An Improved Aggregated-Mosaic Method for the Sparse Object Detection of Remote Sensing Imagery

Boya Zhao 1,*, Yuanfeng Wu 1,2,†, Xinran Guan 1,2, Lianru Gao 1,3 and Bing Zhang 1,3

1 Key Laboratory of Digital Earth Science, Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100094, China; zhaoby@aircas.ac.cn (B.Z.); guanxinran20@mails.ucas.ac.cn (X.G.); gaolr@aircas.ac.cn (L.G.); zb@radi.ac.cn (B.Z.)
2 School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100049, China
3 College of Resources and Environment, University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: wuyf@radi.ac.cn

Abstract: Object detection based on remote sensing imagery has become increasingly popular over the past few years. Unlike natural images taken by humans or surveillance cameras, the scale of remote sensing images is large, which requires the training and inference procedure to be on a cutting image. However, objects appearing in remote sensing imagery are often sparsely distributed and the labels for each class are imbalanced. This results in unstable training and inference. In this paper, we analyze the training characteristics of the remote sensing images and propose the fusion of the aggregated-mosaic training method, with the assigned-stitch augmentation and auto-target-duplication. In particular, based on the ground truth and mosaic image size, the assigned-stitch augmentation enhances each training sample with an appropriate account of objects, facilitating the smooth training procedure. Hard to detect objects, or those in classes with rare samples, are randomly selected and duplicated by the auto-target-duplication, which solves the sample imbalance or classes with insufficient results. Thus, the training process is able to focus on weak classes. We employ VEDAI and NWPU VHR-10, remote sensing datasets with sparse objects, to verify the proposed method. The YOLOv5 adopts the Mosaic as the augmentation method and is one of state-of-the-art detectors, so we choose Mosaic (YOLOv5) as the baseline. Results demonstrate that our method outperforms Mosaic (YOLOv5) by 2.72% and 5.44% on 512 × 512 and 1024 × 1024 resolution imagery, respectively. Moreover, the proposed method outperforms Mosaic (YOLOv5) by 5.48% under the NWPU VHR-10 dataset.

Keywords: remote sensing; object detection; data augmentation; sparse distribution; YOLOv5

1. Introduction

Object detection is a key component of remote sensing research [1–3] and plays a crucial role in the fields of urban planning, environment monitoring, and borderland patrol [4–6]. Studies on object detection-based methods have made great progresses in recent years [7,8], particularly for deep learning applications [9,10]. To further improve the training performance and efficiency, numerous training strategies have been proposed, including loss function design, data augmentation, and regularization techniques. In order to prevent overfitting or the excessive focus on a small set of input training objects, random feature removal (e.g., dropout) has been developed. The dropout randomly deletes neural nodes or erases some regions. While the regional dropout method has improved the generalization of object detection tasks, the training of convolutional neural networks is a data-driven process, and current object detection methods are limited by a lack of training data. In the field of object detection, basic augmentation methods are typically combined with cropping, rotation, translation, and color jittering, and semantic augmentation meth-

Remote Sens. 2021, 13, 2602. https://doi.org/10.3390/rs13132602
ods are composed of background and shape changes (Figure 1). The basic augmentation only augments the objects and neglect the significance of multiple backgrounds.

Figure 1. Basic training data augmentation methods.

In order to efficiently use the training samples, multiple images can be adopted into a single training image via approaches that mix training samples (e.g., CutMix [11], MixUp [12], and Mosaic [13]). However, objects in remote sensing images are typically sparsely distributed, and those sample fusion methods will result in blank training samples (Figure 2). Moreover, the class labels in the remote sensing images are often imbalanced, and thus these objects appear in few images or scenes. The synthetic sample and re-sampling methods proposed to overcome these problems are pre-processing procedures, and are not able to balance the training samples automatically and dynamically.

Figure 2. Mosaic augmentation generates training samples by randomly selecting four images and splicing them together.

The aim of the current paper is to design mosaic training methods for remote sensing images with a sparse object distribution. We address this by introducing the Aggregated-
Mosaic Training augmentation strategy. Based on the state-of-the-art detector Yolov5 [13] and the augmentation method Mosaic, our contributions are summarized as follows:

1. We analyze the object distribution of remote sensing images and highlight the problem of augmentation in object samples with a sparse distribution. We introduce Assigned-Stitch, a constraint for sparse object augmentation, which guarantees that each training sample is not a blank sample and ensures a smooth and stable training procedure;

2. We propose the Auto-Target-Duplication modules to overcome the class-imbalance problem. This module is able to balance the labels of each class and focus on the poor class results by efficiently selecting objects to duplicate.

The rest of the paper is organized as follows. In Section 2, related work, such as object detection and data augmentation, is reviewed, while the proposed Aggregated-Mosaic Training method is detailed in Section 3. In Section 4, we present two datasets and describe the experimental results. Section 5 concludes the paper and outlines future research directions.

