Object Detection in Large-Scale Remote Sensing Images With a Distributed Deep Learning Framework

Linkai Liu, Yuanxing Liu, Jining Yan, *Member, IEEE*, Hong Liu, Mingming Li, Jinlin Wang, and Kefa Zhou

***Abstract*—With the accumulation and storage of remote sensing images in various satellite data centers, the rapid detection of objects of interest from large-scale remote sensing images is a current research focus and application requirement. Although some cutting-edge object detection algorithms in remote sensing images perform well in terms of accuracy, their inference speed is slow and requires high hardware requirements that are not suitable for real-time object detection in large-scale remote sensing images. To address this issue, we propose a fast inference framework for object detection in large-scale remote sensing images. On the one hand, we introduce α-IoU Loss on the YWCSL model to implement adaptive weighted loss and gradient, which achieves 64.62% and 79.54% mAP on DIOR-R and DOTA test sets, respectively. More importantly, the inference speed of the YWCSL model reaches 60.74 FPS on a single NVIDIA GeForce RTX 3080Ti, which is 2.87 times faster than the current state-of-the-art one-stage detector S^{2}A-Net. On the other hand, we build a distributed inference framework to enable fast inference on large-scale remote sensing images. Specifically, we save the images on HDFS for distributed storage and deploy the YWCSL model to the Spark cluster. When using 5 nodes, the speedup of the cluster reaches 9.54, which is 90.80% higher than the theoretical linear speedup (5.00). Our distributed inference framework for large-scale remote sensing images significantly reduces the dependence of object detection on expensive hardware resources, which has important research significance for the wide application of object detection in remote sensing images.**

***Index Terms*—CSL, object detection, remote sensing images, Spark, YOLOv5.**

I. Introduction

OBJECT detection has always been a research hotspot in computer vision. It is frequently used in practical problems, such as pedestrian detection, industrial detection, building detection, etc. With the continuous development of remote sensing technology, large-scale and high-resolution remote sensing image datasets have emerged, such as DOTA [1], DIOR-R [2], HRSC2016 [3], etc. These high-resolution remote sensing image datasets are rich in ground objects, including common ground objects such as airplanes, ships, vehicles, and basketball courts. It is of great significance to classify [4], [5], segment and detect [6] these remote sensing images. Object detection technology in remote sensing images [7] has been widely used in land planning, maritime fisheries, military reconnaissance, deforestation detection, and other fields [8], [9]. However, in practical applications, object detection in remote sensing images is more challenging than in natural scenes [7]. This is mainly reflected in the following difficulties:

a) Greater Detection Difficulty: Compared with natural scene images, the background of remote sensing images is more complex, and there are many small objects. The objects in remote sensing images have the characteristics of arbitrary orientations, scale variations, extremely uneven distributions, and large aspect ratios, which undoubtedly bring more difficulties to object detection in remote sensing images.

b) Higher Inference Costs: Object detection in remote sensing images often involves large-scale datasets (TB or even PB level) [10] in practical applications, which require a high data processing speed. In addition, remote sensing images have a high resolution [11], and direct input into the model for inference will take up a lot of memory. Usually, the original image needs to be cut before being fed into the model for inference, which will undoubtedly further increase the time overhead of detection. More importantly, the current state-of-the-art object detection models in remote sensing images generally rely on expensive GPU hardware resources in the actual inference process to achieve a tolerable inference speed, which significantly increases the practical application cost of object detection models in remote sensing images. It is one of the biggest obstacles to the widespread application of object detection models in remote sensing images to practical tasks.

In recent years, many researchers have made outstanding achievements in the field of object detection in remote sensing images. For example, cutting-edge algorithms, such as RetinaNet OBB [12], Cascade Mask R-CNN [13], and RoI Transformer [14], have achieved high accuracy on the DOTA-v1.0 dataset.

However, most of the algorithms presented above are two-stage detectors, which achieve better detection accuracy results than one-stage detectors. However, they are generally slower in

Manuscript received 31 May 2022; revised 20 August 2022; accepted 9 September 2022. Date of publication 12 September 2022; date of current version 27 September 2022. This work was supported by the National Natural Science Foundation of China under Grant 41925007 and Grant U21A2013. *(Corresponding author: Jining Yan.)*

Linkai Liu, Yuanxing Liu, Jining Yan, and Hong Liu are with the School of Computer Science, China University of Geoscience, Wuhan 430074, China, and also with the Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China (e-mail: 20191000627@cug.edu.cn; liuyx@cug.edu.cn; yanjn@cug.edu.cn; liuhong@cug.edu.cn).

Mingming Li is with the Soarscape Technology Development (Shanghai) Company, Ltd., Chinese Academy of Sciences, Urumqi 830011, China (e-mail: li.mingming@soarscape.com).

Jinlin Wang and Kefa Zhou are with the Xinjiang Key Laboratory of Mineral Resources and Digital Geology and Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi 830011, China (e-mail: wangjinlin@ms.xjb.ac.cn; zhoukf@ms.xjb.ac.cn).

Digital Object Identifier 10.1109/JSTARS.2022.3206085

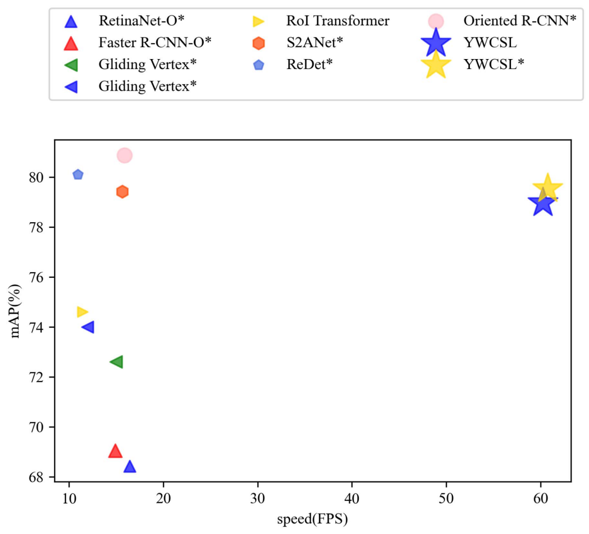


Fig. 1. Comparison of detection accuracy and inference speed between the YWCSL model and some current state-of-the-art models on the DOTA-v1.0 test set. For inference speed, we test all models on a single NVIDIA GeForce RTX 3080 Ti, with ReDet, RetinaNet-O, Gliding Vertex, RoI Transformer models from https://github.com/open-mmlab/mmrotateMMRotate, S^2A-Net, Oriented R-CNN and Faster R-CNN-O from https://github.com/jbwang1997/OBBDetectionOBBDetection. "*" means using multiscale training and testing.

terms of inference speed. More importantly, these algorithms require a high hardware environment, which increases the hardware cost of object detection models in practical applications. In contrast, not only does the algorithm of YOLOv5 combined with circular smooth label (CSL) [15] have excellent detection accuracy [16] (slightly lower than ReDet [17]), but also its performance far exceeds the inference speed of most models (see Fig. 1). It is more suitable for object detection tasks in large-scale remote sensing images. Therefore, we use "YOLOv5 with CSL" [16] as the baseline for our follow-up work, which we refer to as YWCSL for convenience. Our contributions can be summarized as follows:

1) We introduce the α-IoU loss function [18] to regress the bounding box on top of the YWCSL model and analyze in detail how the α-IoU loss function improves the performance of the model. Our improvement work enables the YWCSL model to achieve 64.62%, 79.54%, and 76.28% mAP on the DOIR-R, DOTA-v1.0, and DOTA-v1.5 test sets, respectively, and when $0 < \alpha < 1$, the YWCSL model performs more excellently than baseline on the category of small and medium pixel sizes. More importantly, its detection accuracy on the DOTA-v1.0 test set is 0.12% higher than that of the current state-of-the-art one-stage detector S^2A-Net.
2) We fully demonstrate the excellent inference speed of YWCSL. We use the DOTA-v1.0 test set on a single NVIDIA GeForce RTX 3080 Ti to test the inference speed of some current state-of-the-art algorithms. As shown in Table IV, the YWCSL model achieves an inference speed of 60.74 Frames Per Second (FPS), which is 2.89 times faster than S^2A-Net and 2.83 times faster than Oriented R-CNN. This indicates that the YWCSL model is well suited for the object detection task in large-scale remote sensing images.
3) We deploy the YWCSL model in a Spark distributed cluster to implement a high-performance framework for object detection in large-scale remote sensing images. When tested with five nodes, the speedup ratio of the inference speed of our distributed inference framework reaches 190.8% of the theoretical linear speedup ratio, which greatly reduces the dependence of object detection in remote sensing images on expensive GPU resources [19] and reduces the hardware cost of practical applications. It plays a crucial role in promoting the wide application of object detection in remote sensing images.

We will introduce the excellent object detection algorithms in remote sensing images in recent years in Section II. Section III will introduce our improvements to the YOLOv5 model and detail the steps to deploy the model on a Spark cluster for distributed inference. In Section IV, we compare the detection accuracy and inference speed of YWCSL with other state-of-the-art models. In addition, we also analyze the impact of a different number of nodes and different dataset sizes on the speedup ratio of the distributed inference framework. Finally, we select remote sensing images of a 13.53 square km area near the Optics Valley Plaza in Wuhan, China, to test the detection performance of our distributed inference cluster in practical applications and compare the detection details of our model and baseline on some small objects.

