1-test-dev*. The results are shown in Table VII. When $\mathcal{R}^2$-CNN is implemented without global attention block, there are more false positives because of the limited receptive field. With the global attention block, the confidence of false positives drops obviously. We found that the recall drops a little with global attention block. Comparing to the enhancement of inhibiting false positives, this is a better model for practical engineering.

3) Strong: To validate the effectiveness of $\mathcal{R}^2$-CNN on tiny object detection, we evaluate our network on *Rgb-test-dev* and *Gray-test-dev*. The results are shown in Tables VIII and IX.

How Important Is the Zoom-Out and Zoom-In Architecture? Considering the large memory needed by ResNet-50 with an anchor stride of 8 pixels, a detector with ResNet-50 is implemented with an anchor stride of 16 pixels. The results of *stride-16* with $\mathcal{R}^2$-CNN only get 83.68 mAP in *Gray-test-dev*. Besides, the mAP of different scales is shown in Fig 5. The performance of objects larger than 32 pixels is basically comparable to our method, but the results of small objects drop obviously without the architecture. In addition, we attempt to attach the global attention block to *conv-4* directly. The results of *No Conv-5* drop 2 points compared to our method. This modification shows that *Conv-5* can extract more high-level features that are benefited for object detection, proving the effectiveness of this architecture.

How Important Is Our Anchor Strategy? The number of clustered points is a handcrafted parameter in k-means. We attempt to cluster anchor scales with a number of 4, 5, 6, and 8. The results with different anchor scales are shown in the following.

1) *Four Anchor Scales:* [2.87, 5.44, 11.39, 26.72].
2) *Five Anchor Scales:* [2.32, 4.51, 7.62, 13.57, 28.34].
3) *Six Anchor Scales:* [2.18, 3.90, 6.17, 9.8, 15.91, 29.83].
4) *Eight Anchor Scales:* [1.97, 3.14, 4.8, 7.05, 10.53, 15.52, 24.55].

Max-Recall of different scales is shown in Tables VIII and IX. The results show that the distribution of anchors can better fit the data set to reach better performance with this strategy. We found that *Anchor-6* gets an excellent tradeoff between efficiency and performance so that it is the final parameter of our $\mathcal{R}^2$-CNN.

How Important Is Position-Sensitive RoI Pooling? Compared with Faster R-CNN [7], our $\mathcal{R}^2$-CNN improves mAP by 2.35 points in *Gray-test-dev*, particularly in tiny objects. The spatial information is better encoded via position-sensitive RoI pooling, which is beneficial to tiny object detection. Moreover, the time costs is 49.26 ms for Faster R-CNN but 45.21 ms for $\mathcal{R}^2$-CNN with position-sensitive RoI pooling. Through sharing all proposals' weights, we instead recalculate the feature maps of all proposals of voting from the final feature maps. This modification is also greatly helpful for an efficient process, especially when there are numerous patches.

Comparison Experiments With the State of the Art: To validate the effectiveness of our architecture, the comparison experiments are implemented with FPN Faster R-CNN and Mask R-CNN. Considering the lack of mask annotations, Mask R-CNN is only implemented with RoI Align. Both the methods are implemented with a ResNet-50 backbone. The results are shown in Table X. The comparable results prove the effectiveness of $\mathcal{R}^2$-CNN. We also found that the RoI Align brings little improvement to the FPN Faster R-CNN baseline. Considering that the tiny objects are dominating our