2. Related Work

2.1. Object Detection

Over the last decades, the object detection has developed quickly, and this development composes of two stages. First, the hand-crafted features [7,14] and discriminative classifiers [15,16] are combined for object detection, such as in a Deformable Part Model (DPM) [8,17]. Second, as the hand-crafted features can make it hard to distinguish between objects and backgrounds, the convolution features come to the fore, along with the quick development of the computing power. The convolution neural-network-based classification method was first proposed in 2012 at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2012) [18]. More specifically, Krizhevsky et al. proposed AlexNet [19], based on convolutional layers, fully connected (FC) layers, and a simple softmax classifier, which transfers the convolutional neural networks (CNNs) into the computer vision tasks. There are currently two object detection framework classes.

One-stage object detection: Joseph et al. proposed You Only Look at Once (YOLO) [20], which performs the detection procedure as a regression task. Liu Wei et al. subsequently proposed a Single Shot multibox Detector (SSD) [21], which combines the anchor strategy in Region with Convolutional Neural Networks (R-CNN) [22] and the regression in YOLO to increase accuracy and detection speed. Following this, Yolov2 [23] was proposed, which includes multi-scale training and hierarchical classification, resulting in both a fast detection speed and high accuracy. Yolov3 [24,25] adopts Feature Pyramid Network (FPN) [26], darknet-53, and binary cross entropy loss to balance the model size and detection speed, resulting in a 100x greater detection speed compared with fast R-CNN [27]. Yolov4 [28,29] combines multiple optimization methods (e.g., mosaic data augmentation, self-adversarial-training, mish-activation, and CIoU [30,31] loss) to attain a higher detection accuracy. Based on its predecessors, the state-of-the-art detector Yolov5 [13] adopts a hyper-parameter to balance the model size and detection speed. In the current paper, we adopt Yolov5 as the baseline.

Two-stage object detection: Ross et al. proposed the first CNN-based object detection framework, denoted as R-CNN [22]. CNN features have been proven to demonstrate a greater discrimination between objects and backgrounds compared to hand-craft features. Fast R-CNN [27] and Faster R-CNN [32] were subsequently proposed, accelerating the detection within a region. Mask R-CNN [10] employs multi-task training for object detection and instance segmentation. By accounting for the large amount of computation in the localization procedure, the CenterNet [33] regards the object as a center point and regresses the bounding box. Following Mask R-CNN, D2Det [9] adopts dense local regression and discriminative RoI pooling to enhance the detection and instance segmentation performance.
2.2. Data Augmentation

Random dropout: The dropout concept can be divided into two categories: (i) In the neural network dropout, dropout series methods [34,35] randomly mask a set or region of neural units for robust hyper-parameters. (ii) The training image dropout approach employs Random Erase (RE) [36] to solve overfitting during training and object conclusion by randomly selecting a rectangle region in the training samples and erasing pixels with random values. Following RE, CutOut [37,38] randomly masks out square regions of input training samples, improving the robustness and performance of classification CNNs. The Assigned-Stitch module proposed in this paper adopts random cut and translation, which also involves the random dropout concept.

Training data generation: In order to overcome the lack of training samples and to improve the neural network generalization, much research has explored data generation techniques. Cut Paste Learn (CPL) [39] cuts object instances and pastes them on random backgrounds, which relieves expensive data collection costs. Then, Dvornik et al. [40] improves on this and leverage segmentation annotations by increasing the number of object instances presented in the training data. Geirhos et al. uses stylized-imagenet [41], a stylized version of imagenet, to learn a texture-based representation for shape-based feature representation. The proposed Aggregated-Mosaic is able to generate numerous remote sensing training samples. In particular, it ensures that each training sample contains multiple objects with sparse distribution scenes and, based on the evaluation results, Auto-Target-Duplication automatically pastes classes associated with poor results.

Multiple sample fusion: Multiple sample fusion is currently the focus of much attention due to its effective use of the training dataset. Mixup [12] trains a convolutional neural network on convex combinations of image pairs and labels. Between-Class (BC) [42] mixes two images belonging to different classes with a random ratio. This technique can also work on sounds. Several descendant mixing methods have been proposed based on these fusion techniques. Random Image Cropping and Patching (RICAP) [43] randomly crops four images and patches the images and labels to create a new training sample. Then, CutMix [11] cuts and pastes patches among training sets, and the ground truth labels are also mixed proportionally to the area of the patches. Note that CutMix mixes just two input images, while Yolov4 proposes the Mosaic method [28], which mixes four training images and allows the detection of objects outside their normal context. These multiple sample fusion methods are powerful when applied to natural images. However, the objects in remote sensing images are often distributed sparsely, and thus these multiple sample fusion methods typically generate training samples with no objects, leading to unstable training and inference procedures. With a focus on remote sensing imagery with sparse objects, the proposed Aggregated-Mosaic employs training sample fusion, and also solves the problem of blank training sample.