II. Related Work

A. Object Detection in Remote Sensing Images

Object detection has always been a research hotspot in computational vision. The mainstream object detection models mainly include anchor-based and anchor-free detectors. Anchor-based detectors include one-stage detectors represented by the YOLO series [20] and two-stage detectors represented by Faster R-CNN [21]. Two-stage detectors usually rely on the Region Proposal Network (RPN) to generate high-quality Region of Interests (RoI), which is disadvantageous for real-time detection tasks. The one-stage detectors extract features directly in the convolutional neural network to predict object classification and location, which achieve a good balance between detection accuracy and inference speed. Therefore, the one-stage detectors of the YOLO series are especially widely used in real-time detection tasks. Another type is the anchor-free model, which is represented by CornerNet [22], ExtremeNet [23], etc.

Nowadays, with the rapid development of object detection technology, a series of object detection models suitable for remote sensing images have been proposed, including R2CNN [24], RRPN [25], and SCRDet [26], which are all improved based on Faster R-CNN. In addition, there are some more advanced detection algorithms, such as CSL [15], DCL [27], RoI Transformer [14], ReDet [17]. They all show good performance on various remote sensing datasets, such as DOTA and DIOR-R. Unlike traditional object detection tasks, objects in remote sensing images are more intensive and have different directions. To align objects more accurately, the oriented bounding box

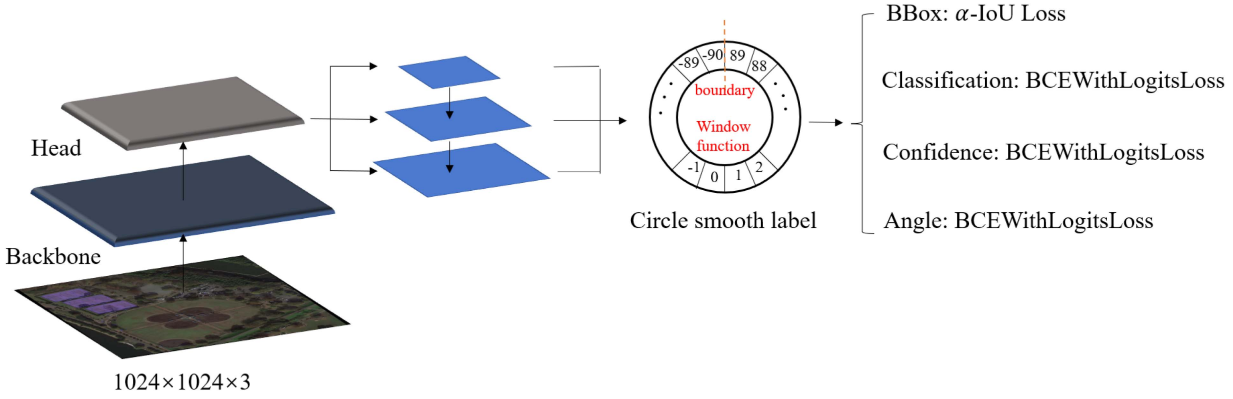


Fig. 2. YWCSL algorithm.

(OBB) is often used instead of the horizontal bounding box (HBB) in detection tasks to wrap the objects in remote sensing images.

For the oriented object detection, Ding et al. [14] proposed the Rotated Region of Interests (RRoI) learner module to learn RRoI from the feature map of the Horizontal Region of Interests. In addition, they proposed a Rotated Position Sensitive RoI Alignment module to extract rotation-invariant object features and finally achieved 69.56% mAP on the DOTA-v1.0 test set, and their work provides essential ideological support for the ReDet algorithm.

Yang et al. [15] discretized the regression of the OBB angle as a classification task and solve the periodicity of angular (PoA) problem using a window function with periodicity. They use the FPN-CSL-based model to improve mAP to 76.17% on the DOTA-v1.0 test set.

On the basis of previous work, Han et al. [17] proposed the Rotation-invariant RoI Align (RiRoI Align) module, which adaptively extracts rotation-invariant features from rotation-equivariant features according to the direction of RoI and improves mAP to 80.10% on the DOTA-v1.0 test set, while significantly reducing the size of the model.

Most of the cutting-edge algorithms mentioned above are two-stage models. Although they bring much improvement in accuracy, they also bring a lot of inference time overhead. Aiming at the balance between model inference speed and accuracy, Han et al. [28] proposed a high-performance one-stage detection model S^2A-Net. They proposed a feature alignment module (FAM) to generate high-quality anchors and used an oriented detection module (ODM) to alleviate the inconsistency between classification scores and localization accuracy, achieving 79.42% mAP on the DOTA-v1.0 test set and improving the inference speed to 16.0 FPS (test on a single NVIDIA Tesla V100 GPU). In addition, Wen et al. [29] chose to use the CSL module to improve the YOLOv5 model (see Fig. 2) and used data enhancement methods such as mosaic to participate in the training. The improved YOLOv5 model achieves 58.2% mAP on the DOTA-v2.0 test-dev set, which is an excellent detection performance. The open-source model in [16] uses a similar approach to combine the YOLOv5 model with CSL (YWCSL) and exhibits 77.30% mAP and 73.19% mAP on the DOTA-v1.0 and DOTA-v1.5 test sets, respectively. Moreover, YWCSL retains the excellent inference speed of YOLOv5, which is very suitable for object detection tasks in large-scale remote sensing images.

B. Distributed Deep Learning Computing Framework

As a mainstream Big Data computing framework, Apache Spark [30] provides efficient Big Data processing capabilities. Unlike Hadoop [31], the intermediate data generated during Spark calculation is directly stored in the memory, reducing the time spent interacting with disks. More importantly, Spark generates a directed acyclic graph (DAG) internally based on program execution logic, which significantly speeds up distributed parallel computing. Therefore, Spark's computing speed is usually much higher than Hadoop's.

Nowadays, machine learning has been integrated into all walks of life. For example, Liu et al. [32] used KNN-XGBoost to predict missing values, which played an essential role in the task of detecting transmission line risks in smart grids. Song et al. [33] proposed a Bi-CLKT to track students' learning, which helps education departments to formulate systematic learning plans for students. However, most of the above work is based on the traditional single-machine computing mode, which is not good at processing Big Data as the amount of data in the actual task increases rapidly. Instead, Spark's distributed computing model, which combines a Big Data processing framework with machine learning algorithms, has been widely used in various industries. The MLlib module of Spark provides a wealth of machine learning algorithms, such as Kmeans, ASL, SVM, and other mainstream algorithms. Users can easily use various machine learning algorithms in the MLlib module by calling a simple interface. In addition, a more efficient deep learning framework suitable for Big Data processing has become one of the current research hotspots. In recent years, many well-known distributed deep learning computing frameworks have been proposed, such as Google's DistBelief [34], Baidu's DeepImage [35], SparkNet [36], TensorFlowOnSpark [37], and BigDL [38], etc.

1) TensorFlowOnSpark: TensorFlowOnSpark is a distributed deep learning computing framework developed by Yahoo, which supports all functions in TensorFlow, including model parallelization, data parallelization, and synchronous or asynchronous training and inference. It uses server-to-server direct communication to speed up the operation, supports Hadoop and Spark clusters, and implements distributed deep learning on GPU and CPU clusters. TensorFlowOnSpark provides distributed TensorFlow training and inference for Spark clusters but currently does not support other mainstream deep learning frameworks such as Pytorch.

2) BigDL: Dai et al. [38] point out that the current mainstream distributed deep learning frameworks (CaffeOnSpark [39], TensorFlowOnSpark [37], etc.) all adopt a "connector approach" method and use an integrated workflow to develop suitable interfaces to connect different data processing and deep learning components. In practice, the adaptation of other frameworks will bring a lot of overhead (such as data serialization, persistence, and interprocess communication). More importantly, this training mode will have the problem of impedance mismatches [40]. Big data systems and deep learning systems have very different execution modes. Tasks in Big Data systems are parallel and independent, while in deep learning systems, tasks are coordinated and interdependent. When a worker in Spark fails, the tasks will be restarted, which may cause the entire workflow of the system to block indefinitely. Therefore, BigDL takes a different approach to directly implementing distributed deep learning support in Big Data systems, eliminating the impedance mismatch problem. BigDL has realized support for mainstream frameworks, such as Pytorch and TensorFlow, which significantly improve the development and operation efficiency of Big Data deep learning applications.

However, whether TensorFlowOnSpark, BigDL, or other mainstream distributed deep learning frameworks, they still face many problems during the training phase [38], [40]. For example, TensorFlowOnSpark and similar distributed deep learning frameworks using the "connector approach" training mode face the above impedance mismatches problem. BigDL does not support training on distributed GPU clusters. Moreover, with the frequent updates of deep learning frameworks such as TensorFlow, various new versions of BigDL need to be maintained frequently. At present, the maintenance of the BigDL community is not enough, and the community users are not active enough, which shows that the technology of applying the distributed deep learning framework to the training phase is not mature enough. Of course, we do not deny this kind of distributed training mode here. On the contrary, we firmly believe that this kind of distributed deep learning training technology will soon get widespread attention and greatly promote the development of deep learning after it is perfected.

Given the impedance mismatches and other issues associated with applying deep learning tasks to training on Big Data clusters, using the traditional stand-alone model for training and inference on Big Data clusters seems to be a better choice. Practice has proved that this is a very convenient and low-cost way to fully demonstrate the performance of object detection models in remote sensing images.

In this work, we encapsulate the inference of the model into a generic function, and each node of the cluster can execute this function independently and in parallel, and their execution resources are uniformly managed by YARN. As a result, we implement a lightweight and high-performance distributed inference framework, which is very suitable for object detection in large-scale remote sensing images. In addition, to improve the detection accuracy of the model on remote sensing images, we introduce the α-IoU [18] loss function to improve the detection effect of the model on small object detection. We will introduce our work in detail in the next section.