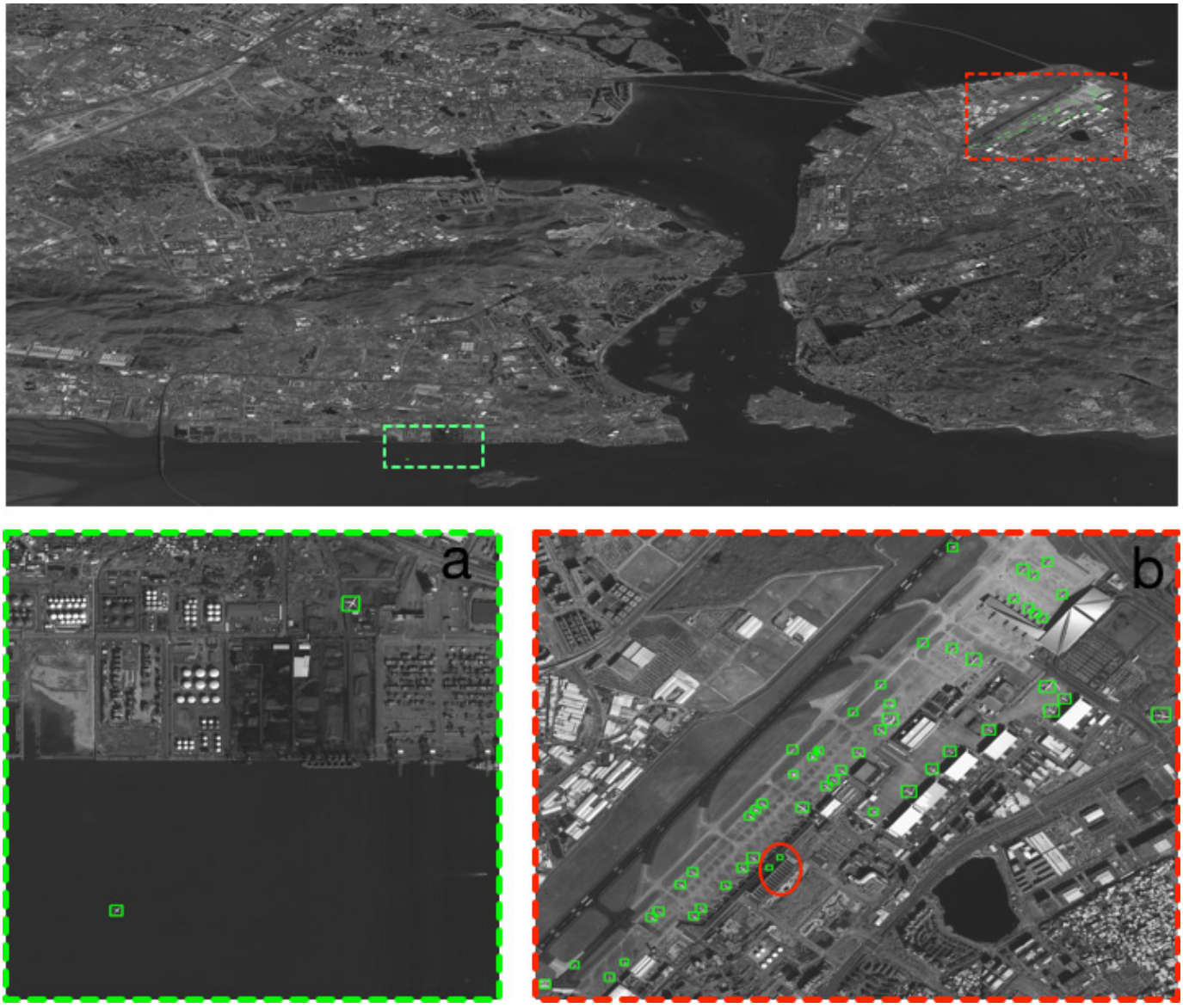

Fig. 6. Results of our $\mathcal{R}^2$-CNN with a score threshold of 0.9. Two airplanes are flying back to airport in image a. Results of the airport are shown in image b. Two objects marked by the red circle in image b are false positives.

TABLE X
COMPARISON EXPERIMENTS WITH FPN FASTER R-CNN AND MASK R-CNN ON *RGB-Test-Dev*. – MEANS TRAINING WITHOUT CLASSIFIER

Metric	mAP (%)	Max Recall (%)
FPN Faster RCNN	96.07	96.55
Mask R-CNN	96.20	96.63
FPN Faster R-CNN with Tiny-Net	95.37	96.28
Ours$^-$	95.81	96.04
$\mathcal{R}^2$**-CNN**	96.04	97.26

data set, this result is reasonable because the potential of RoI Align is not fully explored due to the small feature maps. The result of FPN Faster R-CNN with Tiny-Net is 0.5 points lower than $\mathcal{R}^2$-CNN training without classifier reinforced, proving the effectiveness of the specific design for Tiny-Net. All those results show that our $\mathcal{R}^2$-CNN reaches a great tradeoff within efficient processing, false positives inhibiting, and tiny object detection.

C. Discussion

In our experiments, we get pretty surprising results in Fig 7, i.e., an image is under the heavy haze. We found that we can detect the aircrafts easily. The confidence of those objects is higher than 0.9 but lower than 0.95. It is not a high score in our results (a high confidence is larger than 0.99) but still enough for practical engineering applications. We must recognize that these gratifying results are not only coming from the reasonable architecture of our $\mathcal{R}^2$-CNN but also the precise annotation of similar situations in our training set. The backbone, classifier, and detector are specially designed to converge well while training from scratch. This is a meaningful