Generalization tricks: Due to the requirement of a large amount of training data, ensuring the efficient training of a convolutional neural network is crucial in object detection and classification. Batch Normalization (BN) [44], Adam [45], and weight decay [46] are widely employed in training procedures. In order to combat adversarial training samples [47], several approaches include the addition of noise to feature maps, and manually set a network shortcut to enhance the performance [48]. Then, Multi-Objective Genetic Algorithm (MOGA) [49], Selective Sparse Sampling Networks (S3Ns) [50], and virtual image generation [51] are proposed for the sparse object representation. Distinct from the aforementioned generalization tricks, the proposed Aggregated-Mosaic enhances the performance on the data level, without altering the architecture and feature representations. This allows its application in methods involving remote sensing images with a sparse object distribution.

3. Methods

In this section, the problem of sparsely distributed object training is explained in detail. Then, the Aggregated-Mosaic method, which contains Assigned-Stitch and Auto-
Target-Duplication, is proposed. The Assigned-Stitch focuses on the training procedure for the sparsely distributed object and the Auto-Target-Duplication solves the problem of class-imbalance.

3.1. Sparse Object in Remote Sensing Images

Object detection on remote sensing images has attracted much attention over the past few years. In particular, satellite imagery contains a large amount of information per pixel, and thus the geographic extent per image is extensive. Thus, the training samples are often clipped with a certain user-defined bin size and overlapped by a sliding window [52] (Figure 3).

Remote sensing images often contain sparsely distributed objects that appear at low frequencies, particularly objects in deserted areas. Multiple sample fusion methods, such as Mosaic augmentation, may generate blank training samples (Figure 2). In the current paper, we reviewed an extensive number of satellite images to define the sparse object distribution as the objects cluster on the four corners or the four lines for the sliding images (Figure 4).

With more blank training samples generated, the instability of the number of objects in the training sample seriously disturbs the training process, resulting in an unsatisfactory performance. Figure 5 presents a training procedure and the number of objects in each training sample of the VEDAI dataset, a real remote sensing dataset with sparse object distributions. The number of objects vary greatly between training samples, leading to drastic classification, location losses, a slow convergence, and, ultimately, a poor performance.
With more blank training samples generated, the instability of the number of objects in the training sample seriously disturbs the training process, resulting in an unsatisfactory performance. Figure 5 presents a training procedure and the number of objects in each training sample of the VEDAI dataset, a real remote sensing dataset with sparse object distributions. The number of objects vary greatly between training samples, leading to drastic classification, location losses, a slow convergence, and, ultimately, a poor performance.

3.2. Aggregated-Mosaic

The proposed Aggregated-Mosaic is centered on the data augmentation procedure and contains two modules, Assigned-Stitch and Auto-Target-Duplication. Figure 6 depicts the implementation of the proposed algorithm. The first augmentation step, Assigned-
Stitch, focuses on sparse objects. The Assigned-Stitch module requires that each training sample contain at least one object for a smooth training procedure. Auto-Target-Duplication focuses on classes that have generated poor results. Various copies of such classes are pasted on the training sample generated from the Assigned-Stitch module. This duplication procedure is not only a copy-paste strategy, but also a feedback strategy that can solve the sample imbalance and overfitting problems.

3.2. Aggregated-Mosaic

The proposed Aggregated-Mosaic is centered on the data augmentation procedure and contains two modules, Assigned-Stitch and Auto-Target-Duplication. Figure 6 depicts the implementation of the proposed algorithm. The first augmentation step, Assigned-Stitch, focuses on sparse objects. The Assigned-Stitch module requires that each training sample contain at least one object for a smooth training procedure. Auto-Target-Duplication focuses on classes that have generated poor results. Various copies of such classes are pasted on the training sample generated from the Assigned-Stitch module. This duplication procedure is not only a copy-paste strategy, but also a feedback strategy that can solve the sample imbalance and overfitting problems.

3.3. Assigned-Stitch

Considering that the objects in remote sensing images are often sparsely distributed, the traditional training sample generation methods will randomly generate blank samples, which only consist of backgrounds and make the training procedure unstable. The Assigned-Stitch module constrains the objects in the sample by calculating the object locations and centers of the training sample slice. Figure 7 presents the Assigned-Stitch process flow. First, the dataloader module randomly selects four 640 × 640 samples (A1, B1, C1, and D1) for augmentation. These four samples are merged as a 1280 × 1280 image. Following the application of Mosaic, a 1280 × 1280 grey image is initialized, and the merged sample (with center \((x_{sc}, y_{sc})\)) is randomly placed on this grey image. Third, the Assigned-Stitch guarantees the distribution of the objects on the top-left of the fusion image, denoted as the Assigned-Object sample. Last, the affine transformation is applied on the Assigned-Object sample, and the top-left 640 × 640 section is sliced as the Assigned-Stitch training sample.

In order to explain the Assigned-Stitch in detail, we select image A1 as an example. Figure 8 depicts the detailed Assigned-Stitch for A1.
In order to explain the Assigned-Stitch in detail, we select image A1 as an example. Figure 8 depicts the detailed Assigned-Stitch for A1.

In Figure 8, the purple rectangle denotes the location of the A1 sample, with center \((x_{\text{patch}_c}, y_{\text{patch}_c})\). Following this, an object in A1 is randomly selected (green rectangle) with center \((x_{\text{object}_c}, y_{\text{object}_c})\). Note that the green rectangle has the same area as the...
Algorithm 1. Assigned-Stitch

\textbf{Input}: randomly select four images (A1, B1, C1, and D1) with $w \times h$ resolution; random object ground truth $(g_1^s, g_2^s, g_3^s, g_4^s)$.  
\textbf{Output}: Assigned-Stitch training sample $I_{fu}$.