III. Method

In this section, we first introduce the nature of α-IoU and analyze the effect of different α values on the loss and gradient of the training process in Section III-A. Next, we detail the principles of the distributed inference framework, including data storage, resource management, and model inference steps, in Section III-B. We use cheap CPU cluster resources to build a lightweight and high-performance distributed inference framework suitable for object detection in large-scale remote sensing images, which significantly reduces the dependence of the model on expensive GPU resources in practical applications and has essential research significance for the practical application of object detection in remote sensing images.

A. α-IoU [18]

In the Anchor-based detector, the bounding box regression and object classification are usually divided into two subtasks for learning. In the bounding box regression subtask, the localization loss is generally calculated according to the Intersection over Union (IoU) of the bounding boxes and the ground truths. Since the traditional IoU loss has the problem of gradient vanishing when there is no overlapping area between the bounding boxes, scholars have successively proposed different loss functions, such as generalized intersection over union (GIoU) [41], distance-IoU (DIoU), and complete IoU (CIoU) [42] to solve this problem.

In GIoU, Rezatofigh et al. [41] introduced the minimum enclosing rectangle of the predicted bounding box and the ground truth to reflect their coincidence degree and solve the problem of gradient vanishing caused by the loss of zero when the predicted bounding box and the ground truth do not overlap. In DIoU [42], the central point distance, overlap rate, and scale between the predicted bounding box and the ground truth are also considered, which makes the bounding box regression more stable to achieve a faster convergence speed than GIoU. Based on DIoU, not only the overlap area, central point distance, but also the aspect ratio between bounding boxes are considered in CIoU.

The α-IoU used in this article can cover all of the above IoU loss functions [18]. In addition, the loss and gradient of high-IoU objects and low-IoU objects can be adaptively weighted to further improve the performance of the model. α-IoU can be defined as follows:

$$\mathbb{L}_{\alpha-\mathrm{IoU}} = \frac{1-\mathrm{IoU}^{\alpha}}{\alpha}, \alpha > 0. \tag{1}$$

This approach introduces three important properties as follows [18]:

1) *Order Preservingness*

It is not difficult to obtain that α-IoU loss is the same as GIoU, DIoU, and other loss functions. With the increase

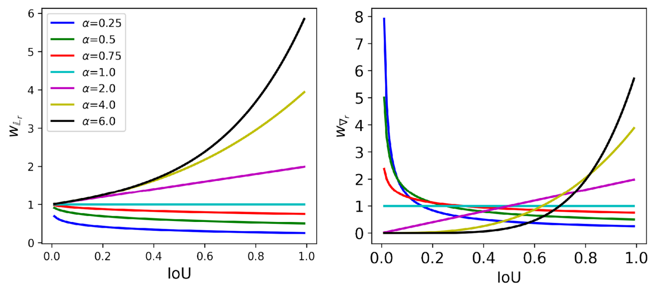


Fig. 3. Influence of different α values on the weight factors of loss and gradient.

of IoU, the loss decreases monotonically. When $\alpha = 1.00$, the α-IoU loss becomes the traditional IoU loss.

2) *Relative Loss Reweighting*
α-IoU loss can be regarded as IoU loss multiplied by a weighting factor. The definition of the weighting factor can be expressed as follows:

$$w_{\mathbb{L}_r} = \frac{\mathbb{L}_{\alpha-\mathrm{IoU}}}{\mathbb{L}_{\mathrm{IoU}}} = 1 + \frac{(\mathrm{IoU} - \mathrm{IoU}^{\alpha})}{(1 - \mathrm{IoU})}. \quad (2)$$

The above formula can be obtained that $\lim_{\mathrm{IoU}\to 0} w_{\mathbb{L}_r} = 1$ and $\lim_{\mathrm{IoU}\to 1} w_{\mathbb{L}_r} = \alpha$. When $\alpha > 1$, $w_{\mathbb{L}_r}$ will increase the loss weight of high-IoU objects as IoU increases, making the model pay more attention to objects with high IoU (see Fig. 3). On the contrary, when $0 < \alpha < 1$, $w_{\mathbb{L}_r}$ will reduce the loss weight of high-IoU objects as IoU increases, and the model naturally pays more attention to regressing the bounding boxes of low-IoU objects.

3) *Relative Gradient Reweighting*
Similar to the second property, α-IoU can also adaptively weight the gradient. The weighting factor is defined as follows:

$$w_{\nabla_r} = \frac{|\nabla_{\mathrm{IoU}}\mathbb{L}_{\alpha-\mathrm{IoU}}|}{|\nabla_{\mathrm{IoU}}\mathbb{L}_{\mathrm{IoU}}|} = \alpha \mathrm{IoU}^{\alpha-1}. \quad (3)$$

When $0 < \alpha < 1$, the weighting factor decreases with the increase of IoU, and when $\alpha > 1$, the weighting factor increases with the increase of IoU. It is worth noting that when $\alpha \neq 1$, the α-IoU loss function re-weights the gradient of the object according to the IoU, and then adjusts the learning speed of different objects.

In order to verify the above theory, we will show the performance of the YWCSL model under different α values through experiments in Section IV.

B. Distributed Inference Framework for Object Detection in Large-Scale Remote Sensing Images

The distributed inference framework for object detection in large-scale remote sensing images built in this article mainly includes a data storage layer, data loading and preprocessing layer, distributed computing layer, and visualization layer (see Fig. 4).

We store large-scale remote sensing image data in HDFS to achieve distributed storage and ensure data reliability and high fault tolerance in the data storage layer. In the data loading layer, we use the interfaces in RasterFrames [43] and PyHDFS to interact with HDFS to realize data reading and writing of conventional remote sensing image formats such as tif, png, and jpeg.

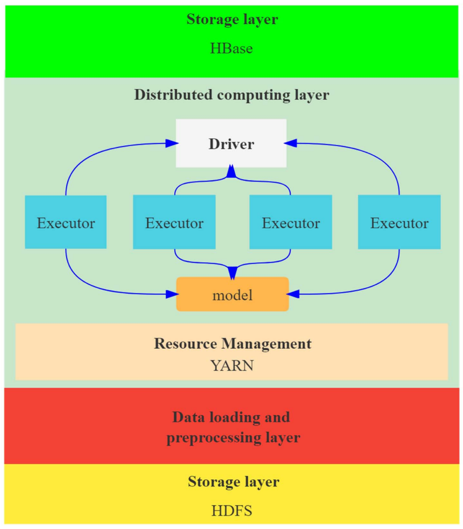


Fig. 4. Distributed Inference framework.

For the distributed computing layer, we choose Apache Spark [30] as the underlying computing engine to build a distributed and fast inference framework suitable for Big Data processing. We deploy the pretrained YWCSL model to each worker node of the Spark distributed cluster. When the program starts to execute, the Driver process will assign the task to the corresponding worker according to the data location to minimize the network transmission overhead. Each worker will load the pretrained YWCSL model into memory and load the RDD partition data to execute the task (see Algorithm 1). The number of tasks is related to the number of partitions of the RDD, which determines the program's parallelism and affects the program's running time. We adopt a custom partitioner (RankPartition) based on self-incrementing primary key modulo to distribute the remote sensing image data into each partition as evenly as possible. As shown in Fig. 11, we built a three-node Spark cluster and compared Spark clusters' inference speed using two different partitioners: RankPartition and HashPartition. To more clearly demonstrate the advantage of RankPartition over HashPartition, we define the inference speedup of Spark clusters using HashPartition partitioners as 1.0 and calculate the speedup ratio of the inference speedup of Spark clusters when using RankPartition. RankPartition is significantly more efficient than the default HashPartition. In the case of the same number of CPU cores, the larger the amount of data, the more pronounced the improvement in inference speed brought by RankPartition.

For the visualization layer, we choose to store the inference results of the model in HBase [44] to support fast querying of massive unstructured data. Users can query the specified remote sensing images in real time according to attributes such as area name and geographic location and visualize the inference results of the model to the front end for further data analysis.

The inference steps of the distributed inference framework for object detection in large-scale remote sensing images implemented in this article are as follows:

Algorithm 1: Principle of Distributed Inference Framework.

```
1:  Initialize numPartition and pID;
2:  // RankPartition;
3:  for each image path in HDFS do
4:      id = pID++ % numPartition;
5:      Send image path information for $task_{id}$;
6:  end for;
7:  for each Worker ∈ Spark cluster do
8:      Worker ← CPU cores and memory, etc.
9:      Worker pull tasks and loading model;
10:     for each batch image paths ∈ Worker's task do
11:         images ← getImage(image paths);
12:         images ← Processing(images);
13:         results = model(images);
14:         if the images is split then
15:             Send the results to Driver for the next merge
                operation.
16:         else
17:             Write the results directly to HBase.
18:         end if;
19:     end for;
20: end for;
```

1) First, split the original image into segments of 1024×1024, and store these segmented images and segment information in the specified path of HDFS.
2) The Driver reads the path information of the above image fragments, builds a DataFrame, and uses the RankPartition partitioner to repartition the DataFrame. The model inference process is encapsulated as a UDF function (including an image loading module, a data processing module, a model Inference module, and non-maximum suppression (NMS) module), the input is batch path information, and the output is the model inference result (including prediction bounding box coordinates, category, confidence, and other information).
3) Yarn is responsible for cluster resource management, assigning program running resources to each Worker, and the Worker pulls the task and calls the above UDF function. The model results are fed back to the Driver.
4) The Driver summarizes the model inference results of each shard and merges the results according to the shard information. Each Worker, in parallel, can perform the merging operation.
5) The merged results are directly written to HBase after the last NMS.

