remote sensing

MDPI

Article

CAMS-AI: A Coarse-to-Fine Framework for Efficient Small Object Detection in High-Resolution Images

Zhanqi Chen [1,2,3], Zhao Chen [1,2,3,*], Baohui Yang [4], Qian Guo [1,2,3], Haoran Wang [1,2,3] and Xiangquan Zeng [1,2,3]

[1] School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China
[2] Engineering Research Center for Forestry-Oriented Intelligent Information Processing, National Forestry and Grassland Administration, Beijing 100083, China
[3] Hebei Key Laboratory of Smart National Park, Beijing 100083, China
[4] Beijing Evialab Technology Co., Ltd., Beijing 100089, China
* Correspondence: cz71@bjfu.edu.cn

Highlights

What are the main findings?

- We propose a "coarse-to-fine" detection framework, CAMS-AI, which intelligently identifies and focuses on target regions using RPN and DBSCAN clustering, thereby avoiding redundant computations on massive invalid backgrounds.
- Experiments demonstrate that compared to the SAHI method, the CAMS-AI framework achieves an 8 to 10-fold+ increase in end-to-end inference speed (End-to-End FPS) while sacrificing only a minimal, acceptable amount of accuracy (a 2–4 percentage point drop in $mAP_{50–95}$).

What are the implications of the main findings?

- The framework successfully overcomes the severe efficiency bottlenecks of existing slicing methods in sparse-target scenarios (e.g., wide-area grasslands), providing a highly efficient and practical solution for high-resolution, wide-area livestock monitoring.
- CAMS-AI's "intelligent focus" paradigm can be seen as a general solution, generalizable to other fields facing the "high-resolution, large field-of-view, sparse-target" challenge, such as remote sensing and UAV inspection.

Abstract

Automated livestock monitoring in wide-area grasslands is a critical component of smart agriculture development. Devices such as Unmanned Aerial Vehicles (UAVs), remote sensing, and high-mounted cameras provide unique monitoring perspectives for this purpose. The high-resolution images they capture cover vast grassland backgrounds, where targets often appear as small, distant objects and are extremely unevenly distributed. Applying standard detectors directly to such images yields poor results and extremely high miss rates. To improve the detection accuracy of small targets in high-resolution images, methods represented by Slicing Aided Hyper Inference (SAHI) have been widely adopted. However, in specific scenarios, SAHI's drawbacks are dramatically amplified. Its strategy of uniform global slicing divides each original image into a fixed number of sub-images, many of which may be pure background (negative samples) containing no targets. This results in a significant waste of computational resources and a precipitous drop in inference speed, falling far short of practical application requirements. To resolve this conflict between accuracy and efficiency, this paper proposes an efficient detection framework named CAMS-AI (Clustering and Adaptive Multi-level Slicing for Aided Inference). CAMS-AI adopts a "coarse-to-fine" intelligent focusing strategy: First, a Region Proposal Network

Check for updates

Academic Editor: Xinghua Li

Received: 6 November 2025
Revised: 8 January 2026
Accepted: 12 January 2026
Published: 14 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

(RPN) is used to rapidly locate all potential target areas. Next, a clustering algorithm is employed to generate precise Regions of Interest (ROIs), effectively focusing computational resources on target-dense areas. Finally, an innovative multi-level slicing strategy and a high-precision model are applied only to these high-quality ROIs for fine-grained detection. Experimental results demonstrate that the CAMS-AI framework achieves a mean Average Precision (mAP) comparable to SAHI while significantly increasing inference speed. Taking the RT-DETR detector as an example, while achieving 96% of the $mAP_{50–95}$ accuracy level of the SAHI method, CAMS-AI's end-to-end frames per second (FPS) is 10.3 times that of SAHI, showcasing its immense application potential in real-world, high-resolution monitoring scenarios.

Keywords: object detection; small object detection; high-resolution images; slice inference; smart agriculture; livestock monitoring

1. Introduction

As a vital ecosystem on Earth, the health and sustainable development of grasslands are crucial for maintaining biodiversity, regulating climate, and supporting animal husbandry [1]. Grasslands provide many important ecosystem functions, such as carbon storage, climate regulation, biodiversity support, and the supply of resources like water [2]. Grasslands also form the foundation for the development of the livestock industry, and their primary productivity directly determines the carrying capacity of animal husbandry; however, fodder and food on such lands are often limited [3]. Overgrazing and poor land management practices can lead to soil degradation, deforestation, and biodiversity loss, impacting ecosystem health [4].

Accurate and timely livestock counting (such as cattle and sheep) is fundamental to preventing overgrazing, assessing grassland degradation, and formulating scientific grazing strategies. Traditional livestock inventory methods rely on manual field surveys, which are not only time-consuming and labor-intensive but also struggle to cover vast regions. Furthermore, the statistical results often suffer from time lags and significant errors [5]. In animal husbandry, the application of various monitoring and identification devices, along with computer vision technologies, has become commonplace. This has significantly improved the accuracy and efficiency of farming, reduced reliance on labor and resources, lowered farming costs, and promoted the sustainable development of the livestock industry [6].

Utilizing these remote sensing and surveillance camera devices for large-scale, all-weather automated livestock monitoring has emerged as a highly promising novel solution. This approach enables continuous monitoring of wide-area grasslands at a low cost, providing unprecedented technological opportunities for precision livestock farming and smart grassland management.

Despite this promising outlook, directly applying existing object detection technologies to high-resolution images faces a series of challenges: (1) Ultra-high resolution and computational bottlenecks: Directly processing images of 2K (2560×1440) or even higher resolutions imposes severe computational and GPU memory burdens. (2) Prevalence of small targets: Distant cattle and sheep occupy only tiny pixel regions in the image, classifying them as typical small targets, which are extremely difficult to detect. (3) Extremely uneven target distribution: Livestock typically cluster together, resulting in large portions of the image being empty grassland background, while a few regions are highly dense with targets.

Baseline tests conducted on our dataset confirm that these problems are very difficult. Constrained by model design, GPU memory, and computational load, modern deep learning object detection models typically need to resize input images to smaller dimensions, such as 512 × 512 or 640 × 640, to ensure the feasibility of training and inference. However, this resizing causes small targets, which already occupy very few pixels, to nearly vanish on the feature maps. This severe information loss further degrades the detector's recall rate. When a pre-trained RT-DETR detector is used directly on the original high-resolution images, its detection accuracy is extremely low, with the model's Average Precision (AP) at only 1.7%. The Toolkit for Identifying Detection Errors (TIDE) reveals that the primary error type is "Miss" (missed detections), which contributes up to 79.34% of the total error. This indicates that standard detectors are almost completely ineffective in this scenario.

Since the breakthrough of Deep Convolutional Neural Networks (DCNNs), general object detection technology has undergone rapid development. Its evolutionary path can be broadly divided into several stages: two-stage detectors represented by the R-CNN family [7] (including Fast R-CNN [8] and Faster R-CNN [9]); one-stage detectors represented by the YOLO series [10] and SSD [11]; and Transformer-based architectures like DETR [12] and its efficient variant, RT-DETR [13]. Although these models perform exceptionally well on general-purpose datasets, their inherent structural flaws are exposed when applied directly to specialized scenarios dominated by high-resolution images and small targets. The primary reason for their deficiency in small object detection scenarios is the hierarchical downsampling structure designed to acquire high-level semantic information [14].

First is the problem of feature dilution and disappearance. As the original image passes through a series of convolutional and pooling layers, the spatial resolution of the feature map is progressively compressed. This inevitably weakens the representation of tiny targets [15]. A small object that occupies only a few dozen pixels in the original image may be reduced to just a few pixels or disappear entirely in the deep, high-dimensional feature maps after multiple downsampling layers, making it impossible for the network to effectively capture the target [16]. Second is the mismatch between the receptive field and target scale. To capture global information from large objects, deep networks typically have large receptive fields. When such large receptive fields are used to process small targets, the perceived area includes a large amount of irrelevant background pixels. This noise seriously interferes with the effective feature extraction for small targets [17].

Cropping the original image to increase the object-to-image pixel ratio, thereby improving object detection and counting performance, is an idea that effectively addresses small object detection in high-resolution scenarios [18]. Consequently, methods represented by Slicing Aided Hyper Inference (SAHI) emerged. SAHI is a universal solution based on slicing-aided inference and fine-tuning for small object detection in high-resolution images, all while maintaining higher memory utilization [19]. The core idea of SAHI is to slice the high-resolution original image into several small patches with overlapping regions, perform detection on each patch individually, and then merge the results back to the original image coordinates. This method can significantly improve the recall rate and detection accuracy of small targets.

Mazen et al. combined the SAHI method with classic models like Faster R-CNN and Cascaded R-CNN, applying it to artworks and notably improving the models' performance in detecting small-sized heads in large photographs [20]. Fotouhi et al. applied the SAHI method to the YOLOv8 model, overcoming the core difficulty of small insect detection and achieving a breakthrough in agricultural pest monitoring [21]. Pereira et al. used a YOLOR-CSP architecture combined with the SAHI framework as a new method for detecting fundus lesions, improving detection metrics on retinal images and significantly enhancing diagnosis and treatment outcomes [22]. Zhorif et al. employed the YOLOv8m object

detection model with the SAHI framework to improve the precision of detection models on high-resolution aerial images, but they also highlighted the problem of a significant reduction in detection efficiency caused by the SAHI framework [23]. On our dataset, the mere introduction of the SAHI framework boosted the model's AP from 1.7% to 64.5%, proving its effectiveness in handling small object problems.

However, SAHI's fixed slicing strategy, which operates according to preset dimensions and overlap rates, sees its drawbacks dramatically amplified in sparsely populated scenes like grasslands. First, slicing generates a large number of redundant negative samples: In the scenario of this study, a 2688 × 1520 resolution image is sliced into 15 sub-images of 640 × 640. Due to the clustering nature of livestock, the vast majority of these 15 sub-images may only contain grassland, acting as "negative samples" with no valid targets. Performing inference on these negative sample slices wastes a significant amount of computational resources. Second, the large number of slices causes a precipitous drop in inference speed. Experiments show that after applying SAHI, the end-to-end processing frame rate, including all preprocessing, prediction, and merging post-processing steps, is only 1.82 frames per second (FPS). Such low efficiency makes it difficult to apply in practical management tasks that require near-real-time feedback.