1. Initialize grey image $I_{grey}$, which value is 127 and Randomly select the slice center $(x_{sc}, y_{sc})$.
2. Purple rectangles for $I_{grey}$:
   - Top-left: $[x_{pmin1}, y_{pmin1}, x_{pmax1}, y_{pmax1}] = [\max(x_{sc} - w, 0), \max(y_{sc} - h, 0), x_{sc}, y_{sc}]$
   - Top-right: $[x_{pmin2}, y_{pmin2}, x_{pmax2}, y_{pmax2}] = [\max(x_{sc} + w, w \times 2), y_{sc}]$
   - Bottom-left: $[x_{pmin3}, y_{pmin3}, x_{pmax3}, y_{pmax3}] = [\max(x_{sc} - w, 0), y_{sc}, x_{sc}, \min(y_{sc} + h, h \times 2)]$
   - Bottom-right: $[x_{pmin4}, y_{pmin4}, x_{pmax4}, y_{pmax4}] = [x_{sc}, y_{sc}, \min(x_{sc} + w, w \times 2), \min(y_{sc} + h, h \times 2)]$
3. Four input images slices (green rectangles) for each purple rectangle $i$:
   $I_{x_{pmin}y_{pmin}}x_{pmax}y_{pmax} = \text{toRect}(I_{grey}, g_i, g_{i1}^s, (x_{pmax} - x_{pmin}), (y_{pmax} - y_{pmin}))$
4. Verify each image slices:
   - \(i_f = \begin{cases} 0, & \text{if } x_{pmin} < 0, y_{pmin} \\ 1, & \text{otherwise} \end{cases}\)
   - \(i_f = \begin{cases} 0, & \text{if } y_{pmin} < 0, y_{pmax} \\ 1, & \text{otherwise} \end{cases}\)
   - \(i_f = \begin{cases} 0, & \text{if } x_{pmax} \geq w, y_{pmax} \\ 1, & \text{otherwise} \end{cases}\)
   - \(i_f = \begin{cases} 0, & \text{if } y_{pmax} \geq h, y_{pmin} \\ 1, & \text{otherwise} \end{cases}\)
5. Place the image slices on $I_{grey}$:
   $I_{grey}(x_{pmin}y_{pmin}x_{pmax}y_{pmax}) = (x_{pmin}y_{pmin}x_{pmax}y_{pmax})$ in $I_{grey}$, $i \in (1, 2, 3, 4)$
6. Perform affine transformation:
   - Repeat $g_i = \text{affine}(g_i, \text{AffineMatrix})$ until the top-left $w \times h$ image contains objects.
7. Assigned-Stitch training sample:
   $I_{fu} = \text{affine}(I_{grey}, \text{AffineMatrix}) (0:w, 0:h)$

In Algorithm 1, \((x_{pmin}y_{pmin}x_{pmax}y_{pmax})\) denote the bounding box of the four corners $i \in (1, 2, 3, 4)$ in grey image $I_{grey}$, \((x_{gi}y_{gi}x_{gi}y_{gi})\), and \((x_{pmin}y_{pmin}x_{pmax}y_{pmax})\) are the center and rectangle forms of the image slices. The \text{toRect} transform the center form to rectangle form and the function is as below:

\[
\begin{align*}
  x_{min} &= x_c - 0.5 \times w_c \\
  y_{min} &= y_c - 0.5 \times h_c \\
  x_{max} &= x_c + 0.5 \times w_c \\
  y_{max} &= y_c + 0.5 \times h_c
\end{align*}
\]

(1)

where the $(x_c, y_c, w_c, h_c)$ and $(x_{min}, y_{min}, x_{max}, y_{max})$ are center and rectangle forms, respectively. Thus, we ensure that the training samples of remote sensing images with sparse object distributions will contain objects.

3.4. Auto-Target-Duplication

The sparse object distribution generally results in a class imbalance and hence complicates the detection results within the same level. The Auto-Target-Duplication module copies the objects of unsatisfactory class and thus forces the network pay more attention on this class. Figure 9 presents the Auto-Target-Duplication pipeline of the proposed Aggregated-Mosaic method.

Following “Copy-Paste” [53], a certain class with unsatisfactory results will be chosen to duplicate. In order to make the detection network pay more attention to the class with poor results, the validation dataset is evaluated after the training of each epoch, and the class with a 0.2 mean Average Precision (mAP) lower than the best class with highest mAP is selected as the unsatisfactory class. Objects of this class are randomly selected from the training set and pasted onto the Assigned-Stitch samples. This flow is a feedback procedure.
and forces the object detection network to pay more attention to the unsatisfactory class. Algorithm 2 details the Auto-Target-Duplication procedure.