The above execution process has a high fault tolerance mechanism. When an exception occurs in a Worker, the Driver will receive the exception feedback and call other available workers to restart the abnormal task.

IV. Experiments

A. Datasets

To validate our proposed method, we conduct experiments using two large-scale remote sensing datasets, DIOR-R [2] and DOTA [1]. The DIOR-R dataset is an extended version of the DIOR dataset [45] with a total of 23 463 images and 192 518 object instances. All objects are annotated with a rotating bounding box, which contains 20 object categories: Airplane (APL), Airport (APO), Baseball Field (BF), Basketball Court (BC), Bridge (BR), Chimney (CH), Dam (DAM), Expressway Service Area (ESA), Expressway Toll Station (ETS), Golf Field (GF), Ground Track Field (GTF), Harbor (HA), Overpass (OP), Ship (SH), Stadium (STA), Storage Tank (STO), Tennis Court (TC), Train Station (TS), Vehicle (VE), and Windmill (WM). The image size of the DIOR-R dataset is 800×800, and the dataset is divided into a trainval set and a test set. The trainval set contains 11 725 images and 68 073 instances, and the test set contains 11 738 images and 124 445 instances.

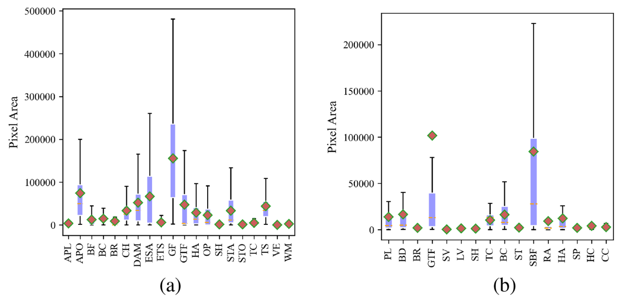


Fig. 5. Size of each category in DIOR-R and DOTA datasets. (a) DIOR-R dataset. (b) DOTA dataset.

There are three versions of the DOTA dataset: DOTA-v1.0, DOTA-v1.5, and DOTA-v2.0. The first version contains 2806 images, 188 282 instances, and a total of 15 categories of objects: plane (PL), baseball diamond (BD), bridge (BR), ground track field (GTF), small vehicle (SV), large vehicle (LV), ship (SH), tennis court (TC), basketball court (BC), storage tank (ST), soccer ball field (SBF), roundabout (RA), harbor (HA), swimming pool (SP), and helicopter (HC). The train, validation, and test datasets contain 1411, 458, and 937 images. DOTA-v1.5 uses the same data as DOTA-v1.0 but adds many small objects (less than 10 pixels) and a new category container crane (CC) containing 402 089 instances. The division of the training set, validation set, and test set is consistent with the DOTA-v1.0 version. Compared with DOTA-v1.5, DOTA-v2.0 adds two categories, airport (Air) and helipad (Heli), with 11 268 images and 1 793 658 instances. The training, validation, test-challenge, and test-dev sets contain 1830, 593, 6053, and 2792 images and 268 627, 81 048, 1 090 637, and 353 346 instances, respectively. The size of each category of the above two remote sensing datasets is shown in Fig. 5.

B. Implementation Details

For the DIOR-R dataset, we directly use the original images of the trainval dataset to train for 100 epochs, and then use the original test set to validate the performance of the model. The image size of the DOTA dataset is between 800×800 and 20 000×20 000, considering the large memory footprint associated with feeding images directly into the network, we slice the original image into a series of 1024×1024 slices with a stride of 824. For multiscale testing, we scale the original test

TABLE I
COMPARISON WITH STATE-OF-THE-ART METHODS ON DIOR-R DATASET

Methods	Backbone	APL	APO	BF	BC	BR	CH	DAM	ETS	ESA	GF	GTF	HA	OP	SH	STA	STO	TC	TS	VE	WM	mAP
Faster R-CNN-O [48]	R-50-FPN	62.79	26.8	71.72	80.91	34.2	72.57	18.95	66.45	65.75	66.63	79.24	34.95	48.79	81.14	64.34	71.21	81.44	47.31	50.46	65.21	59.54
RetinaNet-O [49]	R-50-FPN	61.49	28.52	73.57	81.17	23.98	72.54	19.94	72.39	58.2	69.25	79.54	32.14	44.87	77.71	67.57	61.09	81.46	47.33	38.01	60.24	57.55
Gliding Vertex [50]	R-50-FPN	65.35	28.87	74.96	81.33	33.88	74.31	19.58	70.72	64.7	72.3	78.68	37.22	49.64	80.22	69.26	61.13	81.49	44.76	47.71	65.04	60.06
RoI Transformer [14]	R-50-FPN	63.34	37.88	71.78	87.53	40.68	72.6	26.86	78.71	68.09	68.96	**82.74**	47.71	55.61	81.21	**78.23**	70.26	**81.61**	54.86	43.27	65.52	63.87
AOPG [2]	R-50-FPN	62.39	37.79	71.62	**87.63**	40.9	72.47	31.08	65.42	**77.99**	73.2	81.94	42.32	54.45	81.17	72.69	71.31	81.49	**60.04**	52.38	69.99	64.41
DODet [51]	R-50-FPN	63.4	43.35	72.11	81.32	43.12	72.59	**33.32**	78.77	70.84	**74.15**	75.47	**48.00**	**59.31**	**85.41**	74.04	71.56	81.52	55.47	51.86	66.4	**65.1**
YWCSL (α=0.5)	CSPDarkNet53	70.81	42.67	79.12	81.42	44.28	74.99	27.82	**79.33**	74.59	64.31	74.87	34.51	58.28	72.07	71.41	77.50	81.46	47.22	56.58	**73.28**	64.33
YWCSL (α=0.75)	CSPDarkNet53	71.61	**43.49**	**79.22**	81.50	**45.44**	**76.81**	29.65	78.81	74.70	65.21	75.96	35.61	56.98	72.08	70.15	78.18	81.47	46.86	56.02	72.62	64.62
YWCSL (α=1.0)	CSPDarkNet53	71.37	42.91	72.17	87.06	44.41	75.98	30.49	79.04	74.95	64.74	75.17	35.84	58.39	72.03	69.30	78.81	81.43	45.61	**56.62**	72.83	64.46
YWCSL (α=1.5)	CSPDarkNet53	71.30	41.75	72.09	81.35	42.34	72.41	28.70	78.45	73.49	64.55	75.89	35.56	56.48	72.06	70.38	75.04	81.42	45.33	56.18	70.41	63.26
YWCSL (α=2.0)	CSPDarkNet53	63.40	41.41	72.08	81.24	41.43	75.65	28.88	77.32	74.30	63.80	76.07	33.81	56.14	72.02	70.93	**79.16**	81.34	46.48	55.51	72.34	63.17
YWCSL (α=2.5)	CSPDarkNet53	71.63	41.19	71.98	81.33	40.42	72.44	26.98	76.06	72.67	62.26	75.26	34.06	56.19	71.99	70.02	78.96	81.34	46.37	54.92	71.67	62.89
YWCSL (α=3.0)	CSPDarkNet53	**71.85**	37.73	71.94	81.43	38.98	72.49	27.28	74.18	70.62	59.22	75.07	33.38	55.28	71.97	71.67	68.39	81.29	39.98	53.80	63.69	61.01

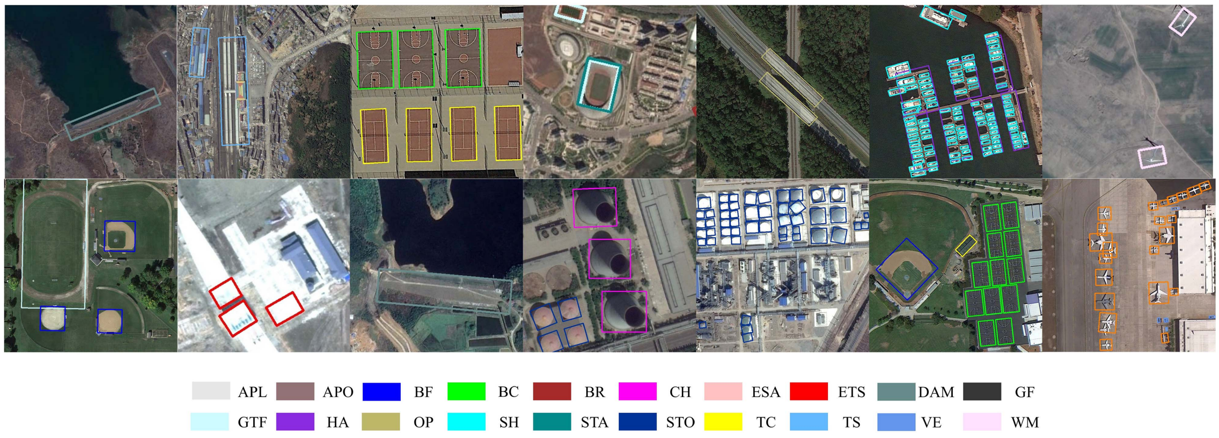


Fig. 6. Detection details of partial images in DIOR-R test set.