Different algorithms or scenarios require trade-offs between speed, accuracy, and efficiency; for agricultural and pastoral tasks, higher speed is usually required [24]. Zhang et al. [25] proposed a novel adaptive slicing method called ASAHI (Adaptive Slicing Aided Hyper Inference), It can significantly reduce redundant computation by using adaptive slice sizes, focusing on the number of slices rather than the slice size, and can adaptively adjust the slice size based on image resolution to control the slice count. Although this method effectively reduces the number of slices, it cannot filter out background regions in the image, meaning a large amount of computational resources is still wasted.

Therefore, how to maintain the high accuracy of SAHI while overcoming the severe speed bottleneck it introduces has become the core problem this study aims to solve.

The main contributions of this paper are as follows:

(1) We propose an efficient coarse-to-fine detection framework. CAMS-AI first uses a Region Proposal Network to rapidly lock onto potential target areas and then performs streamlined processing only within these regions, thereby avoiding useless computations on massive background areas.
(2) We construct a novel multi-level slicing strategy. We designed a three-level adaptive slicing strategy—comprising centered slicing, expanded slicing, and grid slicing—to target ROIs of different sizes and densities, achieving precise allocation of computational resources.
(3) We achieve a balance between accuracy and speed. Extensive experiments demonstrate that the CAMS-AI framework achieves inference speeds nearly 10 times faster than SAHI while attaining comparable accuracy, providing a practical and feasible solution for efficient detection in high-resolution, small-target scenarios.

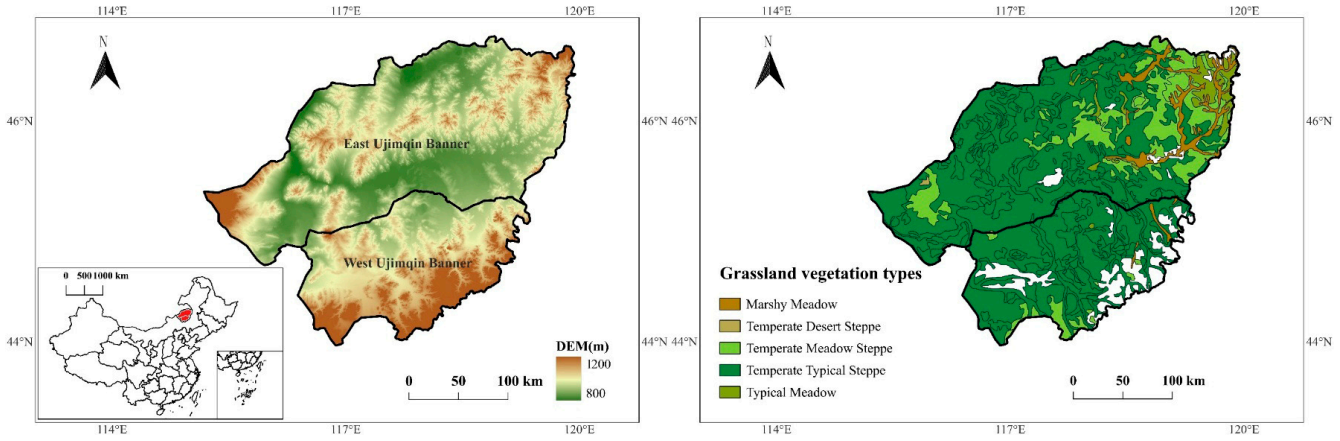
2. Materials and Methods

2.1. Dataset Details

2.1.1. Study Area

To realistically and effectively evaluate the algorithm's performance in wide-area grassland monitoring scenarios, we constructed a dataset named "PrairieLivestock." All images in this dataset originate from the Wulagai Management Area, Xilingol League, Inner Mongolia Autonomous Region (Figure 1), located between 118°44′–119°50′E longitude and 45°29′–46°38′N latitude. This area spans the West Ujimqin Banner and East Ujimqin

Banner. As a typical temperate grassland, this region's geographical environment and pastoral practices are broadly representative.



(**a**) (**b**)

Figure 1. Wulagai Management District: (**a**) digital elevation model and location; (**b**) grassland vegetation types.

The dataset comprises a total of 460 high-resolution (2688 × 1520 pixels) images. The data was collected from July to August 2025. During this period, the weather was clear, the climate was stable, and lighting conditions were good, ensuring the high quality of the images and the consistency of the scenes. To increase data diversity, we collected samples from two different locations, as detailed in Table 1.

Table 1. Introduction to the sample plot.

Locations	Latitude and Longitude	Number
Wulagai Bulinquan	118.810122, 45.800929	243
Wulagai Hechang Reservoir	119.240714, 46.249838	217

As shown in Figure 2, the images from the Bulinquan collection point contain some relatively clear, larger-sized medium-to-close-range targets, whereas the images from the Hechang Reservoir collection point consist almost entirely of distant small targets. The data from these two locations complement each other, jointly forming a dataset that comprehensively reflects the scale variation challenges present in real-world scenarios.

2.1.2. Target Class and Size Distribution

The dataset includes three main detection categories: Cattle, Horse, and Sheep. To quantify the "small target" characteristics of the dataset, we compiled statistics on the area of all bounding boxes based on the official Microsoft Common Objects in Context (MS COCO) definitions. The results are shown in Table 2.

The MS COCO metric uses absolute pixel scales to classify target sizes, but this absolute definition has significant limitations. It ignores the relationship between the target's size and its image context, failing to truly reflect the practical challenges models face when processing high-resolution images. To better define the scale of small targets, researchers often need to create custom definitions based on the dataset's characteristics, rather than just relying on the bounding box size [26].

Table 2. Dataset object size distribution based on MS COCO.

Locations	Class	Small Area < 32^2	Medium $32^2 \leq$ Area < 96^2	Large Area $\geq 96^2$
Wulagai Bulinquan	Cattle	877	1650	0
	Horse	41	0	0
	Sheep	1546	44	0
Wulagai Hechang Reservoir	Cattle	5043	1389	0
	Horse	4449	1584	0
	Sheep	179	0	0
Total		12,135	4667	0

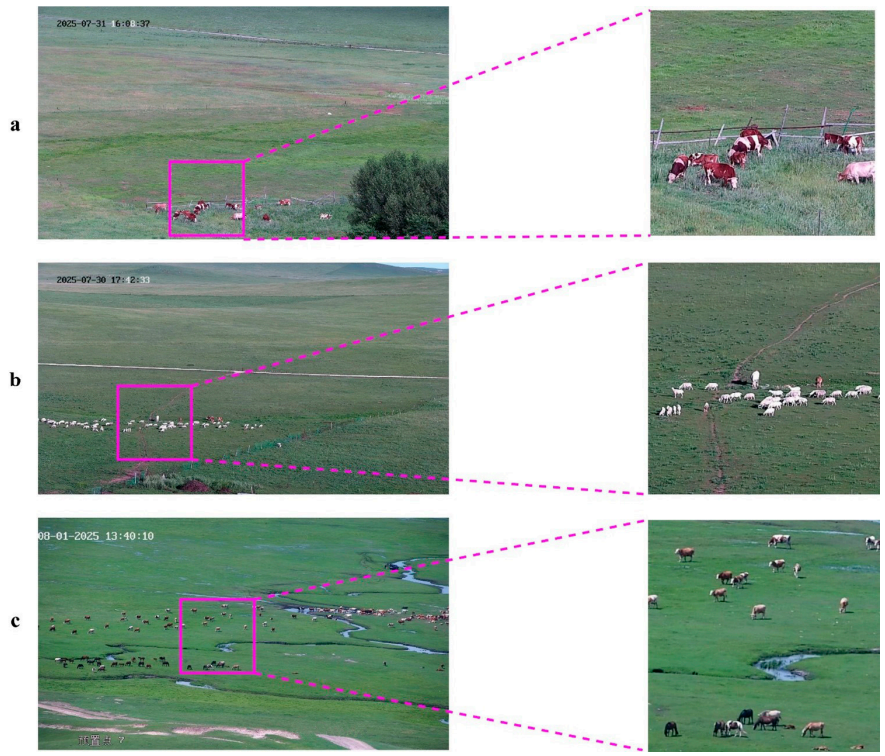


Figure 2. Sample images from the "PrairieLivestock" dataset. (**a**) Cattle and (**b**) Sheep from the Wulagai Bulin Spring location. (**c**) Cattle and horses from the Wulagai Hechang Reservoir location, with zoomed-in sections highlighting the targets.

To more accurately characterize the dataset used in this study, we adopt a relative size measure (rs) that better reflects detection difficulty. This measure is defined as the square root of the ratio of the target bounding box area to the total image area:

$$\mathrm{rs} = \sqrt{\frac{a_{\mathrm{object}}}{a_{\mathrm{image}}}} \tag{1}$$

Here, a_{object} represents the target bounding box area, and a_{image} represents the total image area. It normalizes the target scale, providing a more robust, resolution-independent evaluation metric. We classify target sizes based on this metric.

Analysis of the relative size results (Table 3) reveals that a staggering 98.94% of the targets fall into the "Tiny" category, making this a dataset overwhelmingly dominated by relatively small targets. This characteristic places high demands on detection algorithms, as conventional Feature Pyramid Networks (FPN) may completely lose the effective information of these minute targets during the initial downsampling stages [27].

Table 3. Dataset object size distribution based on custom relative size definition.

Statistics	Tiny rs < 0.03	Small 0.03 ≤ rs < 0.1	Medium 0.1 ≤ rs < 0.3	Large rs > 0.3
Number	16,624 98.94%	178 1.06%	0 0%	0 0%

2.2. Dataset Processing

2.2.1. Data Processing Strategy

Because the CAMS-AI framework proposed in this study slices the dataset images, its final End-to-End performance—particularly its inference speed (FPS)—cannot be accurately evaluated through conventional, idealized model validation methods. Conventional methods ignore the additional overhead introduced by processes such as slice generation, image cropping, and results merging. To authentically measure the efficiency of CAMS-AI in a real-world deployment scenario, it is necessary to time the complete detection workflow on a sufficiently large test set.