![Object Detection Network](image)

**Algorithm 2. Auto-Target-Duplication**

| Input: detection network \( \text{net}() \), training samples and ground truth for this epoch \((\text{img}, \text{label})\), results of validation dataset from last training model \( \text{result} \). |
| Output: Auto-Target-Duplication samples for this epoch \((\text{img}_A, \text{label}_A)\). |

Repeat epochs:

1. Generate the Assigned-Stitch samples: \((\text{img}_A, \text{label}_A) = \text{AS}(\text{img}, \text{label})\)
2. Sort results and identify class with lowest result: \( \text{class}_l = \min(\text{result}) \)
3. Randomly select multiple objects for \( \text{class}_l \) from training dataset:
   \((\text{obj}_i, \text{label}_i), i = 1, 2, 3 \ldots\)\)
4. Duplicate \((\text{obj}_i, \text{label}_i)\) to each Assigned-Stitch samples:
   \((\text{img}_A, \text{label}_A) = \text{Duplicate}(\text{img}_A, \text{label}_A, \text{obj}_i, \text{label}_i), i = 1, 2, 3 \ldots\)\)
5. Use Assigned-Stitch samples to train the detection network in each iteration:
   \( \text{net} = \text{train}(\text{net}(), \text{img}_A, \text{label}_A) \)

Testing the validation dataset from the last training model determines the class with lowest result and the \((\text{obj}_i, \text{label}_i)\) of this class is duplicated on the Assigned-Stitch samples. The detection network is trained on the Auto-Target-Duplication samples. In contrast to traditional copy-paste methods, the Auto-Target-Duplication is a feedback procedure and could make the detection network pay more attention to the class with the lowest result.

![Figure 9. Auto-Target-Duplication module.](image)
4. Experiment

In this section, we choose VEADI and NWPU VHR-10 datasets, which are two real remote sensing datasets and contain the objects with sparse distribution, to evaluate the proposed method. Then, the training settings and discussions are explained in detail.

4.1. Datasets

4.1.1. VEADI

VEADI [54] contains two subsets: (i) VEADI_512 includes 1268 RGB tiles (512 × 512) and the associated infrared (IR) images at a 0.25 m spatial resolution; and (ii) VEADI_1024 compromises 1268 RGB tiles (1024 × 1024) and the associated infrared (IR) images at a 0.125 m spatial resolution. For each tile, annotations with nine classes (Car, Truck, Ship, Tractor, Camping Car, Van, Vehicle, Pick-Up, and Plane) are provided with bounding boxes in the image. We choose this dataset for its sparse object distribution. The images of VEADI_1024 dataset are sliced to a size of 640 × 640 with 256 strides and images with no objects are deleted. Figure 10 depicts the number of objects in VEADI and the XY distribution for the Plane class and its sparse distribution.

![Figure 10. XY distribution for the Plane class in the VEDAI dataset.](image)

4.1.2. NWPU VHR-10

The NWPU VHR-10 [55] dataset contains 10 geospatial object classes including airplane (PL), baseball diamond (BD), basketball court (BC), bridge, harbor (BH), ground track field (GT), ship (SH), storage tank (ST), tennis court (TC), and vehicle (VH). It consists of 715 RGB images and 85 pan-sharpened color infrared images. The 715 RGB images are collected from Google Earth and their spatial resolutions vary from 0.5 m to 2 m. The 85 pan-sharpened infrared images, with a spatial resolution of 0.08 m, are obtained from Vaihingen data. This dataset contains a total of 3775 object instances that are manually annotated with horizontal bounding boxes. The training and testing sets are randomly selected, and contain 520 and 130 images, respectively. As with VEADI_1024, the images are sliced to a size of 640 × 640 with 256 strides and images with no objects are deleted. Figure 11 presents the number of objects in VHR-10 and the XY distribution for the Ground Track Field class of the dataset and the distribution of it is sparse obviously. Based on the explanation in Section 3.1, multiple sample fusion methods, such as Mosaic augmentation, may generate blank training samples, and the instability of the number of objects in the training sample seriously disturbs the training process, resulting in an unsatisfactory performance.
where \( E \) is the objective function.

### 4.2. Training Settings

Figure 11. XY distribution for the Ground Track Field class of the NWPU VHR-10 dataset.

#### 4.2.1. Anchor Generation

The anchor size for each dataset is generated via K-means clustering, which is adopted to restrain the anchors in Yolov5. The following is the function of K-means, which minimizes the clustering centers and each ground truths.

\[
E = \min \sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2 \\
\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x
\]

(2)

where \( E \) is the objective function. \( C_i \) is the clustering settings, which contain ground truths \( x \). The \( x \) is composed of \( (w, h) \) for each ground truths. \( k \) is the number of clustering centers; in this experiment, \( k \) is 9. Finally, \( \mu_i \) is the clustering centers, which is also anchor shapes.