TABLE II
COMPARE THE ACCURACY OF THE STATE-OF-THE-ART MODEL ON THE DOTA-V1.0 TEST SET

Methods	Backbone	mAP	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC
DRN [52]	H-104	70.70	88.91	80.22	43.52	63.35	73.48	70.69	84.94	90.14	83.85	84.11	50.12	58.41	67.62	68.60	52.50
RSDet [53]	R-101-FPN	72.20	89.80	82.90	48.60	65.20	69.50	70.10	70.20	90.50	85.60	83.40	62.50	63.90	65.60	67.20	68.00
DAL [54]	R-50-FPN	71.44	88.68	76.55	45.08	66.80	67.00	76.76	79.74	90.84	79.54	78.45	57.71	62.27	69.05	73.14	60.11
S^2A-Net* [28]	R-50-FPN	79.42	88.89	83.60	57.74	**81.95**	79.94	83.19	89.11	90.78	84.87	87.81	70.30	68.25	78.30	77.01	69.58
R3Det [55]	R-101-FPN	73.79	88.76	83.09	50.91	67.27	76.23	80.39	86.72	90.78	84.68	83.24	61.98	61.35	66.91	70.63	53.94
RoI Transformer [14]	R-101-FPN	69.56	88.64	78.52	43.44	75.92	68.81	73.68	83.59	90.74	77.27	81.46	58.39	53.54	62.83	58.93	47.67
CSL(FPN based) [15]	ResNet152	76.17	90.25	85.53	54.64	75.31	70.44	73.51	77.62	90.84	86.15	86.69	69.60	68.04	73.83	71.10	68.93
CenterMap OBB [56]	ResNet-101	76.03	89.83	84.41	54.60	70.25	77.66	78.32	87.19	90.66	84.89	85.27	56.46	69.23	74.13	71.56	66.06
Li et al. [57]	ResNet101	76.36	**90.41**	85.21	55.00	78.27	76.19	72.19	82.14	90.70	87.22	86.87	66.62	68.43	75.43	72.70	57.99
OPLD [58]	ResNet101-FPN	76.43	89.37	85.82	54.10	79.58	75.00	75.13	86.92	90.88	86.42	86.62	62.46	68.41	73.98	68.11	63.69
Yu et al. [59]	Res2Net50	77.18	89.84	84.90	51.55	74.42	78.54	83.28	87.36	90.73	87.17	85.70	65.20	62.23	75.56	72.21	69.01
Gliding Vertex [50]	R-101-FPN	75.02	89.64	85.00	52.26	77.34	73.01	73.14	86.82	90.74	79.02	86.81	59.55	70.91	72.94	70.86	57.32
ReDet* [17]	ReR50-ReFPN	80.10	88.81	82.48	60.83	80.82	78.34	**86.06**	88.31	90.87	**88.77**	87.03	68.65	66.90	79.26	79.71	**74.67**
DODet* [51]	R-50-FPN	80.62	89.96	85.52	58.01	81.22	78.71	85.46	88.59	**90.89**	87.12	87.80	70.50	**71.54**	82.06	77.43	74.47
Oriented R-CNN [60]	R-50-FPN	**80.87**	89.84	85.43	**61.09**	79.82	79.71	85.35	88.82	90.88	86.68	87.73	**72.21**	70.80	**82.42**	78.18	74.11
YWCSL* (α=0.5)	CSPDarkNet53	79.54	89.22	86.48	54.47	72.26	81.09	85.47	89.04	90.75	88.34	87.76	69.74	67.35	76.27	**82.98**	71.84
YWCSL* (α=1.0)	CSPDarkNet53	79.05	88.5	**86.62**	53.19	70.84	80.83	85.71	**89.14**	90.84	88.06	88.38	68.67	69.93	76.22	81.38	67.4
YWCSL* (α=3.0)	CSPDarkNet53	79.43	88.79	86.33	52.95	68.85	**81.25**	85.61	89.13	90.74	88.07	88.03	70.13	68.80	76.02	82.56	74.17

set images by three scaling factors (0.5, 1.0, 1.5) and slice them into 1024×1024 slices with a stride of 512. We first train on the DOTA training set for 100 epochs, and then train on the training and validation sets for 50 epochs.

All models are trained on a single NVIDIA RTX 3080 Ti with a batch size of 8. We employ a stochastic gradient descent (SGD) optimizer with an initial learning rate of 0.01, momentum of 0.937, and weight decay of 0.0005. For training, we used data augmentation with random flips, Mosaic [46], Mixup [47], and no data augmentation for testing.

C. Comparison With State-of-the-Arts

a) Results on DIOR-R: As shown in Table I, when $\alpha = 0.75$, the YWCSL model achieves 64.62% mAP on the DIOR-R test set, which is 0.21% higher than AOPG. Compared with the state-of-the-art algorithm DODet, the YWCSL model has higher detection accuracy on small and medium-sized objects such as APL, APO, BF, BR, STO, ETS, WM, VE, and CH. Especially on the extremely small objects such as APL, BF, STO, and WM, the detection accuracy of the YWCSL model is 8.21%, 7.11%, 6.62%, 6.22% higher than that of the DODet model, respectively. The partial inference results of YWCSL on the DIOR-R test set are shown in Fig. 6.

b) Results on DOTA: As shown in Tables II and III, the YWCSL model can achieve up to 79.54% and 76.60% mAP on the DOTA-v1.0 and DOTA-v1.5 test sets, respectively. Notably, when $\alpha = 0.5$, YWCSL reaches the state-of-the-art of one-stage detectors, which is 0.12% higher than S^2A-Net. Compared with advanced algorithms such as Oriented R-CNN and DODet, the

TABLE III
COMPARE THE DETECTION ACCURACY ON THE DOTA-V1.5 TEST SET

Methods	BackBone	mAP	PL	BD	BR	GTF	SV	LV	SH	TC	BC	ST	SBF	RA	HA	SP	HC	CC
RoI Transformer [14]	R-101-FPN	65.03	71.92	76.07	51.87	69.24	52.05	75.18	80.72	90.53	78.58	68.26	49.18	71.74	67.51	65.53	62.16	9.99
USTC-NELSLIP [61]	ResNet-50	**78.34**	89.19	85.32	57.27	80.86	73.87	81.26	89.50	90.84	**85.94**	85.62	69.50	76.73	76.34	76.00	**77.84**	**57.33**
ReDet* [17]	ReR50-ReFPN	76.80	88.51	86.45	**61.23**	**81.20**	67.60	**83.65**	**90.00**	90.86	84.30	75.33	**71.49**	72.06	**78.32**	74.73	76.10	46.98
YWCSL* (α=0.5)	CSPDarkNet53	76.28	**89.70**	86.85	53.38	70.73	73.82	83.16	89.64	**90.89**	84.53	86.65	65.70	75.43	76.22	**77.11**	70.08	46.67
YWCSL* (α=1.0)	CSPDarkNet53	76.60	89.63	**87.13**	51.63	66.51	74.22	82.83	89.74	90.82	85.4	**86.81**	67.2	77.38	75.12	76.65	76.53	47.98
YWCSL* (α=3.0)	CSPDarkNet53	75.93	88.94	86.87	52.84	69.36	**74.65**	83.05	89.66	90.75	84.37	86.76	63.94	**77.59**	75.08	77.02	75.89	38.09

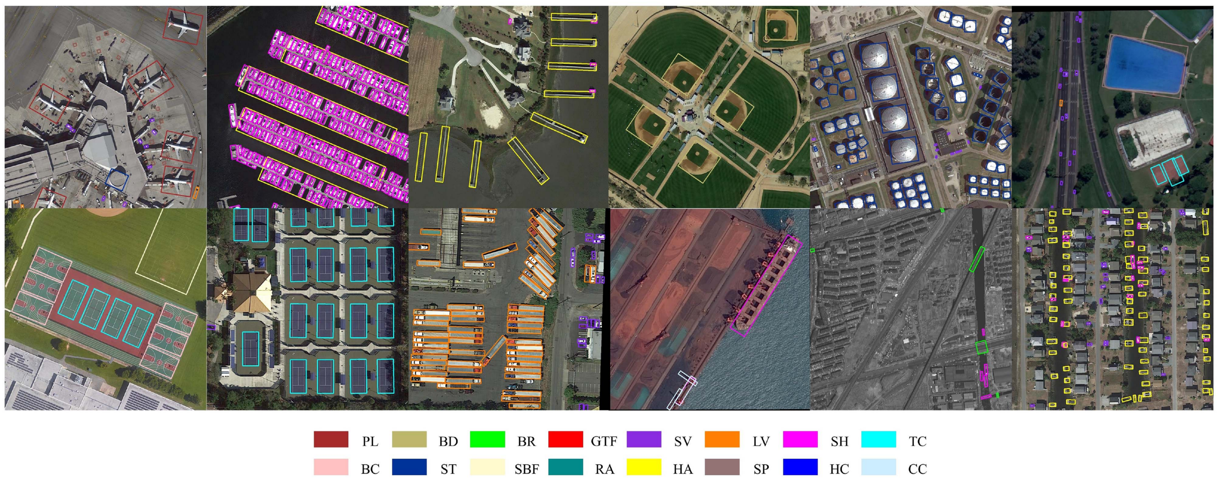


Fig. 7. Detection details of partial images in DOTA-v1.5 validation set.

TABLE IV
WE USE THE DOTA-V1.0 TEST SET TO TEST THE PERFORMANCE OF SOME CURRENT STATE-OF-THE-ART MODELS ON AN NVIDIA GEFORCE RTX 3080 TI

Methods	BackBone	Speed (FPS)	mAP
One-stage			
RetinaNet-O*	R-50-FPN	16.43	68.43
S2ANet*	R-50-FPN	15.63	79.42
Two-stage			
Faster R-CNN-O*	R-50-FPN	14.9	69.05
Gliding Vertex*	R-50-FPN	14.96	72.60
Gliding Vertex*	R-101-FPN	11.95	74.00
RoI Transformer	R-50-FPN	11.43	74.61
ReDet*	R-50-FPN	10.95	80.10
Oriented R-CNN*	R-50-FPN	15.87	**80.87**
Ours			
YWCSL	CSPDarkNet53	60.23	78.98
YWCSL*	CSPDarkNet53	**60.74**	79.54

* means using multiscale training and testing.