If the dataset were split using a standard 8:1:1 ratio, the test set would only contain about 46 images, which is insufficient for obtaining stable and credible end-to-end performance metrics. Considering the different data requirements of CAMS-AI's two-stage model, the Region Proposal Network (RPN) in the CAMS-AI framework is trained on the original resolution images. Its main task is to identify general target regions, and its requirement for training data volume is relatively low. Conversely, the fine-grained detection model is trained on sliced images. By slicing the original training set, the data volume is already expanded several-fold, which is sufficient to meet its training needs.

Based on these considerations, this study abandons the traditional proportional splitting method. Instead, it designs a specialized data processing workflow with the goal of reserving a larger-scale test set for end-to-end performance validation, while simultaneously ensuring that the models in both stages are adequately trained.

2.2.2. Full-Resolution Dataset

This dataset directly uses the original 2688 × 1520 resolution images. Its primary purposes are: first, to train and validate the Region Proposal Network (RPN); and second, to serve as the final test benchmark for evaluating the end-to-end accuracy and speed of the entire CAMS-AI framework.

This study adopts the Hold-out Method. From the 460 original images, 100 images are prioritized and set aside as an independent, fixed Test Set. This test set does not participate in any training phase and is used only for the final performance evaluation. The remaining 360 images are then divided into a training set and a validation set at an 8:2 ratio. The detailed split is shown in Table 4.

Table 4. Splitting of the full-resolution dataset using the hold-out method.

Use	Locations	Train	Valid	Test	Total
Train	Wulagai Bulinquan Wulagai Hechang Reservoir	154 133	39 34		360
Test	Wulagai Bulinquan Wulagai Hechang Reservoir			50 50	100

2.2.3. Sliced Dataset

This dataset is specifically used for training and validating the fine-grained detection model. It is derived from the 360 training and validation set images (from the Full-Resolution Dataset described above). The SAHI slicing algorithm is used to preprocess

these images. The slice width and height are set to 640 pixels, with horizontal and vertical overlap rates set to 0.2.

As shown in Figure 3, each 2688 × 1520 original image is uniformly cut into 15 overlapping 640 × 640 slices. This initial slicing of the entire dataset generated 5400 sub-images. Among these, 3535 were negative samples containing no targets, meaning the negative sample proportion was as high as 65.5%.

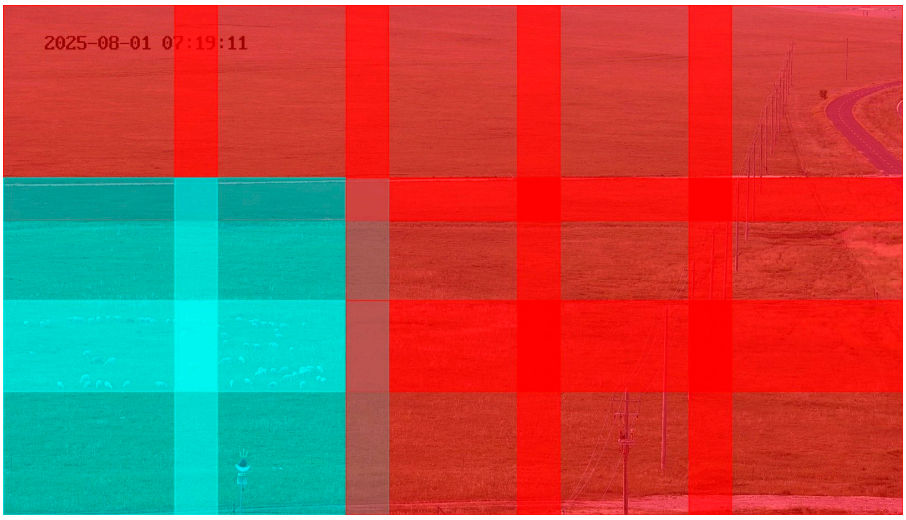


Figure 3. SAHI slicing visualization (640 × 640 slices, 20% overlap): blue slices contain targets, red slices contain no targets.

An excessively high proportion of negative samples is detrimental to model convergence and reduces training efficiency. To address this, the negative samples were filtered, setting the final ratio of negative samples to positive samples in the dataset to approximately 0.2 (i.e., 1 negative sample was kept for every 5 positive samples). This filtered sliced dataset was then further divided into training, validation, and test sets for the independent development and evaluation of the fine-grained detection model. The final composition of the sliced dataset is shown in Table 5.

Table 5. Splitting of the sliced dataset after negative sample filtering.

Locations	Type	Train	Valid	Test
Wulagai Bulinquan	Positive Negative	576 115	72 14	73 0
Wulagai Hechang Reservoir	Positive Negative	915 182	114 22	115 24
Total		1788	222	227

2.3. Methods

To address the efficiency and accuracy challenges of detecting sparse small targets in high-resolution grassland images, this study proposes the CAMS-AI framework (Figure 4). It is a two-stage "Coarse-to-Fine" detection pipeline. Its core idea is to move away from fixed slicing of the entire image and instead use intelligent analysis to focus computational resources on the Regions of Interest (ROIs) most likely to contain targets. The algorithm primarily consists of the following four modules:

- Coarse-Grained Candidate Localization: Performing a rapid, preliminary detection sweep on the full-resolution image to recall all potential targets.
- Adaptive ROI Generation: Using the DBSCAN clustering algorithm to organize the discrete, preliminarily detected targets into several compact ROIs.

- Multi-Level Slicing and Fine-Grained Detection: Applying a multi-level slicing strategy to crop the generated ROIs and then invoking a high-performance model for precise detection.
- Result Fusion and Post-Processing: Merging the detection results from both the coarse and fine stages and applying Non-Maximum Suppression (NMS) to obtain the final output.

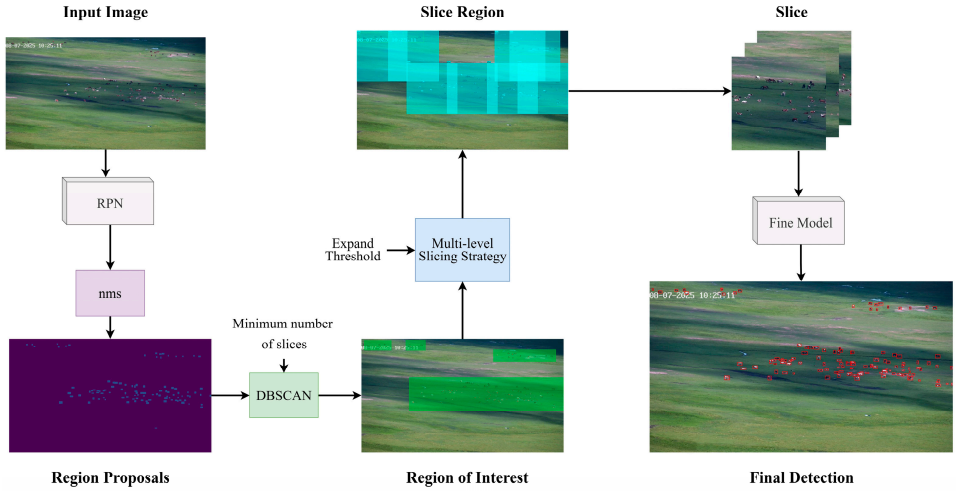


Figure 4. The architecture of CAMS-AI. (1) Input Image and coarse localization via RPN; (2) Region Proposals filtering and Region of Interest (ROI) generation using DBSCAN; (3) Slice Region identification and Slice generation through the multi-level slicing strategy; and (4) Final Detection via result fusion.

2.3.1. Coarse-Grained Candidate Localization

The objective of this stage is to rapidly identify all regions on the 2688 × 1520 full-resolution image that might contain targets, while keeping computational overhead as low as possible. This study employs a Region Proposal Network (RPN) using ResNet18 as the backbone, which is trained using the MMDetection toolbox [28] (as shown in Figure 5).

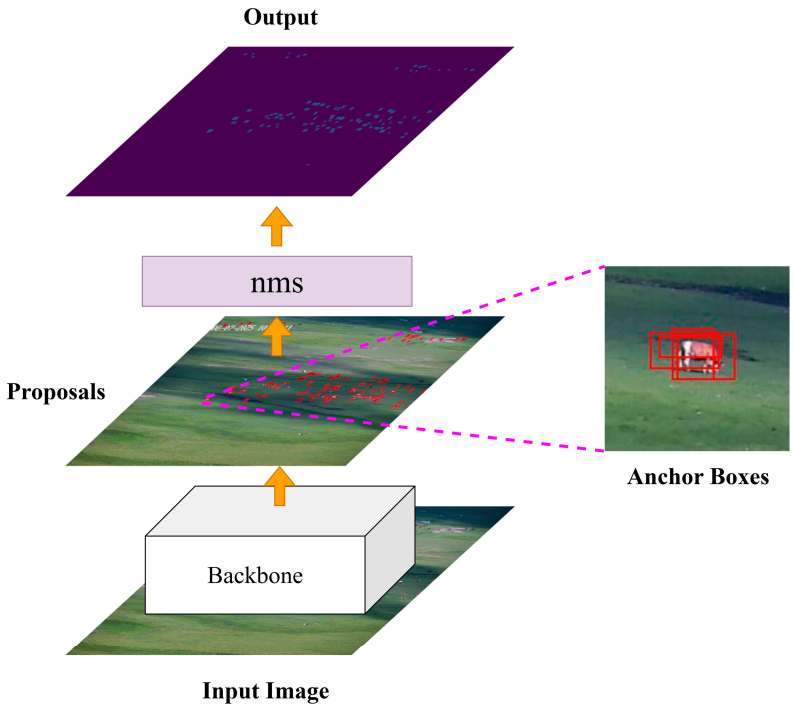


Figure 5. The architecture of RPN in CAMS-AI.