The anchor shapes are generated by clustering the shapes of the ground truths and employing genetic evolution with the actual training loss criteria used during training. Table 1 reports the anchor shapes for each dataset. The detection procedure is applied on the P3, P4, and P5 feature maps and the detection area is subsequently increased. For the VEDAI_512 dataset, the anchors shapes \( (w, h) \) are \((27,11)\) \((13,27)\) \((27,18)\) for P3, \((22,25)\) \((17,43)\) \((46,17)\) for P4, and \((35,32)\) \((33,78)\) \((82,70)\) for P5. For the VEDAI_1024 dataset, the anchors shapes are \((35,34)\) \((25,54)\) \((47,52)\) for P3, \((64,40)\) \((70,63)\) \((55,92)\) for P4, and \((87,83)\) \((152,110)\) \((210,248)\) for P5. All anchor shapes are detailed in Table 1.

### Table 1. Anchor shapes for each feature map.

| Dataset       | Feature Maps (Size) | Anchor Shapes (w, h) |
|---------------|---------------------|----------------------|
| VEDAI_512     | P3 (80 × 80)        | (27, 11) (13, 27) (27, 18) |
|               | P4 (40 × 40)        | (22, 25) (17, 43) (46, 17) |
|               | P5 (20 × 20)        | (35, 32) (33, 78) (82, 70) |
| VEDAI_1024    | P3 (80 × 80)        | (44, 18) (19, 41) (42, 26) |
|               | P4 (40 × 40)        | (28, 39) (29, 58) (42, 41) |
|               | P5 (20 × 20)        | (71, 29) (48, 147) (115, 93) |
| NWPU VHR-10   | P3 (80 × 80)        | (35, 34) (25, 54) (47, 52) |
|               | P4 (40 × 40)        | (64, 40) (70, 63) (55, 92) |
|               | P5 (20 × 20)        | (87, 83) (152, 110) (210, 248) |
4.2.2. Learning Policy

For each dataset, the network is warmed up by a learning rate of $10^{-4}$ for 10 epochs. The initial learning rate is 0.01 and we used the cosine annealing of 0.012 to reduce the learning rate step by step. The optimization trick has a momentum of 0.9 and a weight decay of 0.0005. We employed the CIoU and GIoU [30,31] loss functions for the bounding box regression and classification, respectively.

4.3. Experiment Results

In this section, we evaluate Aggregated-Mosaic for its capability to improve the object detection performance compared to other current methods. All experiments were implemented and evaluated on the Pytorch framework. We compared the proposed method with the Basic [21], CutOut [37], GridMask [56,57], MixUp [12], CutMix [11], and Mosaic [28] augmentation methods. The Mosaic augmentation method is the baseline of the proposed method. Following the SSD settings, Basic represents the basic data augmentation (e.g., horizontal flip, random sample, and color distortion). Hyper-parameter $\alpha$ is equal to 0.5 for MixUp and CutMix, while the hyper-parameter $p$ of GridMask is 0.7. We adopted Soft-NMS to accurately fix the candidate bounding boxes.

4.3.1. VEDAI 512

Table 2 reports the mAP results on the VEDAI_512 validation dataset. Bold values denote the best performances. From the table, the baseline Mosaic achieves a mAP of 68.49%, while our proposed method achieves 71.21%, which is a 2.72% improvement compared to the baseline. Moreover, compared with the recent data augmentation methods (Basic, CutOut, GridMask, MixUp, and CutMix), the proposed Aggregated-Mosaic method achieves the best performance.

Figure 12 shows the detection results in VEDAI 512 dataset.

![Detection result: Ground Truth (right), Mosaic (middle), Aggregated-Mosaic (left).](image)

Table 2. Results of VEDAI 512.

| Class   | Basic [21] | CutOut [37] | GridMask [56] | MixUp [12] | CutMix [11] | Mosaic [28] | Proposed |
|---------|------------|-------------|---------------|------------|-------------|-------------|----------|
| Car     | 84.2       | 86.1        | 84.5          | 83.4       | 78.6        | 89.9        | 90.7     |
| Truck   | 63.1       | 59.6        | 59.2          | 60.8       | 57.3        | 60.1        | 61.3     |
| Ship    | 53.5       | 55.8        | 56.3          | 55.9       | 56.2        | 54.4        | 58.2     |
| Tractor | 76.6       | 80.3        | 79.7          | 78.5       | 79.5        | 82.1        | 83.1     |
| Camping | 71.3       | 69.9        | 68.5          | 67.7       | 68.1        | 66.7        | 72.1     |
| Van     | 55.9       | 52.8        | 49.7          | 53.2       | 56.9        | 57.6        | 57.4     |
| Vehicle | 44.8       | 39.2        | 42.9          | 47.5       | 43.5        | 46.2        | 48.1     |
| PickUp  | 72.6       | 73.6        | 71.9          | 74.6       | 75.3        | 74.8        | 76.1     |
| Plane   | 90.1       | 86.5        | 86.7          | 86.7       | 85.8        | 84.6        | 93.9     |
| mAP     | 68.01      | 67.09       | 66.60         | 67.59      | 66.80       | 68.49       | 71.21    |

4.3.2. VEDAI 1024

Table 3 reports the mAP results on the VEDAI_1024 validation dataset. Bold values denote the best performances. From the table, the baseline Mosaic achieves a mAP of 71.38%, while our proposed method achieves 76.82%, which is a 5.44% improvement compared to the baseline. Moreover, compared with the recent data augmentation methods (Basic, CutOut, GridMask, MixUp, and CutMix), the proposed Aggregated-Mosaic method achieves the best performance.