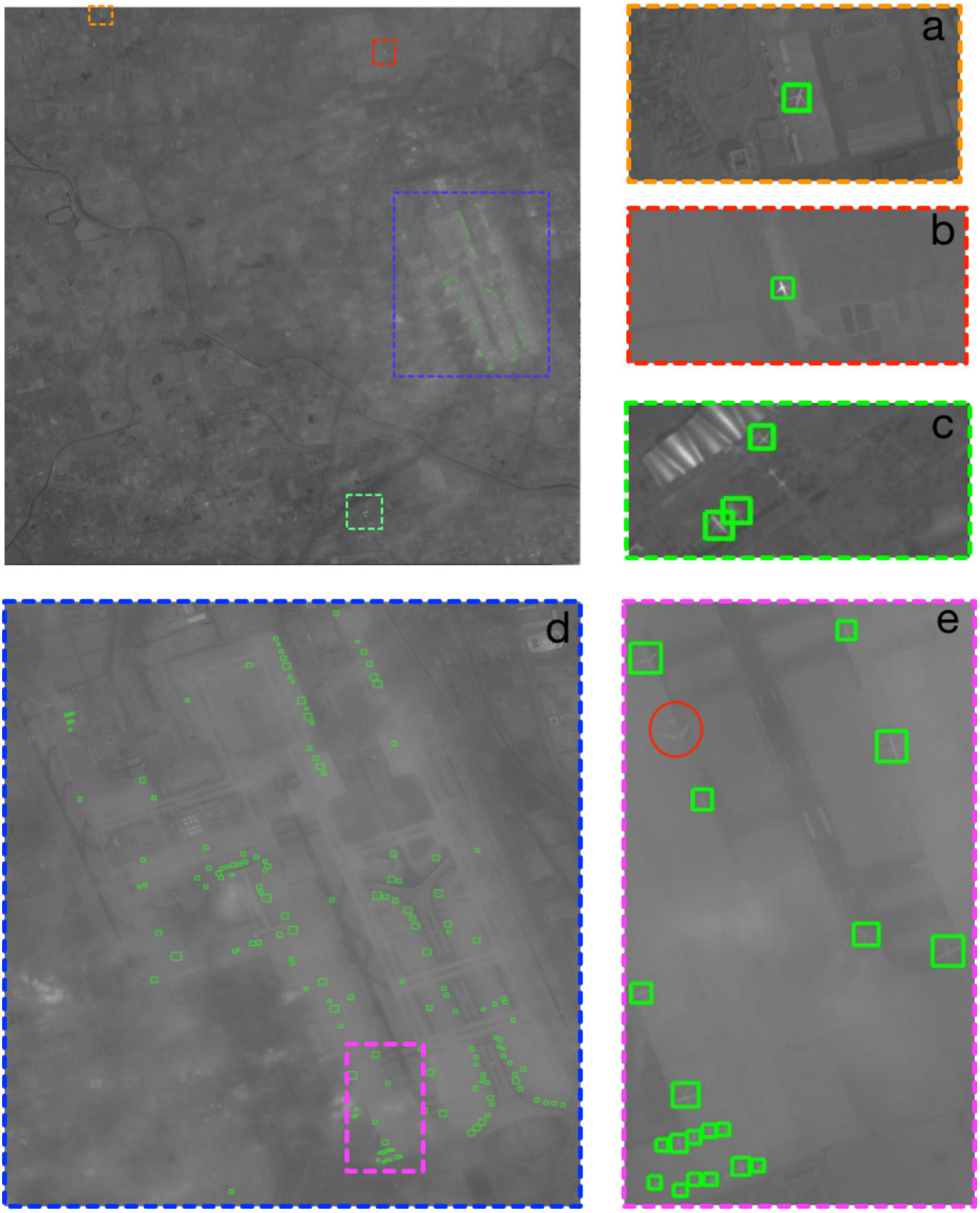

Fig. 7. Results of our $\mathcal{R}^2$-CNN with a score threshold of 0.9. The area is under the heavy haze. Image a shows an airplane in the wild. Image b shows an airplane flying in sky. Image c shows an airport with some airplanes. Image d shows the results of airport. Image e shows the details of d. The ignored airplane marked by a red circle in image e has a confidence of 0.83.

result for us to understand the powerful generalization ability of deep convolutional neural networks. However, annotating all those terrible conditions very well is not a sensible selection. However, we can still explore why this is the case and how does CNN execute such well to push the meaningful research in those situations. There are numerous remote sensing resources to utilize and problems to solve. With more and more powerful operators and theories, hopefully, we can fast promote the development of real-time remote sensing systems in the future.

V. Conclusion

We proposed $\mathcal{R}^2$-CNN, a unified and self-reinforced convolutional neural network under the end-to-end training framework, which joint the classifier and detector elegantly. The lightweight backbone Tiny-Net extracts the powerful features from inputs quickly, and the intermediate global attention block enlarges the receptive field to inhibit false positives. The classifier first predicts the existence of detection target in the current patch, and the specifically designed detector is followed to locate them accurately if available. The high recall and precision in GF-1 and GF-2 validate the effectiveness of our network. Specifically, we can process a GF-1 image in 29.4 s on Titian X just with single thread. All those experiments prove that our $\mathcal{R}^2$-CNN is *efficient* in both computation and memory consumption, *robust* to false positives, and *strong* to detect tiny objects.