To enhance the recall rate for small targets in high-resolution imagery, this study adopted several optimization strategies. First, to mitigate potential overfitting issues arising from the limited training dataset, data augmentation, such as PhotoMetricDistortion, was introduced into the training pipeline. Second, due to the relatively small area covered by targets in the images, matching RPN anchor boxes is difficult. Therefore, the base scales of the RPN anchor generator were adjusted to (4, 8). This generated smaller anchor boxes, significantly enhancing the matching and recall capabilities for small targets.

In this stage, the trained RPN possesses a parameter count of 14.38 M and a computational complexity of 78.1 GFLOPs. During the training and inference phases, the RPN automatically resizes the original 2688 × 1520 resolution images to 1344 × 768 using an aspect-ratio-preserving scaling strategy. This approach effectively avoids significant target deformation while ensuring that the input dimensions remain within the scale robustness zone of Convolutional Neural Networks (CNNs). Experimental results demonstrate that after 100 training epochs, the RPN exhibits commendable recall performance in candidate box generation; specifically, the Average Recall (AR) reaches 0.4019 for AR@100, 0.4183 for AR@300, and 0.4183 for AR@1000.

2.3.2. Region of Interest (ROI) Generation

This is a core component of the CAMS-AI framework, designed to refine the candidate boxes generated by the proposal network into Regions of Interest (ROIs) worthy of fine-grained detection. It is an unsupervised region selection algorithm that can be easily embedded into any network.

Common clustering algorithms include K-means and DBSCAN. While K-means is simple to implement and converges quickly, it requires the user to pre-specify the number of clusters (K-value) and tends to find spherical or convex clusters. This makes it difficult to adapt to the scenarios in this paper, which involve naturally formed livestock clusters that are irregular in shape and unknown in quantity.

Therefore, this study uses the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [29] to cluster all center points. DBSCAN can discover clusters of arbitrary shapes and is insensitive to noise points (outlying individual targets), making it highly suitable for identifying livestock aggregation areas.

The DBSCAN algorithm has two main parameters: the neighborhood radius, Eps-neighborhood (eps); and the minimum number of points required within that radius to be considered a core object, MinPts.

The Eps-neighborhood of a data point p refers to the set of all points in the dataset D whose distance from p is no greater than eps. This set can be expressed as:

$$N_\varepsilon(p) = \{q \in D | dist(p, q) \leq \varepsilon\} \quad (2)$$

Here, ε is the eps (neighborhood radius), D is the set of all data points, and dist(p,q) is the distance function between point p and point q (this study uses Euclidean distance); the parameter ε defines the radius of the neighborhood. If the number of sample points within a given object's Eps-neighborhood is greater than or equal to MinPts, that object is termed a core object. MinPts thus defines the minimum number of data points required to constitute a dense region.

As shown in Figure 6, after clustering is complete, each point cluster generates a rectangular ROI. The coordinates of all points within the cluster are computed to form a rectangular bounding box that completely encloses the target group. Subsequently, a certain amount of padding is added to this bounding box. This ensures the integrity of the target group's edges and allows surrounding contextual information to be included in the ROI, providing richer features for the subsequent fine-grained detection.

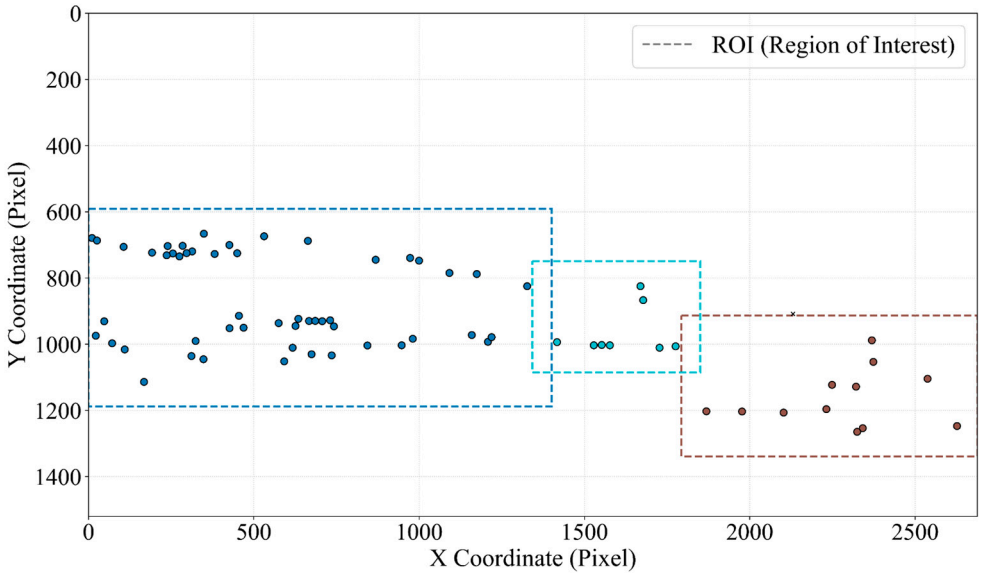


Figure 6. Schematic Diagram of Clustering to ROI Generation. Different colors are used to distinguish the clustered ROIs for visual clarity. The grey dashed line in the legend represents ROI (Region of Interest) for illustrative purposes.

2.3.3. Multi-Level Slicing Strategy

After obtaining the ROIs, the goal of this stage is to perform efficient and precise detection on these small regions. A simple and intuitive approach is to apply a fixed sliding-window grid slice to all ROIs whose dimensions exceed the model's input size. This strategy leads to a severe redundant slicing problem for boundary-case sizes. For example, if an ROI has dimensions of 641 $\times$ 600 pixels, a fixed grid slicing algorithm would generate two slices along the width axis with an overlap of 99.8%. These two detections are highly redundant computationally, drastically reducing efficiency.

Modern object detection models (like YOLO, RT-DETR, etc.) preset input dimensions based on limitations such as computational load and training device memory. Before an image is fed into the model during the prediction phase, it is first resized to the model's native input size. Directly resizing an ultra-high-resolution image (e.g., 2688 $\times$ 1520) to the model's set size causes small object information to be lost due to drastic pixel compression and deformation. However, resizing a square-like image that is slightly larger than the native size and within a reasonable range results in negligible accuracy loss. This is attributed to the fact that state-of-the-art Convolutional Neural Networks (CNNs) exhibit a degree of scale robustness; as scale-related information is progressively encoded through convolutional layers, model performance remains relatively stable even when a slight discrepancy exists between the inference and training resolutions. Consequently, by maintaining the input within a "robustness zone" near the original scale, we can effectively preserve small target information with negligible accuracy loss while avoiding the redundant computational overhead of multi-scale tiling [30].

Based on this idea, this study designs a Multi-level Slicing Strategy (Figure 7). It can flexibly select scales to maximize the avoidance of redundant slices while guaranteeing accuracy.

Given a Region of Interest R with attributes width W_R and height H_R, the set of slices it produces is:

$$\text{Slices}(R)\begin{cases}\text{CenterSlice}(R,S) & \text{if } \max(W_R,H_R)\le S\\ \text{ExpandSlice}(R) & \text{if } S<\max(W_R,H_R)\le T\\ \text{GridSlice}(R,S,O) & \text{if } \max(W_R,H_R)>T\end{cases} \quad (3)$$

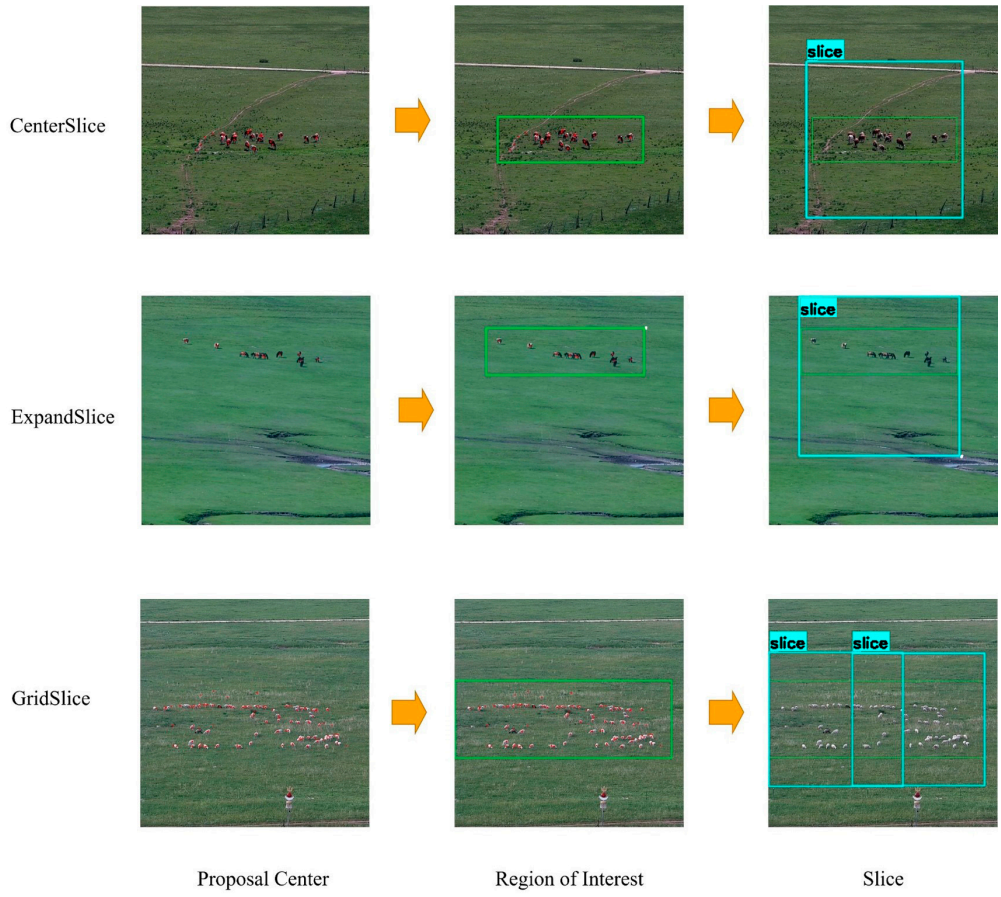


Figure 7. Visualization of the Multi-level Slicing Strategy.