Figure 13 shows the detection results in VEDAI 1024 dataset.
Table 2. Results of VEDAI 512.

| Class   | Basic [21] | Cutout [37] | GridMask [36] | MixUp [12] | CutMix [11] | Mosaic [28] | Proposed |
|---------|------------|-------------|---------------|------------|-------------|-------------|-----------|
| Car     | 84.2       | 86.1        | 84.5          | 83.4       | 78.6        | 89.9        | 90.7      |
| Truck   | 63.1       | 59.6        | 59.2          | 60.8       | 57.3        | 60.1        | 61.3      |
| Ship    | 53.5       | 55.8        | 56.3          | 55.9       | 56.2        | 54.4        | 58.2      |
| Tractor | 76.6       | 80.3        | 79.7          | 78.5       | 79.5        | 82.1        | 83.1      |
| Camping | 71.3       | 69.9        | 68.5          | 67.7       | 68.1        | 66.7        | 72.1      |
| Van     | 55.9       | 52.8        | 49.7          | 53.2       | 56.9        | 57.6        | 57.4      |
| Vehicle | 44.8       | 39.2        | 42.9          | 47.5       | 43.5        | 46.2        | 48.1      |
| PickUp  | 72.6       | 73.6        | 71.9          | 74.6       | 75.3        | 74.8        | 76.1      |
| Plane   | 90.1       | 86.5        | 86.7          | 86.7       | 85.8        | 84.6        | 93.9      |
| mAP     | 68.01      | 67.09       | 66.60         | 67.59      | 66.80       | 68.49       | 71.21     |

4.3.2. VEDAI 1024

Table 3 reports the mAP results on the VEDAI_1024 validation dataset. Bold values denote the best performances. From the table, the baseline Mosaic achieves a mAP of 71.38%, while our proposed method achieves 76.82%, which is a 5.44% improvement compared to the baseline. Moreover, compared with the recent data augmentation methods (Basic, CutOut, GridMask, MixUp, and CutMix), the proposed Aggregated-Mosaic method achieves the best performance.

Figure 13 shows the detection results in VEDAI 1024 dataset.

![Detection result: Ground Truth (right), Mosaic (middle), Aggregated-Mosaic (left).](image)

Figure 13. Detection result: Ground Truth (right), Mosaic (middle), Aggregated-Mosaic (left).

Table 3. Results of VEDAI 1024.

| Class   | Basic [21] | Cutout [37] | GridMask [36] | MixUp [12] | CutMix [11] | Mosaic [28] | Proposed |
|---------|------------|-------------|---------------|------------|-------------|-------------|-----------|
| Car     | 90.4       | 91.1        | 88.2          | 85.4       | 81.3        | 91.1        | 92.9      |
| Truck   | 63.6       | 62.5        | 63.5          | 63.7       | 59.1        | 48.3        | 65.5      |
| Ship    | 66.4       | 57.3        | 58.7          | 62.1       | 62.4        | 67.3        | 69.7      |
| Tractor | 83.5       | 86.7        | 82.9          | 77.3       | 80.3        | 82.4        | 87.2      |
| Camping | 68.7       | 69.3        | 69.2          | 69.8       | 70.8        | 69.5        | 73.1      |
| Van     | 75.9       | 68.9        | 62.8          | 69.3       | 73.2        | 78.2        | 84.4      |
| Vehicle | 51.2       | 48.7        | 49.8          | 48.7       | 45.6        | 49.3        | 53.3      |
| PickUp  | 81.3       | 78.9        | 80.2          | 79.1       | 75.8        | 82.3        | 83.5      |
| Plane   | **93.6**   | 88.4        | 89.3          | 83.2       | 78.6        | 74          | 81.8      |
| mAP     | 74.96      | 72.42       | 71.62         | 70.96      | 69.68       | 71.38       | 76.82     |
4.3.3. NWPU-VHR 10

Table 4 reports the mAP results on the NWPU-VHR 10 validation dataset. Bold values denote the best performances. From the table, the baseline Mosaic achieves a mAP of 90.64%, while our proposed method achieves 96.12%, which is a 5.48% improvement compared to the baseline. Moreover, compared with the recent data augmentation methods (Basic, CutOut, GridMask, MixUp, and CutMix), the proposed Aggregated-Mosaic method achieves the best performance.

Figure 14 shows the detection results in NWPU-VHR 10 dataset.