References

[1] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, "Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 6, pp. 3325–3337, Jun. 2015.

[2] Y. Long, Y. Gong, Z. Xiao, and Q. Liu, "Accurate object localization in remote sensing images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 5, pp. 2486–2498, May 2017.

[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. Zhang, B. Du, L. Zhang, and M. Xu, "Weakly supervised learning based on coupled convolutional neural networks for aircraft detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 9, pp. 5553–5563, Sep. 2016.

[5] Z. Lei, T. Fang, H. Huo, and D. Li, "Rotation-invariant object detection of remotely sensed images based on texton forest and Hough voting," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 4, pp. 1206–1217, Apr. 2012.

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

[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. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*. IEEE Computer Society, vol. 1, 2005, pp. 886–893.

[11] Z. Xiao, Q. Liu, G. Tang, and X. Zhai, "Elliptic Fourier transformation-based histograms of oriented gradients for rotationally invariant object detection in remote-sensing images," *Int. J. Remote Sens.*, vol. 36, no. 2, pp. 618–644, 2014.

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

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

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

[15] M. Lin, Q. Chen, and S. Yan. (2013). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400

[16] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[17] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. (2013). "OverFeat: Integrated recognition, localization and detection using convolutional networks." [Online]. Available: https://arxiv.org/abs/1312.6229

[19] 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.

[20] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2014, pp. 346–361.

[22] R. Girshick. (2015). "Fast R-CNN." [Online]. Available: https://arxiv.org/abs/1504.08083

[23] 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.*, 2016, pp. 779–788.

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

[25] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, Jul. 2017, vol. 1, no. 2, pp. 2117–2125.

[26] S. Zhang, X. Zhu, Z. Lei, H. Shi, X. Wang, and S. Z. Li. (2017). "S^3FD: Single shot scale-invariant face detector." [Online]. Available: https://arxiv.org/abs/1708.05237

[27] P. Hu and D. Ramanan, "Finding tiny faces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 951–959.

[28] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2881–2890.

[29] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 761–769.

[30] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, 2015.

[31] L. Zhang, Z. Shi, and J. Wu, "A hierarchical oil tank detector with deep surrounding features for high-resolution optical satellite imagery," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, no. 10, pp. 4895–4909, Oct. 2015.

[32] T. Ishii, R. Nakamura, H. Nakada, Y. Mochizuki, and H. Ishikawa, "Surface object recognition with CNN and SVM in Landsat 8 images," in *Proc. 14th IAPR Int. Conf. Mach. Vis. Appl. (MVA)*, May 2015, pp. 341–344.

[33] I. Ševo and A. Avramović, "Convolutional neural network based automatic object detection on aerial images," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 5, pp. 740–744, May 2016.

[34] A.-B. Salberg, "Detection of seals in remote sensing images using features extracted from deep convolutional neural networks," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 1893–1896.

[35] 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.

[36] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, "Orientation robust object detection in aerial images using deep convolutional neural network," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2015, pp. 3735–3739.

[37] Q. Jiang, L. Cao, M. Cheng, C. Wang, and J. Li, "Deep neural networks-based vehicle detection in satellite images," in *Proc. Int. Symp. Bioelectron. Bioinf. (ISBB)*, Oct. 2015, pp. 184–187.
[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.
[39] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4898–4906.
[40] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.
[41] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

Jiangmiao Pang received the B.S. degree from Tianjin University, Tianjin, China, in 2016. He is currently pursuing the Ph.D. degree with Zhejiang University, Hangzhou, China.

His research interests include computer vision and deep learning, especially on object detection, semantic segmentation, remote sensing, and autonomous driving.

Cong Li received the M.Sc. degree from Tsinghua University, Beijing, China, in 2014.

He is currently a Senior Researcher with SenseTime Research, Beijing, China. His research interests include computer vision and 3-D reconstruction, especially on segmentation, detection in remote sensing, and structure from motion.

Jianping Shi received the B.S. degree from Zhejiang University, Hangzhou, China, in 2011, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2015.

She is currently a Research Director with SenseTime Research, Beijing, China. Her research interests include fundamental algorithms and practical systems for autonomous driving, perception, localization, mapping, decision and planning, and control.

Zhihai Xu received the B.S., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1986, 1989, and 1996, respectively.

He is currently a Professor with the State Key Laboratory of Modern Optical Instrumentation, Zhejiang University. His research interests include optical remote sensing and imaging chain of cameras.

Huajun Feng received the B.S. and M.Sc. degrees from Zhejiang University, Hangzhou, China, in 1983 and 1986, respectively.

He is currently a Professor with the State Key Laboratory of Modern Optical Instrumentation, Zhejiang University. His research interests include imaging technique, image processing, precision testing technology, and optical system design.