Here, S is the standard slice size (S = 640 in this study), and T is Expand Threshold (T = 720 in this study). The specific definitions for each slicing strategy are as follows:

1. Center Slice Strategy: When the ROI's dimensions are less than or equal to the standard slice size, we adopt the center slice strategy. Let the center point of the ROI be (c_x, c_y). The single generated slice's coordinates are:

$$s_c = (c_x - \frac{S}{2}, c_y - \frac{S}{2}, c_x + \frac{S}{2}, c_y + \frac{S}{2}) \quad (4)$$

This is the most efficient solution and provides sufficient contextual information for small target clusters.

2. Expand Slice Strategy: When the ROI's dimensions are larger than the standard slice size but smaller than the expansion threshold, we adopt the expand-to-square slice strategy. The maximum edge length of this ROI is D_{max}, and the center point is still (c_x, c_y). The single generated slice's coordinates are:

$$s_e = (c_x - \frac{D_{max}}{2}, c_y - \frac{D_{max}}{2}, c_x + \frac{D_{max}}{2}, c_y + \frac{D_{max}}{2}) \quad (5)$$

This strategy generates the smallest square slice that fully encloses the original ROI, avoiding high overlapping slices caused by slight size deviations beyond the threshold. For example, for a 641 × 600 ROI, the system generates a 641 × 641 slice and feeds it directly into the model. Thanks to the model's scale robustness, the accuracy of this single detection is nearly indistinguishable from processing two highly overlapping 640 × 640 slices, while halving computational overhead. This strategy fundamentally resolves redundant slicing issues at boundary dimensions.

3. Grid Slice Strategy: If a standard grid slice is applied to a large ROI with only one long edge exceeding the threshold, it may lose context on the short edge that could contain valid information. Therefore, this study proposes a hybrid slicing strategy combining centering and grid slicing. First, define the slice step size $P = S(1 - O)$. Then, a set of starting offsets must be calculated for the *X*-axis and *Y*-axis separately. For any given axis (using the *X*-axis as an example, with its starting point in the ROI as x_1 and its size as W_R), the set of offsets Offset_X is:

$$\text{Offset}_x = \begin{cases} x_1 + \frac{W_R}{2} - \frac{S}{2} & \text{if } W_R \leq S \\ \{x_1 + k \cdot P | k \in \mathbb{N}_0, x_1 + k \cdot P + S \leq x_1 + W_R\} \cup \{x_1 + W_R - S\} & \text{if } W_R > S \end{cases} \quad (6)$$

The calculation for Offset_Y is identical to Offset_X. The final generated set of slices C_{grid} is the Cartesian product of these two offset sets:

$$C_{grid} = \{(x, y, x + S, y + S) | x \in \text{Offset}_X, y \in \text{Offset}_Y\} \quad (7)$$

This performs a sliding-window slicing along the length and width of the ROI at a certain overlap rate, ensuring complete coverage of the entire area while also ensuring that each slice's dimensions are within a range the model can efficiently process.

2.3.4. Fine-Grained Detection and Post-Processing

After slicing is complete, a more powerful fine-grained detection model (Fine Model) is used, along with a relatively strict confidence threshold that still ensures recall, to detect these intelligently selected and generated slices.

The final detection results are composed of both the region proposal boxes and the fine-grained detection boxes. This is done to retain those individual, outlying targets that might exist, compensating for potential missed detections during the clustering process. These two sets of detection results are merged, and a global Non-Maximum Suppression (NMS) is performed once to eliminate potential overlapping boxes from the different sources, ultimately generating a precise and non-redundant detection output.

3. Results

3.1. Experimental Details

For the training phase, the setup details used in the subsequent experiments of this study are shown in Table 6. The RPN was trained using the MMDetection toolbox (v3.1.0). RT-DETR, YOLOv8, and YOLOv11 were trained using Ultralytics. All training was conducted under identical hardware conditions and operating systems.

During the model training phase, the input resolution for all base detectors (YOLOv8, YOLOv11, and RT-DETR) was uniformly set to 640 pixels. The training configuration consisted of 100 epochs with a batch size of 8, and all methods were evaluated using identical pre-processing and post-processing steps. To ensure the rigor of formal testing, a GPU warm-up was conducted using a sample image to activate Compute Unified Device Architecture (CUDA) cores and initialize video memory prior to evaluation. Subsequently, all timing operations were executed following CUDA synchronization commands to eliminate timing biases caused by asynchronous CUDA execution. The recorded inference latency encompasses the entire pipeline, from the input of the raw high-resolution image to the output of the final results. The final FPS was derived by timing each of the 100 full-resolution images in the test set individually and calculating the average value.

Table 6. Runtime environment.

Type	Details
Hardware	
CPU	16 CPU Intel(R) Xeon(R) Platinum 8352 V @ 2.10 GHz
Memory	32 GB
GPU	Nvidia GeForce RTX 4090 (24 GB)
Software	
OS	Ubuntu 22.04
CUDA	12.1
Python	3.8.10
Pytorch	1.11.0
mmcv	2.0.1
mmdet	3.1.0
mmengine	0.8.3
Ultralytics	8.0.201

3.2. Evaluation Metrics

To comprehensively and objectively evaluate the performance of the CAMS-AI framework proposed in this paper, as well as other comparative methods, this study employs an evaluation system that includes both standard accuracy metrics and custom efficiency metrics.

- Accuracy Metrics
  This study uses the industry-standard MS COCO evaluation metrics to measure the detection accuracy of the models. These metrics provide a comprehensive assessment of the model's localization and classification capabilities from multiple dimensions. The main evaluation metrics used in this paper are mean Average Precision (mAP), including mAP_{50}, $mAP_{50–95}$, and mAP_{small}, as well as Average Recall (AR), reported as AR_{100}.
- Efficiency Metrics
  Standard model speed evaluations only measure the model's forward inference time and cannot accurately gauge the framework's practical performance in a slicing scenario. To address this, this paper defines and adopts a more practical efficiency metric: End-to-End FPS (End-to-End Frames Per Second). This metric measures the actual processing speed of the entire detection pipeline, in units of "frames per second." Unlike traditional FPS, the timing for End-to-End FPS starts when a raw high-resolution image is input and ends when the final, post-processed detection results are output. Its timing range explicitly includes the time spent on all steps: clustering and ROI generation, preprocessing (slice generation), model inference, and post-processing (such as coordinate transformation, results fusion, and Non-Maximum Suppression NMS). Therefore, this metric can more realistically and fairly reflect the actual operational efficiency of different methods in deployed applications.

3.3. Parameter Settings

3.3.1. Negative Sample Ratio

When filtering negative samples in the original sliced dataset, a specific positive-to-negative sample ratio must be established. Tests conducted across three ratios—0.1, 0.2, and 0.3—demonstrate that the choice of ratio has a discernible impact on the mAP (Figure 8). Among these, the 0.2 ratio achieves the optimal balance, yielding an $mAP_{50–95}$ of 0.647, which outperforms both the 0.1 (0.603) and 0.3 (0.633) ratios. This optimized ratio effectively reduces the dataset scale and accelerates training speed while ensuring that the model can learn efficiently.

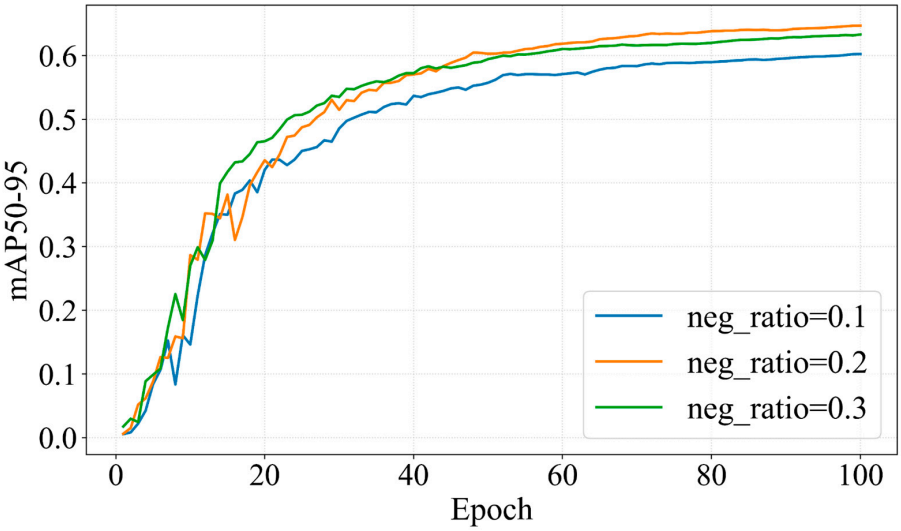


Figure 8. Sensitivity analysis of the negative-to-positive sample ratio on detection performance (mAP).

3.3.2. DBSCAN Parameters

In the DBSCAN clustering algorithm, the value of MinPts should be greater than or equal to the data dimension D + 1. The clustering is based on the 2D spatial coordinates (x, y) of the detection box center points, so D = 2. Based on this, we chose MinPts = 3 as the starting point for clustering.

The other parameter, eps (neighborhood radius), is critical in determining the size and number of clusters. The purpose of clustering in this step is not to achieve semantic segmentation of targets (e.g., distinguishing different object classes) but rather to serve as an intermediate step for generating the minimum number of covering slices. Therefore, conventional methods (like k-distance) cannot be used to determine the value of eps.

An eps value that is too small will cause the detected points to be segmented into a large number of scattered, fragmented clusters. Since each cluster generates an independent ROI, and each ROI (regardless of proximity) produces an independent slice, this leads to a large number of highly overlapping and redundant slices, causing severe computational waste (Figure 9).