Table 4. Results of NWPU-VHR 10.

| Class | Basic [21] | Cutout [37] | GridMask [56] | MixUp [12] | CutMix [11] | Mosaic [28] | Proposed |
|-------|-------------|-------------|---------------|------------|-------------|-------------|----------|
| PL    | 91.3        | 89.1        | 92.8          | 95.9       | 90.3        | 96.5        | 98.2     |
| SH    | 90.1        | 92.3        | 82.6          | 89.6       | **94.1**    | 93.1        | 98.9     |
| ST    | 85.2        | 91.2        | 87.9          | 95.3       | 89.9        | 83.3        | **98.4** |
| BD    | 94.3        | 95.4        | 94.7          | 96.4       | 94.3        | 99.2        | 99.5     |
| TC    | 92.1        | 93.5        | 92.5          | 95.7       | 96.1        | 92          | **99.6** |
| BC    | 85.3        | 82.6        | 91.9          | 93.9       | 87.6        | 88.6        | **99.5** |
| GT    | 89.1        | 93.8        | 95.1          | 83.9       | 92.8        | 96.5        | **98.0** |
| HA    | 88.5        | 91.7        | 90.3          | 88.5       | 91.5        | 90.7        | **92.0** |
| BR    | 76.2        | 82.5        | 88.2          | 84.6       | 78.9        | 76.4        | **89.8** |
| VH    | 91.1        | 89.1        | 91.7          | 95.3       | 92.7        | 90.1        | **97.3** |
| mAP   | 88.32       | 90.12       | 90.77         | 91.91      | 90.82       | 90.64       | **96.12** |

Figure 14. Detection result: Ground Truth (right), Mosaic (middle), Aggregated-Mosaic (left).
4.4. Discussion

From the experiment results, the proposed method fails to achieve the best results in some types of objects. The main reason is that if the objects are not distributed sparsely, the proposed method has a similar effect compared with the basic or sample fusion methods. Then, due to the Auto-Target-Duplication module, the training procedure will distract from those types of objects, and could not get the best result. Considering the influence of the hyper-parameter, we conducted an ablation study for Aggregated-Mosaic. The Assigned-Stitch component of the proposed method does not include any hyper-parameters in the augmentation procedure, while the Auto-Target-Duplication contains a hyper-parameter equal to the object duplication times. Thus, we perform an additional experiment that evaluates the effects of duplication times and gaussian blur on the three datasets based on the following.

X1: In Algorithm 2, randomly select one object for \(class_l\) from the training data and duplicate this to the Assigned-Stitch sample.

X5: In Algorithm 2, randomly select five objects for \(class_l\) from the training data and duplicate them to the Assigned-Stitch sample.

X10: In Algorithm 2, randomly select ten objects for \(class_l\) from the training data and duplicate them to the Assigned-Stitch sample.

Gaussian blur: When the object is duplicated to the Assigned-Stitch sample, the edge of the object patch is smoothed by Gaussian blurring (0 and 1 mean and variance, respectively) to the Assigned-Stitch sample.

Table 5 reveals that the Gaussian edge blur does not improve the detection performance, and thus the sharp edges of the object patch are better than the smoothed edges. In addition, the performance is optimized for the duplication times of 5 or 10.

| Duplication Times | Gaussian Blur | VEDAI_512 | VEDAI_1024 | NWPU-VHR 10 |
|-------------------|---------------|-----------|------------|-------------|
| X1                | ✓             | 70.82     | 76.45      | 95.85       |
| X5                | ✓             | 70.85     | 76.25      | 96.04       |
| X10               | ✓             | 70.93     | 76.30      | 95.68       |
|                   | ×             | 71.21     | 76.82      | 96.08       |
|                   | ×             | 71.13     | 76.53      | 96.12       |

5. Conclusions

In this paper, for training the object detection network in remote sensing imagery with sparse object distributions, we propose the Aggregated-Mosaic object detection method, which includes the Assigned-Stitch and Auto-Target-Duplication modules. The main improvements consist of two parts: (1) Based on the ground truth and Mosaic image sizes, the Assigned-Stitch augmentation enhances each training sample by accounting for objects, thus ensuring an efficient training procedure. (2) Objects are then carefully selected for duplication, and the Auto-Target-Duplication solves the sample imbalance, particularly for classes with poor results. The proposed method is evaluated with three remote sensing datasets, outperforming Mosaic (YOLOv5) by 2.72% and 5.44% at the 512 × 512 and 1024 × 1024 resolutions of the VEDAI dataset, respectively. Furthermore, on the VHR-10 dataset, the proposed method outperforms Mosaic (YOLOv5) by 5.48%. In future work, for the problem of lacking training samples, the proposed method will be combined with Generative Adversarial Networks to improve the detection performance for remote sensing imagery.
54. Razakarivony, S.; Jurie, F. Vehicle detection in aerial imagery: A small target detection benchmark. *J. Vis. Commun. Image Represent.* **2016**, *34*, 187–203. [CrossRef]

55. Cheng, G.; Han, J. A survey on object detection in optical remote sensing images. *ISPRS J. Photogramm. Remote Sens.* **2016**, *117*, 11–28. [CrossRef]

56. Chen, P.; Liu, S.; Zhao, H.; Jia, J. Gridmask data augmentation. *arXiv* 2020, arXiv:2001.04086.

57. Wang, J.; Jin, S.; Liu, W.; Liu, W.; Qian, C.; Luo, P. When human pose estimation meets robustness: Adversarial algorithms and benchmarks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Virtual, 19–25 June 2021; pp. 11855–11864.