TABLE V
COMPARING THE ACCURACY OF MODELS WITH DIFFERENT IMPROVEMENT STRATEGIES ON THE TEST SET

	Different Settings			
CIoU Loss	✓		✓	
α-CIoU Loss		✓		✓
single scale test	✓	✓		
multi scale test			✓	✓
DOTA-v1.0 OBB mAP	77.09	77.61	78.00	78.77
DOTA-v1.5 OBB mAP	72.87	73.33	75.02	75.06

"Single Scale Test" means that the original image is not scaled, and the test set image is cut into 1024×1024 slices with a stride of 824. "Multi Scale Test" means to first scale the original image with a scaling factor of (0.5, 1.0, 1.5), and then cut the image into 1024×1024 slices with a stride of 512. For training with α-CIoU loss, α defaults to 0.50.

YWCSL model achieves superior detection accuracy on small objects such as SV, SH, ST, SP, and BD. The partial inference results of YWCSL on the DOTA test set are shown in Fig. 7

c) Comparison of Model Inference Speed: As shown in Table IV, the biggest advantage of the YWCSL model is its fast inference speed. We use the DOTA-v1.0 test set to test state-of-the-art algorithms on a single NVIDIA GeForce RTX 3080 Ti. The table shows that the inference speed of the YWCSL model reaches 60.74 FPS, which is 3.83 times that of the current state-of-the-art model, the Oriented R-CNN. More importantly, it still maintains a high detection accuracy, which is higher than the current state-of-the-art one-stage algorithm S^2A-Net. This shows that the YWCSL model is very suitable for the real-time detection task of large-scale remote sensing images.

D. Ablation Studies

a) The Effect of Different α Values on the Performance of YWCSL: We conduct ablation experiments on the DOTA dataset and the DIOR-R dataset, respectively. We train 100 epochs using the conventional CIoU loss and α-CIoU loss, respectively. As shown in Table V, the YWCSL model trained with α-CIoU loss is 0.52% and 0.46% mAP higher than the model trained with CIoU loss for the single scale processing of the DOTA-v1.0 and DOTA-v1.5 test sets, respectively, and when the multiscale expanded test set is used, the detection accuracy of the former is 0.77% and 0.04% mAP higher than that of the latter. Fig. 9(a) shows the detection accuracy of YWCSL in each epoch when α is 0.5/0.75/1.00/3.00. The results show that when $0 < \alpha < 1$, the detection accuracy of the YWCSL model improves faster.

According to Tables I–III, when $0 < \alpha < 1$, the detection accuracy of the YWCSL model is usually higher than that of the

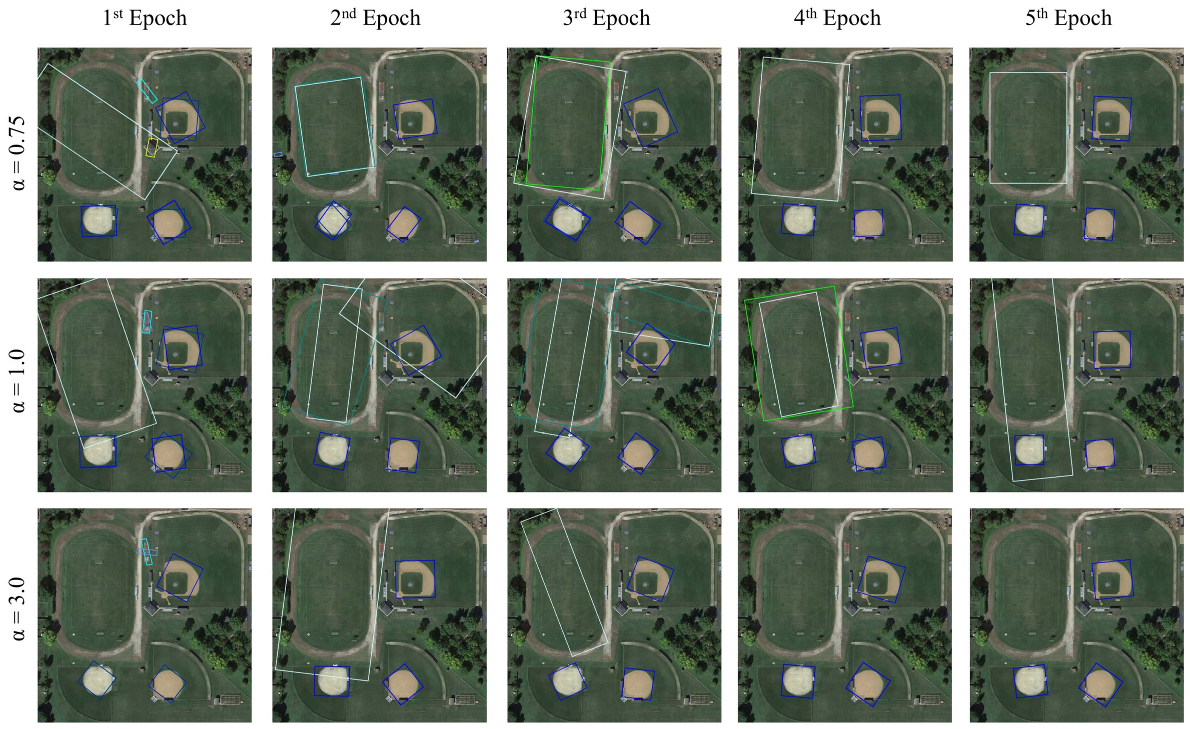


Fig. 8. Comparison of the bounding box regression of the first five epochs of YWCSL with different α values.

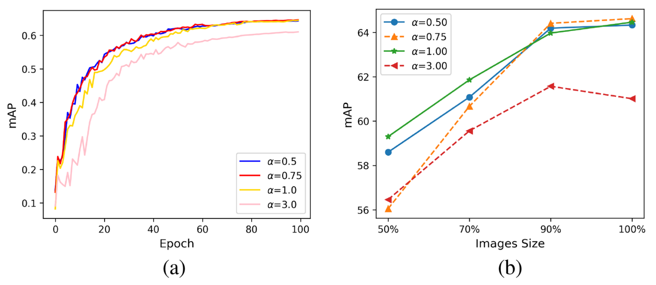


Fig. 9. (a) When the value of α is 0.50, 0.75, 1.00, and 3.00, respectively, YWCSL compares the detection accuracy of each epoch on the DIOR-R test set. (b) The effect of different DIOR-R training set sizes on the performance of the YWCSL model when α takes different values.

TABLE VI
EFFECT OF DIFFERENT TRAINING SET SIZES ON THE TRAINING TIME OF THE YWCSL MODEL

Proportion	Number of images	Size	Images/Second
100%	11725	3.9GB	29.46
90%	10769	3.6GB	29.67
70%	8613	2.9GB	29.80
50%	6213	2.1GB	30.76

model when $\alpha > 1$. The experimental results in the table show that, consistent with the analysis in Section III-A, when $0 < \alpha < 1$, the YWCSL model pays more attention to learning difficult examples with lower IoU. For example, when $0 < \alpha < 1$, the detection accuracy of the model on low mAP categories such as APO, BR, and TS is significantly higher than that of the model when $\alpha >= 1$. The former also performs better on classes with smaller pixel sizes, such as BF, CH, ETS, and WM. Fig. 8 shows the bounding box regression in the first five epochs of YWCSL

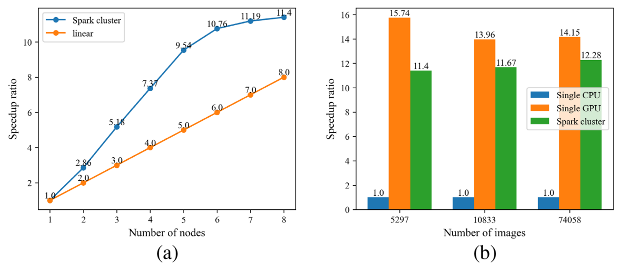


Fig. 10. Effect of the number of nodes and the number of images on cluster performance.