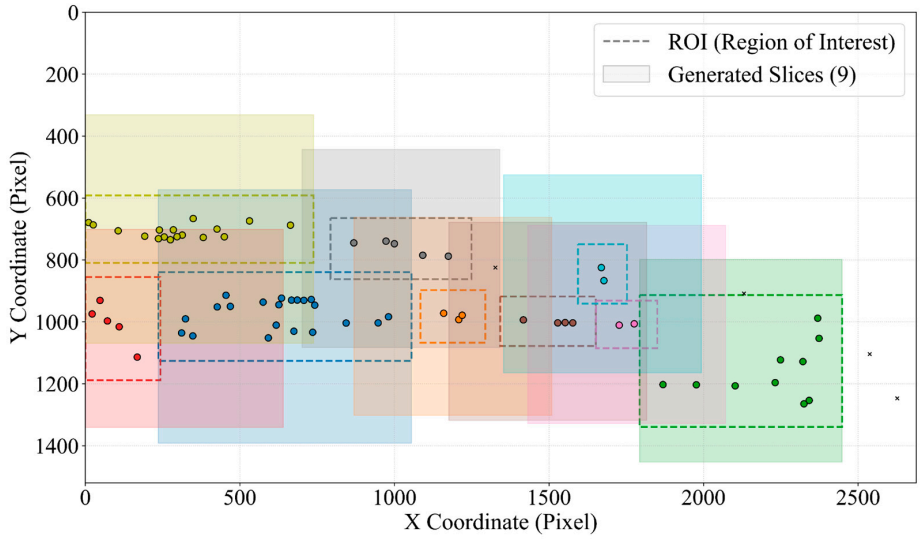


Figure 9. Clustering with an excessively small eps value (ε = 150) results in fragmented ROIs and redundant slicing (9 slices in total). Different colors are used to distinguish the clustered ROIs and slices for visual clarity. The grey dashed line and gray areas in the legend represents ROI (Region of Interest) and Generated Slices for illustrative purposes.

An eps value that is too large will cause multiple target clusters that should be spatially separate to be excessively merged into a single, massive ROI. In our slicing strategy, once an ROI's size exceeds the preset Expand Threshold, it triggers the least efficient "Grid Slice" strategy, which likewise leads to a surge in the number of slices (Figure 10).

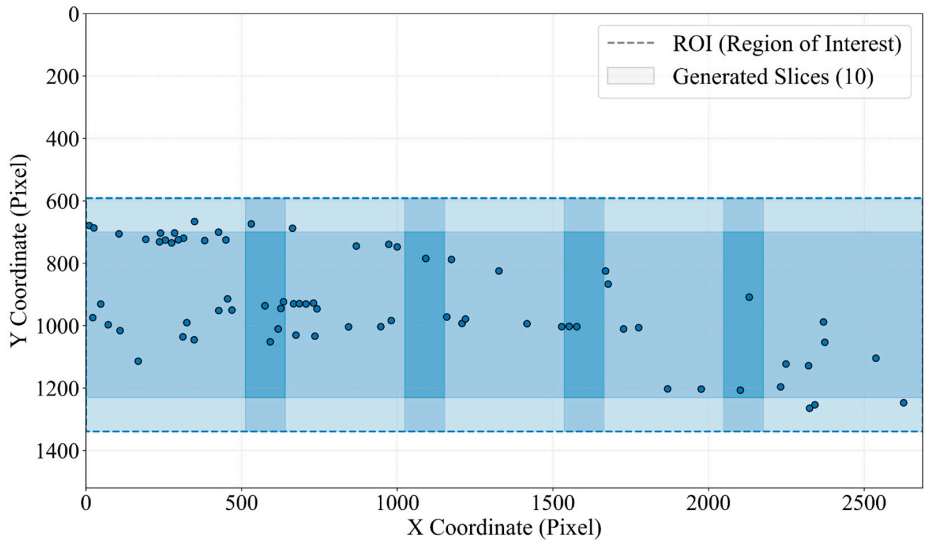


Figure 10. Clustering with an excessively large eps value (ε = 250) causes over-merged ROIs and oversized slicing regions (10 slices in total). Different colors are used to distinguish the clustered ROIs and slices for visual clarity. The grey dashed line and gray areas in the legend represents ROI (Region of Interest) and Generated Slices for illustrative purposes.

Only by finding a suitable eps value can a balance be struck between "over-splitting" and "over-merging." This ensures the ROIs generated by clustering both fully cover all targets and remain spatially compact and separate. This, in turn, allows the subsequent slicing module to preferentially use the lowest-cost Tier 1 or Tier 2 strategies, covering all ROIs with a minimal number of slices and thus minimizing overall computational overhead and maximizing detection efficiency (Figure 11).

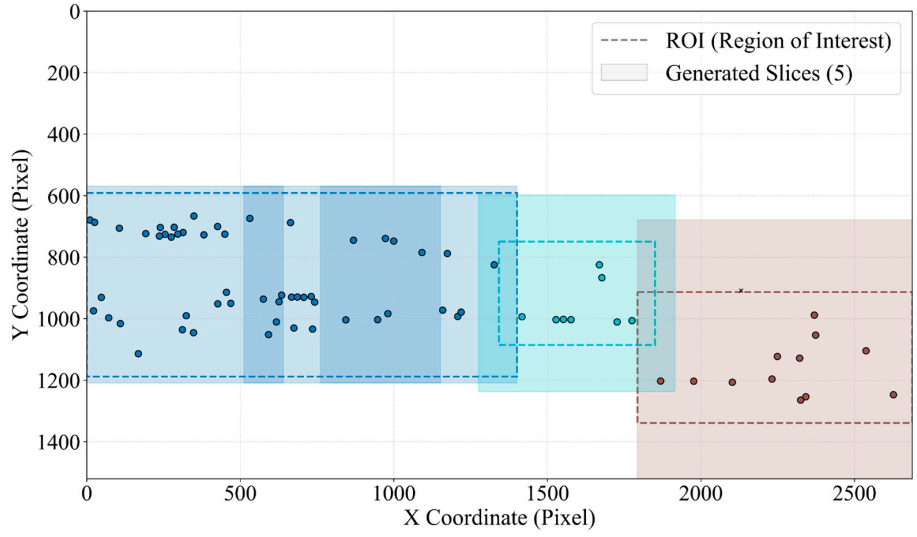


Figure 11. Clustering with an optimal eps value (ε = 170) achieves balanced ROI generation and minimal slicing cost (5 slices in total). Different colors are used to distinguish the clustered ROIs and slices for visual clarity. The grey dashed line and gray areas in the legend represents ROI (Region of Interest) and Generated Slices for illustrative purposes.

We tested a set of eps values (140, 210) on the dataset, using the total number of slices generated by the CAMS-AI method as the evaluation metric. The test results are shown in Figure 12. When eps equals 170, the system's total slice count reaches its minimum. This value is representative for the dataset. Therefore, we selected eps = 170 as the fixed hyperparameter for all subsequent experiments.

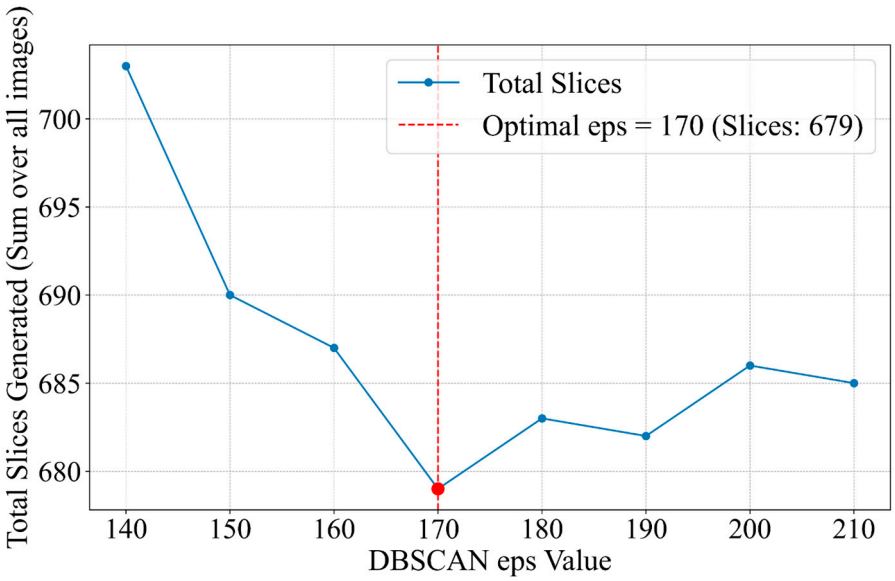


Figure 12. Total slice count versus eps value.

To further validate the rationality of the selected neighborhood radius, we conducted sensitivity experiments to evaluate the impact of different eps values on both detection accuracy and inference efficiency. As summarized in Table 7, the results show that varying eps leads to a clear trade-off between accuracy and computational cost. Among the tested settings, eps = 170 yields a favorable balance, achieving competitive detection accuracy (mAP_{50} = 0.810, $mAP_{50–95}$ = 0.622) while maintaining the lowest average processing time per image (0.0711 s). Based on these observations, eps = 170 was adopted as a stable and efficient configuration for subsequent experiments.

Table 7. Performance evaluation of the CAMS-AI framework under different neighborhood radius (eps) values. The best results are highlighted in bold.

Eps	mAP_{50}	$mAP_{50–95}$	Average Time per Image(s)
140	0.793	0.612	0.0746
150	0.792	0.611	0.0736
160	0.795	0.614	0.0749
170	**0.810**	**0.622**	**0.0711**
180	0.804	0.620	0.0736
190	0.799	0.617	0.0720
200	0.799	0.619	0.0712
210	0.800	0.618	0.0753

3.3.3. Expand Threshold Parameters

In the multi-level slicing strategy, to test for the optimal Expand Threshold, this study tested the impact of different slice size parameters (640, 720, 800, 960) on accuracy, using a detection model trained with a 640-pixel input size. The results are shown in Figure 13.

