Improving Small Object Detection from Drone Images Using SAHI_YOLO

Ubugunov Artem, Chun-Ming Tsai
University of Taipei, University of Taipei
g11016025@go.utaipei.edu.tw, cmtsai@go.utaipei.edu.tw

Abstract

Small object detection is vital across diverse applications, including traffic monitoring and medical diagnostics. However, conventional approaches that rely on attention modules integrated into neural networks are often labor-intensive, time-consuming, and computationally expensive. To address these challenges, we introduce the Slicing Aided Hyper Inference (SAHI)_YOLO method, an efficient and accurate solution for small object detection. SAHI operates by dividing an image into smaller segments (slices) and performing detection on each, optimizing resource use while maintaining high performance. In our experiments, we trained three YOLO variants (v9, v10, v11) and integrated them with SAHI, comparing the performance of all six resulting models. Our top-performing model achieved a 23% increase in precision, a 38% increase in recall, and a 66% improvement in mAP50 compared to baseline models. Evaluated on the VisDrone-2019-DET dataset, these results underscore the robustness and effectiveness of our approach. SAHI_YOLO offers a powerful and practical advancement for small object detection tasks.

Keywords: Slicing Aided Hyper Inference, SAHI_YOLO, Small Object Detection, VisDrone

Introduction

In recent years, object detection has become one of the fundamental tasks in computer vision, increasingly penetrating our daily lives to solve various problems, from face identification to autonomous driving. Modern hybrids and multi-level approaches, such as Feature Pyramid Networks (FPN), the latest versions of YOLO (YOLOv10, YOLOv11), and Vision Transformers (ViTs), actively address challenges related to the low resolution of small objects, loss of details during processing, and scaling difficulties.

However, in these models, the detection of small objects requires improvement. Additionally, high computational costs and the difficulty of integrating attention modules into the network remain critical challenges.

F.C. Akyon et al.[10] proposed Slicing Aided Hyper Inference (SAHI), similar to our approach, to reduce the computational cost of small object detection. However, they used SAHI with the outdated YOLOv5 model. Additionally, they used an image size of 640×440, which is significantly tiny for such tasks. Akhil Meethal et al.[3] also addressed the challenges of reducing computational costs and improving usability. Their proposed Cascaded Zoom-in (CZ) Detector demonstrates impressive performance, achieving a mAP50 of 58.3%. However, for large objects, this method performs weaker.

Thus, this study aims to identify an optimal approach for small object detection that minimizes additional computational costs and maintains usability. This study can help researchers struggling with small object detection save time finding the optimal module. Additionally, the police can adopt this technology to catch criminals. At the same time, this technology can be highly useful for city governments in optimizing traffic flow and for journalists who can use this technology to count the number of participants in demonstrations.

This paper proposes the Slicing Aided Hyper Inference (SAHI)_YOLO object detection method, which performs inference on already pre-trained models, thereby ensuring high accuracy and a reduced computational burden.

Related Works

2.1 Object Detection in Aerial Images

Y. W. Chien et al.[1] proposed YOLOv9, a powerful computer vision model for object detection. It introduces the YOLOv9 and GELAN architectures, accessible through the accompanying Python repository. This solution fundamentally relies on two groundbreaking innovations: Programmable Gradient Information (PGI) and the lightweight Generalized Efficient Layer Aggregation Network (GELAN) architecture. It allows the model to retain complete information during forward propagation and generate a reliable gradient flow, improving its stability and accuracy during training.[11] A. Wang et al.[12] introduced YOLOv10, a new approach to real-time object detection. Unlike previous YOLO versions, this model eliminates the need for non-maximum suppression (NMS) and optimizes key architectural components. This significantly reduces computational costs while maintaining high accuracy. Experiments confirm that YOLOv10 outperforms its counterparts regarding accuracy-speed balance across different model scales. R. Khanam, M. Hussain, et al.[13] provided an overview of the main principles behind the operation of YOLOv11. The key innovation of this model is enhancing SPPF, C2PSA, and C3K2 modules for image processing. They support various computer vision tasks, such as object detection,

instance segmentation, pose estimation, oriented object detection, and object tracking. Akhil Meethal et al.[4] dedicated their work to the problem of semi-supervised object detection (SSOD) in aerial images, where data labeling requires significant effort, especially for small objects. The authors propose a Density Crop-guided Semi-supervised Detector that uses density crops to improve detection accuracy. They used VisDrone and DOTA datasets, which contain 8,599 with 540k annotated instances and 1,869 images with 280k annotated cases, respectively. As for the VisDrone dataset, the resolution is about 2000 ×1500 pixels; DOTA has a resolution ranging from 800×800 to 4000×4000. On VisDrone detector displays, APs increased by 6%, and AP (overall accuracy) increased by 8% compared to QueryDet. As for DOTA, APs increased by 3%, and overall AP by 2% compared to the base mean-teacher. Compared to QueryDet, the method handles small objects better and does not require special modules. It achieves a better balance between accuracy and computational complexity.