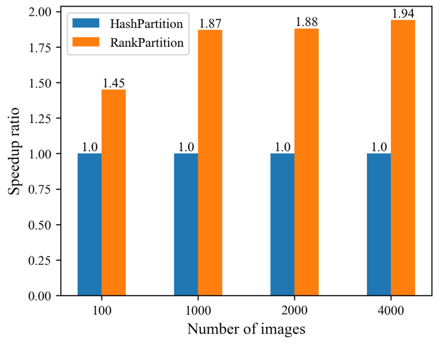


Fig. 11. Comparison of Spark cluster performance when setting different partitioners HashPartition and RankPartition.

training with different α values. It can be seen that more and higher-quality bounding boxes are produced when $\alpha = 0.75$.

b) Effect of Training Set Size on YWCSL Performance: Fig. 9(b) shows the effect of different data amounts on the performance of the YWCSL model. We divided the DIOR-R

TABLE VII
EXPERIMENTAL ENVIRONMENT UNDER DIFFERENT INFERENCE MODES

Inference mode	Hardware environment	Memory	Core
Single CPU	Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz	16GB	8
Single GPU	GeForce RTX 2080Ti GPU & Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz	CPU 128GB&GPU 11GB	64
Spark cluster	Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz	16G×8	8×8

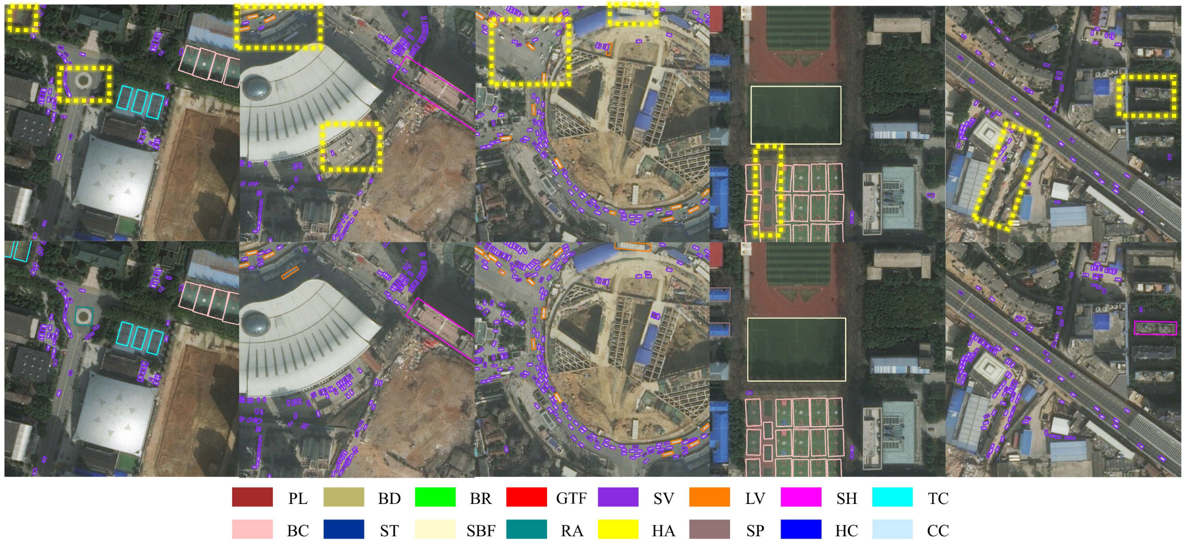


Fig. 12. Detection effect of the YWCSL model in remote sensing images of China Optics Valley Square. For the first row, we use CIoU Loss as the loss function for the bounding box, trained for 100 epochs on the DOTA-v1.5 training set, and for the second row, we use α-CIoU Loss ($\alpha = 0.5$) as the bounding box regression loss function. The same epochs were trained on the DOTA-v1.5 training set. We use the yellow dotted box to mark the areas with different detection results. It can be seen that α-CIoU Loss ($\alpha = 0.5$) can effectively improve the detection effect of the model on small objects.

training set by 50%, 70%, 90%, and 100% in each category, and the test set was not processed in any way. To avoid chance, we trained on each case three times and took the average mAP as the experimental result. The results show that α-CIoU only shows advantages when the data volume of the training set reaches 90% of the original data set and performs poorly when the data volume is small. To sum up, we should set α to around 0.75 when the training data is large and set α to 1.00 when the training data is small. The impact of different data sizes on the training speed of the YWCSL model is shown in Table VI. It can be seen that it takes about 400 s to train an epoch using all the data sets.

E. Spark Cluster Performance Comparison

To reflect the advantages of the distributed inference framework built in this article, as shown in the Table VII, we compared the inference speed in three different modes: single-machine CPU, single-machine GPU, and 8-node Spark cluster mode. The comparison results are shown in Fig. 10. It can be observed that the average speedup ratio (11.785) of the 8-node Spark cluster built in this article is much larger than the linear speedup ratio (8.000). Moreover, it can be seen that as the amount of data or the number of nodes increases, the greater the speedup ratio brought by the Spark cluster (the Spark cluster partitioner below uses RankPartition by default). Compared with the Single GPU mode, the Spark cluster inference framework we established can achieve a detection speed of 80.61% of the Single GPU mode in an environment that only uses the same CPU resources (same memory and CPU cores). Moreover, as the amount of data increases, the speeds of the two different inference modes of Spark cluster and Single GPU are closer.

In addition, to investigate the influence of different node numbers on Spark cluster inference speed, we compared the inference times on the validation set (5297 images, 6.77 GB, image size is 1024×1024) from a traditional Single CPU to eight nodes. As shown in Fig. 10(a), as the number of nodes increases, the Spark cluster inference speed increases, and the actual speedup ratio is far beyond the theoretical linear speedup ratio. However, after the number of nodes reaches 5, the speedup ratio of Spark cluster gradually degenerates to a linear speedup ratio.

F. Practical Application of Distributed Inference Framework

To test the detection effect of our distributed inference framework for object detection in large-scale remote sensing images in practical applications, we used the baseline in [16] and our improved model for deploying to the Spark cluster. The test dataset was intercepted from the 13.53-square-kilometer area near the Optics Valley Square in Wuhan, China, in the ArcGIS map. The image resolution reaches 17520×11712, and the size is 211 MB. As in Section IV-B, we first scale the original image by three scaling factors (0.5, 1.0, 1.5) and then cut the image into slices of 1024×1024 with a stride of 512. After cutting, the number of images is 2669, and the data size is 3.3 GB.

As shown in Fig. 12, when α takes the value of 0.5, the YWCSL model outperforms the baseline on small objects such as SV and SH. In addition, we compared the Single CPU inference mode and the Spark cluster inference mode. The former took 2883.98 s, while the 8-node Spark cluster we built took only 268.53 s, which improved the inference speed by 9.74 times.

V. Conclusion

This article proposes a distributed inference framework for large-scale remote sensing images that can achieve rapid object detection in large-scale remote sensing images using relatively cheap CPU cluster resources. Notably, the actual speedup ratio of our established distributed inference framework for object detection in large-scale remote sensing images far exceeds the theoretical linear speedup ratio. As the number of data increases or the number of nodes increases, the actual speedup ratio increases (see Fig. 10). When using only eight nodes, the speedup ratio can reach 12.28 under the data volume of 84.97 GB in this experiment, which is 53.5% higher than the linear speedup ratio of 8.00. It has crucial research significance for the practical application of object detection in large-scale remote sensing images.

For the balance between model inference speed and accuracy, we further improve the accuracy and inference speed of the model based on [16]. Our improved model achieves 64.62%, 79.54%, and 76.28% mAP on the DIOR-R, DOTA-v1.0, DOTA-v1.5 test sets, respectively, which is better than the current state-of-the-art one-stage detector S^2A-Net performs better. The distributed inference framework for object detection in large-scale remote sensing images established in this article can also support distributed GPU inference. However, we did not conduct experiments on GPU clusters due to the lack of sufficient GPU cluster machines. In addition, the distributed inference framework for object detection in large-scale remote sensing images that we have established supports the flexible switching of models. Users can deploy their models directly on distributed clusters without paying attention to the underlying distributed inference details.

There are still some shortcomings in our improvement work. As analyzed in Section IV-D, the YWCSL model performs poorly when the training data is insufficient after the introduction of α-IoU. More importantly, the detection performance of the YWCSL model is more sensitive to the value of α taken. In general, if a dataset contains many difficult examples, α can be set between 0.50 and 1.00. Otherwise, α can be set between 1.00 and 3.00. According to the analysis in this article, the model learns better on the bounding boxes with low IoU when $0 < \alpha < 1$, and performs better on the bounding boxes with high IoU when $\alpha > 1$. In the early stage of training, most of the bounding boxes have low IoU, so the former performs better in the early stage of training, while in the later stage of training, a proper boost of α will help the model learn better. Our subsequent work will investigate the adaptive α-IoU loss to ensure that the YWCSL model can be trained more stably and efficiently.

References

[1] G.-S. Xia et al., "DOTA: A large-scale dataset for object detection in aerial images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3974–3983.

[2] G. Cheng et al., "Anchor-free oriented proposal generator for object detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5625411.

[3] Z. Liu, L. Yuan, L. Weng, and Y. Yang, "A high resolution optical satellite image dataset for ship recognition and some new baselines," in *Proc. Int. Conf. Pattern Recognit. Appl. Methods*, 2017, vol. 2, pp. 324–331.

[4] K. Makantasis, A. Voulodimos, A. Doulamis, N. Doulamis, and I. Georgoulas, "Hyperspectral image classification with tensor-based rank-r learning models," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 3148–3125.

[5] K. Makantasis, A. D. Doulamis, N. D. Doulamis, and A. Nikitakis, "Tensor-based classification models for hyperspectral data analysis," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 12, pp. 6884–6898, Dec. 2018.

[6] K. Makantasis, K. Karantzalos, A. D. Doulamis, and K. Loupos, "Deep learning-based man-made object detection from hyperspectral data," in *Proc. Int. Symp. Vis. Comput.*, 2015, pp. 717–727.

[7] J. Ding et al., "Object detection in aerial images: A large-scale benchmark and challenges," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2021.3117983.

[8] M. Kaselimi, A. Voulodimos, I. Daskalopoulos, N. Doulamis, and A. Doulamis, "A vision transformer model for convolution-free multilabel classification of satellite imagery in deforestation monitoring," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2022.3144791.

[9] D. Yi, J. Su, and W. Chen, "Probabilistic faster R-CNN with stochastic region proposing: Towards object detection and recognition in remote sensing imagery," *Neurocomputing*, vol. 459, pp. 290–301, 2021.

[10] J. Yan, L. Wang, H. He, D. Liang, W. Song, and W. Han, "Large-area land-cover changes monitoring with time-series remote sensing images using transferable deep models," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 4409917.