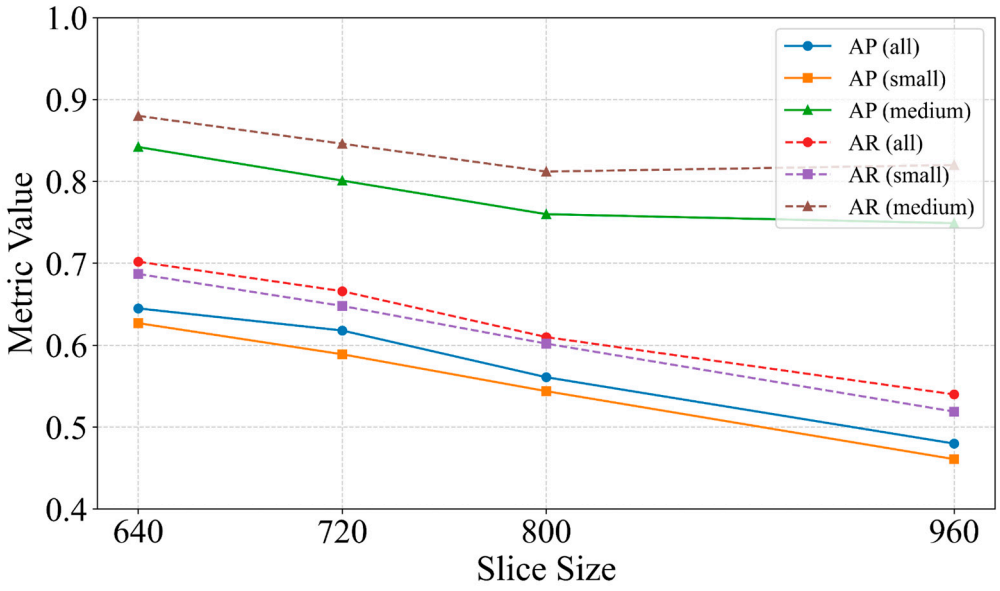


Figure 13. Effect of SAHI Slice on COCO Metrics.

The test results show that as the slice size increases, the model's mean Average Precision (AP) and mean Average Recall (AR) both show a downward trend. Using 640 pixels as the baseline, the mAP for 720 pixels dropped by only 4.19%, whereas the mAP for 800 and 960 pixels both dropped by over 13%. Based on this, a relatively suitable Expand Threshold value was chosen: 720 pixels. It impacts the metrics by less than 5% but effectively solves the redundant slicing problem.

3.4. Experimental Results

To comprehensively evaluate the effectiveness of our proposed CAMS-AI framework, we selected four representative, mainstream object detection models as base detectors: Faster-RCNN, YOLOv8n, YOLOv11n, and RT-DETR.

This study compared three methods: training on original resolution (Baseline), the SAHI framework, and the CAMS-AI framework. All three methods used the same base detectors and identical hyperparameter settings to ensure a fair comparison. All experiments were conducted on the same hardware platform, and the previously defined End-to-End FPS was used as the sole efficiency standard to truly reflect their performance in practical applications. The comparison results are shown in Table 8.

Table 8. Performance comparison of different methods on various models.

Model	Method	mAP_{50}	$mAP_{50–95}$	mAP_{small}	AR_{100}	End-to-End FPS
Faster-RCNN	Baseline	0.061	0.025	0.007	0.031	27.23
	SAHI	0.824	0.59	0.572	0.657	1.48
	CAMS-AI	0.781	0.559	0.547	0.619	12.44
YOLOv8n	Baseline	0.064	0.029	0.016	0.036	31.31
	SAHI	0.794	0.602	0.59	0.643	1.53
	CAMS-AI	0.75	0.57	0.552	0.605	15.63
YOLOv11n	Baseline	0.052	0.025	0.016	0.031	26.76
	SAHI	0.846	0.61	0.594	0.65	1.51
	CAMS-AI	0.763	0.571	0.557	0.623	14.97
RT-DETR	Baseline	0.211	0.139	0.125	0.180	26.5
	SAHI	0.853	0.645	0.629	0.694	1.43
	CAMS-AI	0.810	0.622	0.609	0.674	14.69

From the various accuracy metrics, the Baseline model, trained directly on the original resolution dataset, has accuracy and recall metrics that are essentially unusable, with poor recognition results, even though its End-to-End FPS can reach 30.

The SAHI method demonstrates a slight leading edge over CAMS-AI in accuracy across all four base models. For example, when using the RT-DETR model, SAHI's $mAP_{50–95}$ reached 0.645, while CAMS-AI was 0.622, a gap of only 2.3 percentage points. Similarly, on the critical small-target metric mAP_{small}, SAHI was 0.629 and CAMS-AI was 0.609, a gap of only 2 percentage points. Other models (Faster-RCNN, YOLOv8n, YOLOv11n) also showed similar trends.

In Figure 14, we show the performance of the three methods on the "PrairieLivestock" dataset. Due to the large image size, only partial cropped regions are displayed. As seen in sample groups (b) and (c), the results of SAHI and CAMS-AI are nearly identical. Only in sample (a) is SAHI slightly better than our method. This minor accuracy difference is expected.

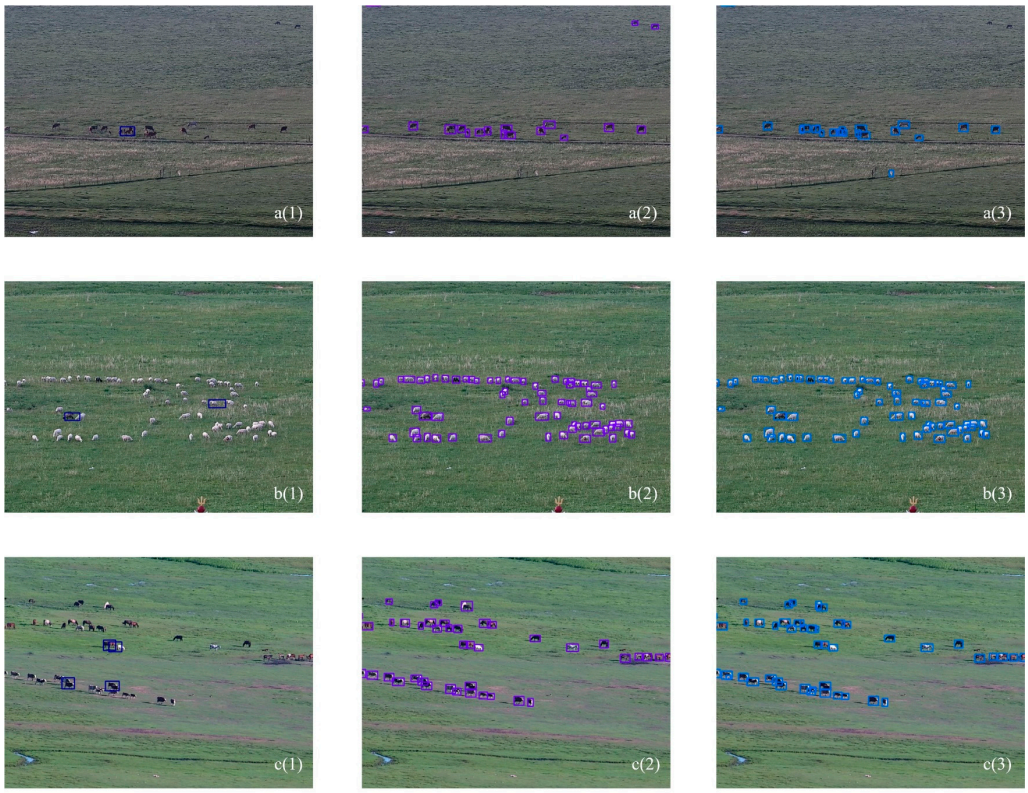


Figure 14. Visualization of detection results. (**a1**–**a3**,**b1**–**b3**,**c1**–**c3**) correspond to three different input images. For each image, the detection results of Baseline, SAHI, and CAMS-AI are shown from left to right. Different colors of the bounding boxes indicate detection results produced by different methods.

End-to-End FPS is the key to measuring a method's practicality, and this is precisely where the CAMS-AI framework demonstrates its core advantage. When using RT-DETR, CAMS-AI's End-to-End FPS reached 14.69, while SAHI's was only 1.43, representing a 10.3-fold increase in end-to-end processing speed. Other models also showed speed increases ranging from 8 to 10 times.

4. Discussion

In wide-area, sparse-target scenarios, the efficiency bottleneck of slicing methods like SAHI stems primarily from the redundant computation performed on a massive number of pure background (negative sample) slices. The CAMS-AI framework, through its "coarse-to-fine" strategy, achieves a trade-off: in exchange for an acceptable accuracy drop (approximately 2–4 percentage points in $mAP_{50–95}$), it gains up to a 10.3-fold increase in end-to-end inference speed (using RT-DETR as an example).

Our work resides in the same research domain as SAHI and ASAHI but addresses a different problem. SAHI solves the issue of small targets being "invisible" (due to resizing) by using slicing, which dramatically improves recall and accuracy. However, as noted by Zhorif et al. [23] and confirmed in our study, it sacrifices efficiency. ASAHI [25] attempts to improve efficiency by adaptively adjusting slice sizes to reduce the slice count, but it still needs to process the entire image, failing to solve the computational waste on pure background regions. CAMS-AI, by using RPN + DBSCAN clustering, filters out a large number of negative sample regions before the fine-grained detection (Fine Model) stage. This allows computational resources to be focused on high-value ROIs. Therefore, CAMS-AI is not a simple replacement for SAHI or ASAHI, but rather a more intelligent, two-stage slicing paradigm better suited for scenarios with sparse target distributions.

CAMS-AI exhibits a slight accuracy drop compared to SAHI across all metrics (e.g., a 2.3 percentage point drop with RT-DETR). This difference likely stems from the RPN stage failing to recall (or the clustering stage failing to include) some extremely isolated, peripheral targets. SAHI's "global uniform slicing" mechanism, while inefficient, theoretically guarantees complete coverage of the image, giving it a slight advantage in recalling such outliers.