[11] W. Han et al., "Sample generation based on a supervised Wasserstein generative adversarial network for high-resolution remote-sensing scene classification," *Inf. Sci.*, vol. 539, pp. 177–194, 2020.

[12] 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.*, 2017, pp. 2980–2988.

[13] K. Chen et al., "Hybrid task cascade for instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4974–4983.

[14] J. Ding, N. Xue, Y. Long, G.-S. Xia, and Q. Lu, "Learning RoI transformer for oriented object detection in aerial images," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2849–2858.

[15] X. Yang and J. Yan, "Arbitrary-oriented object detection with circular smooth label," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 677–694.

[16] hukaixuan19970627, "Yolov5 obb," 2022. [Online]. Available: https://github.com/hukaixuan19970627/yolov5_obb

[17] J. Han, J. Ding, N. Xue, and G.-S. Xia, "Redet: A rotation-equivariant detector for aerial object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2786–2795.

[18] J. He, S. M. Erfani, X. Ma, J. Bailey, Y. Chi, and X. Hua, "Alpha-IoU: A family of power intersection over union losses for bounding box regression," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 20230–20242.

[19] J. Yan et al., "Land-cover classification with time-series remote sensing images by complete extraction of multiscale timing dependence," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 1953–1967, 2022.

[20] 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.

[21] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[22] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 734–750.

[23] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 850–859.

[24] Y. Jiang et al., "R2CNN: Rotational region CNN for orientation robust scene text detection," 2017, *arXiv:1706.09579*.

[25] J. Ma et al., "Arbitrary-oriented scene text detection via rotation proposals," *IEEE Trans. Multimedia*, vol. 20, no. 11, pp. 3111–3122, Nov. 2018.

[26] X. Yang et al., "SCRDET: Towards more robust detection for small, cluttered and rotated objects," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 8232–8241.
[27] X. Yang, L. Hou, Y. Zhou, W. Wang, and J. Yan, "Dense label encoding for boundary discontinuity free rotation detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15819–15829.
[28] J. Han, J. Ding, J. Li, and G.-S. Xia, "Align deep features for oriented object detection," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2021, Art. no. 5602511.
[29] Q. Wen, R. Liu, X. Wu, B. Ao, and K. Li, "Aerial image object detection based on improved YOLOv5," in *Proc. 2nd Int. Conf. Consum. Electron. Comput. Eng.*, 2022, pp. 561–564.
[30] F. Wang, X. Wang, W. Cui, X. Xiao, and J. Li, "Distributed retrieval for massive remote sensing image metadata on spark," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2016, pp. 5909–5912.
[31] M. M. Rathore, A. Ahmad, A. Paul, and A. Daniel, "Hadoop based real-time Big Data architecture for remote sensing earth observatory system," in *Proc. 6th Int. Conf. Comput., Netw. Technol.*, 2015, pp. 1–7.
[32] H. Liu, Y. L. Chen, N. Cui, D. Z. Xu, and J. Li, "An effective data fusion model for detecting the risk of transmission line in smart grid," *IEEE Internet Things J.*, to be published, doi: 10.1109/JIOT.2021.3100522.
[33] X. Song, J. Li, Q. Lei, W. Zhao, Y. Chen, and A. Mian, "Bi-CLKT: Bi-graph contrastive learning based knowledge tracing," *Knowl.-Based Syst.*, vol. 241, 2022, Art. no. 108274.
[34] J. Dean et al., "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1223–1231.
[35] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," 2015, *arXiv:1501.02876*.
[36] P. Moritz, R. Nishihara, I. Stoica, and M. I. Jordan, "SparkNet: Training deep networks in spark," 2015, *arXiv:1511.06051*.
[37] A. Apoorva, G. K. Mishra, R. R. Sahoo, S. K. Bhoi, and C. Mallick, "Deep learning-based ship detection in remote sensing imagery using Tensor-Flow," in *Proc. Adv. Mach. Learn. Comput. Intell.*, 2021, pp. 165–177.
[38] J. J. Dai et al., "BIGDL: A distributed deep learning framework for Big Data," in *Proc. ACM Symp. Cloud Comput.*, 2019, pp. 50–60. [Online]. Available: https://arxiv.org/pdf/1804.05839.pdf
[39] Yahoo, "Caffeonspark," 2016. [Online]. Available: https://github.com/yahoo/CaffeOnSpark
[40] J. Lin and D. Ryaboy, "Scaling Big Data mining infrastructure: The Twitter experience," *ACM SIGKDD Explorations Newslett.*, vol. 14, no. 2, pp. 6–19, 2013.
[41] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 658–666.
[42] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 12993–13000.
[43] D. Nguyen and A. L. Hong, "A Big Data framework for satellite images processing using apache hadoop and rasterframes: A case study of surface water extraction in PHU tho, Viet Nam," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 12, 2020, Art. no. 0111289.
[44] J. Weipeng, Dongxue, and Tian, "An improved distributed storage and query for remote sensing data," *Procedia Comput. Sci.*, vol. 129, pp. 238–247, 2018.
[45] 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. Photogrammetry Remote Sens.*, vol. 159, pp. 296–307, 2020.
[46] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.
[47] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," 2017, *arXiv:1710.09412*.
[48] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[49] T.-Y. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.
[50] Y. Xu et al., "Gliding vertex on the horizontal bounding box for multi-oriented object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 4, pp. 1452–1459, Apr. 2021.
[51] G. Cheng et al., "Dual-aligned oriented detector," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, 2022, Art. no. 5618111.
[52] X. Pan et al., "Dynamic refinement network for oriented and densely packed object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11204–11213.
[53] W. Qian, X. Yang, S. Peng, J. Yan, and Y. Guo, "Learning modulated loss for rotated object detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 3, pp. 2458–2466, 2021.
[54] Q. Ming, Z. Zhou, L. Miao, H. Zhang, and L. Li, "Dynamic anchor learning for arbitrary-oriented object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 2355–2363.
[55] X. Yang, Q. Liu, J. Yan, and A. Li, "R3Det: Refined single-stage detector with feature refinement for rotating object," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 3163–3171.
[56] J. Wang, W. Yang, H.-C. Li, H. Zhang, and G.-S. Xia, "Learning center probability map for detecting objects in aerial images," *IEEE Trans. Geosci. Remote Sens.*, vol. 59, no. 5, pp. 4307–4323, May 2021.
[57] C. Li et al., "Learning object-wise semantic representation for detection in remote sensing imagery," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 20–27.
[58] Q. Song, F. Yang, L. Yang, C. Liu, M. Hu, and L. Xia, "Learning point-guided localization for detection in remote sensing images," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 14, pp. 1084–1094, 2020.
[59] D. Yu et al., "Anchor-free arbitrary-oriented object detector using box boundary-aware vectors," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 15, pp. 2535–2545, 2022.
[60] X. Xie, G. Cheng, J. Wang, X. Yao, and J. Han, "Oriented R-CNN for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 3500–3509.
[61] Y. Zhu, J. Du, and X. Wu, "Adaptive period embedding for representing oriented objects in aerial images," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 10, pp. 7247–7257, Oct. 2020.

Linkai Liu is currently working toward the B.E. degree in data science and big data technology with the China University of Geosciences, Wuhan, China.

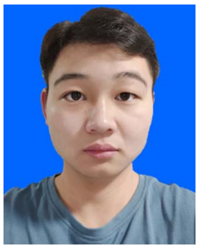
His research interests include object detection in large-scale remote sensing images, Big Data technology, and machine learning.

Yuanxing Liu received the B.E. degree from China University of Geosciences, Wuhan, China, in 1997. He is currently working toward the Ph.D. degree in geoscience information engineering with the School of Computer Science, China University of Geosciences, Wuhan, China.

His research interests include Big Data management and visualization.

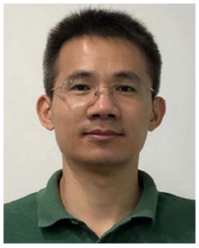
Jining Yan (Member, IEEE) received the Ph.D. degree in signal and information processing from the University of Chinese Academy of Sciences, Beijing, China, in 2017.

He is an Associate Professor with the School of Computer Science, China University of Geosciences, Wuhan, China. His research is focused on remote sensing data management and time-series analysis.



Hong Liu received the B.E. degree in software engineering from North China University of Water Resources and Electric Power, Zhengzhou, China, in 2020. He is currently working toward the M.E. degree in electronic information from China University of Geosciences, Wuhan, China.

His research is focused on time-series remote sensing data organization and management.



Mingming Li received the M.E. degree in cartography and geographic information system from the Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences (CAS), Urumqi, China, in 2010.

He is currently a registered surveyor and a deputy general Manager of Soarscape Technology Development, (Shanghai) Company, Ltd. His research interests include renconstruction and application of real 3-D modeling technology.

Jinlin Wang received the Ph.D. degree in cartography and geographic information system from the Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi, China, in 2010.

He is currently a Senior Engineer with the Xinjiang Key Laboratory of Mineral Resources and Digital Geology, the Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi, China. His research interests include remote sensing geology, mathematical geology, hyperspectral application in geology and mineral resources, and GIS technology and application.

Kefa Zhou received the Ph.D. degree in cartography and geographic information system from the Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi, China, in 2008.

He is currently a Professor Member with the Xinjiang Key Laboratory of Mineral Resources and Digital Geology, the Xinjiang Institute of Ecology and Geography, Chinese Academy of Sciences, Urumqi, China. His research interests include mathematical geology, remote sensing applications, Big Data, and metallogenic prediction.