The significance of this research extends beyond grassland livestock monitoring. Building upon the slicing paradigm established by SAHI, the CAMS-AI framework provides a practical and feasible solution for all fields facing the challenge of "high-resolution, large field-of-view, sparse targets." This includes applications such as specific feature detection (e.g., vehicles, ships) in satellite remote sensing, or locating minute defects (e.g., cracks, rust) during Unmanned Aerial Vehicle (UAV) inspections.

Currently, an RPN is employed to implement the coarse detection stage. It should be noted that the RPN is not designed as a lightweight detector in terms of FLOPs or parameter count. Instead, its role is to provide a high-recall, low-frequency global scan that is executed only once per image. As a result, its computational cost does not constitute the primary bottleneck of the overall pipeline. The dominant computation arises from the fine detection stage applied to multiple sliced ROIs. From a system-level perspective, CAMS-AI reduces redundant computation by minimizing where fine-grained detection is applied, rather than by aggressively simplifying the coarse network itself.

Despite its advantages in clustered scenarios, the CAMS-AI framework may exhibit limitations in environments characterized by extreme target dispersion. The intelligent focusing strategy relies on the DBSCAN algorithm to aggregate discrete proposals into cohesive Regions of Interest (ROIs). If targets are distributed so sparsely that they fail to satisfy the density criteria—specifically, the minimum points (MinPts) within a given neighborhood radius (eps)—the clustering module will be unable to generate valid ROIs. Future optimizations could explore more flexible region proposal mechanisms or dynamic density thresholds to maintain high-precision detection across varying target distributions.

5. Conclusions

To address the severe inefficiency caused by slicing methods like SAHI in high-resolution small object detection—a result of processing massive amounts of redundant

background—this paper proposes an efficient detection framework named CAMS-AI. CAMS-AI adopts a "coarse-to-fine" intelligent focusing strategy. The framework utilizes a lightweight Region Proposal Network (RPN) to rapidly lock onto Regions of Interest (ROIs) that contain targets, and then employs an adaptive multi-level slicing strategy for efficient, fine-grained detection.

Extensive experiments on our self-built "PrairieLivestock" dataset from Inner Mongolian grasslands demonstrated the effectiveness of the CAMS-AI framework. Compared to the standard SAHI method, when using various mainstream detectors (Faster-RCNN, YOLOv8n, YOLOv11n, RT-DETR) as baselines, CAMS-AI achieved an 8 to 10-fold+ increase in end-to-end inference speed while sacrificing only a minor, acceptable amount of detection accuracy (an approximate 2–4 percentage point drop in $mAP_{50–95}$).

By filtering out a large number of useless background regions, CAMS-AI precisely focuses computational resources on valuable information. It successfully strikes a better balance between detection accuracy and inference efficiency than existing methods, providing a solution that delivers both precision and speed for object detection in high-resolution, wide-area, sparse-target scenarios.

Author Contributions: Z.C. (Zhanqi Chen) was responsible for conducting the experiments and drafting the manuscript. Q.G. was responsible for data collection, and both Z.C. (Zhanqi Chen) and Q.G. jointly completed the dataset annotation. B.Y. was responsible for data processing. H.W. conducted the literature review and organization. X.Z. prepared part of the figures. Z.C. (Zhao Chen) supervised the overall work and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by China Tower Intelligent Connectivity 2024 Forest Intelligent Protection Phase II—Forage-Livestock Balance Model R&D Service Procurement Project, grant number TZC-ZBZB-2024-000030.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: We would like to thank Yingran Che for their contributions to data collection. Lingnan Dai and Lishuo Huo provided valuable suggestions for the revised manuscript.

Conflicts of Interest: Author Baohui Yang was employed by the company Beijing Evialab Technology Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Fraser, M.D.; Vallin, H.E.; Roberts, B.P. Animal Board Invited Review: Grassland-Based Livestock Farming and Biodiversity. *Animal* **2022**, *16*, 100671. [CrossRef]
2. Stevens, N.; Bond, W.; Feurdean, A.; Lehmann, C.E.R. Grassy Ecosystems in the Anthropocene. *Annu. Rev. Environ. Resour.* **2022**, *47*, 261–289. [CrossRef]
3. Piipponen, J.; Jalava, M.; De Leeuw, J.; Rizayeva, A.; Godde, C.; Cramer, G.; Herrero, M.; Kummu, M. Global Trends in Grassland Carrying Capacity and Relative Stocking Density of Livestock. *Glob. Change Biol.* **2022**, *28*, 3902–3919. [CrossRef]
4. Adam, M.; Song, J.; Yu, W.; Li, Q. Deep Learning Approaches for Automatic Livestock Detection in UAV Imagery: State-of-the-Art and Future Directions. *Future Internet* **2025**, *17*, 431. [CrossRef]
5. Chen, A.; Jacob, M.; Shoshani, G.; Charter, M. Using Computer Vision, Image Analysis and UAVs for the Automatic Recognition and Counting of Common Cranes (Grus Grus). *J. Environ. Manag.* **2023**, *328*, 116948. [CrossRef]
6. Fang, C.; Li, C.; Yang, P.; Kong, S.; Han, Y.; Huang, X.; Niu, J. Enhancing Livestock Detection: An Efficient Model Based on YOLOv8. *Appl. Sci.* **2024**, *14*, 4809. [CrossRef]
7. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 142–158. [CrossRef]
8. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.

9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2016; Volume 9905, pp. 21–37. ISBN 978-3-319-46447-3.
12. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. In *Computer Vision–ECCV 2020*; Springer: Cham, Switzerland, 2020.
13. Zhao, Y.; Lv, W.; Xu, S.; Wei, J.; Wang, G.; Dang, Q.; Liu, Y.; Chen, J. DETRs Beat YOLOs on Real-Time Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 16–22 June 2024; pp. 16965–16974.
14. Jiang, L.; Yuan, B.; Du, J.; Chen, B.; Xie, H.; Tian, J.; Yuan, Z. MFFSODNet: Multiscale Feature Fusion Small Object Detection Network for UAV Aerial Images. *IEEE Trans. Instrum. Meas.* **2024**, *73*, 5015214. [CrossRef]
15. Cheng, G.; Yuan, X.; Yao, X.; Yan, K.; Zeng, Q.; Xie, X.; Han, J. Towards Large-Scale Small Object Detection: Survey and Benchmarks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 13467–13488. [CrossRef]
16. Wei, W.; Cheng, Y.; He, J.; Zhu, X. A Review of Small Object Detection Based on Deep Learning. *Neural Comput. Appl.* **2024**, *36*, 6283–6303. [CrossRef]
17. Nikouei, M.; Baroutian, B.; Nabavi, S.; Taraghi, F.; Aghaei, A.; Moghaddam, M.E. Small Object Detection: A Comprehensive Survey on Challenges, Techniques and Real-World Applications. *arXiv* **2025**, arXiv:2503.20516. [CrossRef]
18. Biggs, D.R.; Theart, R.P.; Schreve, K. Sub-Window Inference: A Novel Approach for Improved Sheep Counting in High-Density Aerial Images. *Comput. Electron. Agric.* **2024**, *225*, 109271. [CrossRef]
19. Akyon, F.C.; Onur Altinuc, S.; Temizel, A. Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection. In Proceedings of the 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 16–19 October 2022; pp. 966–970.
20. Mazen, F.M.A.; Shaker, Y. Small Object Detection in Complex Images: Evaluation of Faster R-CNN and Slicing Aided Hyper Inference. *Int. J. Adv. Comput. Sci. Appl.* **2025**, *16*, 951–960. [CrossRef]
21. Fotouhi, F.; Menke, K.; Prestholt, A.; Gupta, A.; Carroll, M.E.; Yang, H.-J.; Skidmore, E.J.; O'Neal, M.; Merchant, N.; Das, S.K.; et al. Persistent Monitoring of Insect-Pests on Sticky Traps through Hierarchical Transfer Learning and Slicing-Aided Hyper Inference. *Front. Plant Sci.* **2024**, *15*, 1484587. [CrossRef]
22. Pereira, A.; Santos, C.; Aguiar, M.; Welfer, D.; Dias, M.; Ribeiro, M. Improved Detection of Fundus Lesions Using YOLOR-CSP Architecture and Slicing Aided Hyper Inference. *IEEE Lat. Am. Trans.* **2023**, *21*, 806–813. [CrossRef]
23. Zhorif, N.N.; Anandyto, R.K.; Rusyadi, A.U.; Irwansyah, E. Implementation of Slicing Aided Hyper Inference (SAHI) in YOLOv8 to Counting Oil Palm Trees Using High-Resolution Aerial Imagery Data. *Int. J. Adv. Comput. Sci. Appl.* **2024**, *15*, 869–874. [CrossRef]
24. Badgujar, C.M.; Poulose, A.; Gan, H. Agricultural Object Detection with You Only Look Once (YOLO) Algorithm: A Bibliometric and Systematic Literature Review. *Comput. Electron. Agric.* **2024**, *223*, 109090. [CrossRef]
25. Zhang, H.; Hao, C.; Song, W.; Jiang, B.; Li, B. Adaptive Slicing-Aided Hyper Inference for Small Object Detection in High-Resolution Remote Sensing Images. *Remote Sens.* **2023**, *15*, 1249. [CrossRef]
26. Nguyen, N.-D.; Do, T.; Ngo, T.D.; Le, D.-D. An Evaluation of Deep Learning Methods for Small Object Detection. *J. Electr. Comput. Eng.* **2020**, *2020*, 3189691. [CrossRef]
27. Tong, K.; Wu, Y. Deep Learning-Based Detection from the Perspective of Small or Tiny Objects: A Survey. *Image Vis. Comput.* **2022**, *123*, 104471. [CrossRef]
28. Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv* **2019**, arXiv:1906.07155. [CrossRef]
29. Ismail, Z.H.; Chun, A.K.K.; Shapiai Razak, M.I. Efficient Herd–Outlier Detection in Livestock Monitoring System Based on Density–Based Spatial Clustering. *IEEE Access* **2019**, *7*, 175062–175070. [CrossRef]
30. Graziani, M.; Lompech, T.; Müller, H.; Depeursinge, A.; Andrearczyk, V. On the Scale Invariance in State of the Art CNNs Trained on ImageNet. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 374–391. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.