George Adaimi et al.[5] proposed Butterfly Detector, a new object detection method that solves problems of scaling, object orientation, and partial occlusions. During the experiment, the average number of false positives decreased by 30% compared to CenterNet. However, vector prediction errors can lead to inaccurate bounding boxes. Ziyang Tang et al. [6] proposed PENet to detect objects of various sizes from high-resolution and imbalanced datasets. They presented a re-sampling algorithm MRM, a cluster-generating algorithm NMM, and a hierarchical loss approach. Each of them can be independently applied to CNN-based detectors. However, in some cases, an object may be cropped into two parts in different clusters and have multiple bounding boxes. Merging the two bounding boxes into a larger box, as they proposed, may take more time. Zhiwei Wei et al.[7] proposed AMRNet network is an abbreviation for the three methods they presented: adaptive cropping, mosaic augmentation, and mask resampling. Their scale adaptive cropping method relieves scale variation problems in the training stage, and the withal mask resampling method relieves the class imbalance problem. However, their mask resampling method increases only 0.2 points when it unites with mosaic augmentation. Onur Can Koyun et al.[8] proposed Focus-And-Detect framework for small object detection in aerial images based on region search. Their method uses the Gaussian mixture model to generate object clusters, where the generated clusters are scale normalized. They also proposed the 'Incomplete Box Suppression' (IBS) approach to suppress incomplete boxes caused by overlapping focal regions. Their proposed method achieves a 42.06 AP score on the VisDrone validation set and a 54.16 AP@70 score on the UAVDT test set.

2.2 Small Object Detection

Xiaohui Guo[9] introduced a Multi to Single Module for Small Object Detection. The principle of the work is as follows: first, the five-level features from the backbone network are aggregated into three-level features. Then, the three-level features with different characteristics are utilized to enhance the low-level features of the neck network. This method improves the perspective of a single layer of features and demonstrates superior performance for small object detection. However, for larger-sized object detection, his method performs similarly to Yolov5s. Shiyi Tang et al.[11] improved YOLOv5 for small object detection. Their method improved mAP@[.5:.95] by 6.42% and mAP@0.5 by 9.38% onVisDrone-2019-DET dataset. However, it is rather strange to improve YOLOv5 in 2023, given that YOLOv8 and YOLOv9 had already been introduced, along with many more efficient modules.

Proposed Method

3.1 The proposed system diagram

As shown in Figure 1, in the "VisDrone trainset and valet" sections, we uploaded our prepared VisDrone dataset (see Implementation Details for further information). In the three sections titled "Training YOLOv9, v10, v11", we illustrate using the YOLOv9, YOLOv10, and YOLOv11 models for training the dataset. The next three ovals, called "best.pt" represent the process following the completion of training, where the 'best.pt' files of the trained models were utilized for testing under YOLO's default settings. This testing phase evaluated the performance metrics of all three models. Subsequently, as indicated by the "YOLOv9, v10, v11 + SAHI detection" sections, we implemented these models using SAHI (Slice-Aided Hyper Inference). During implementation, we utilized the 'best.pt' files generated after training the YOLO models and the "VisDrone testset-dev" dataset to perform inference on the dataset images. The last three sections, titled "YOLOv9, v10, v11 + SAHI detection results," represent the Precision, Recall, mAP50, and mAP50-95 results of each model obtained after implementation.

3.2 SAHI

SAHI(Slicing Aided Hyper Inference)[14] is an innovative library that optimizes object detection algorithms for large-scale and high-resolution imagery. Its core functionality lies in partitioning images into manageable slices, running object detection on each slice, and then stitching the results together. SAHI is compatible with a range of object detection models, including the YOLO series, offering flexibility while ensuring optimized use of computational resources. SAHI has three key features that provide high performance in detecting small objects:

(1) Seamless Integration: SAHI integrates effortlessly with YOLO models, meaning you can

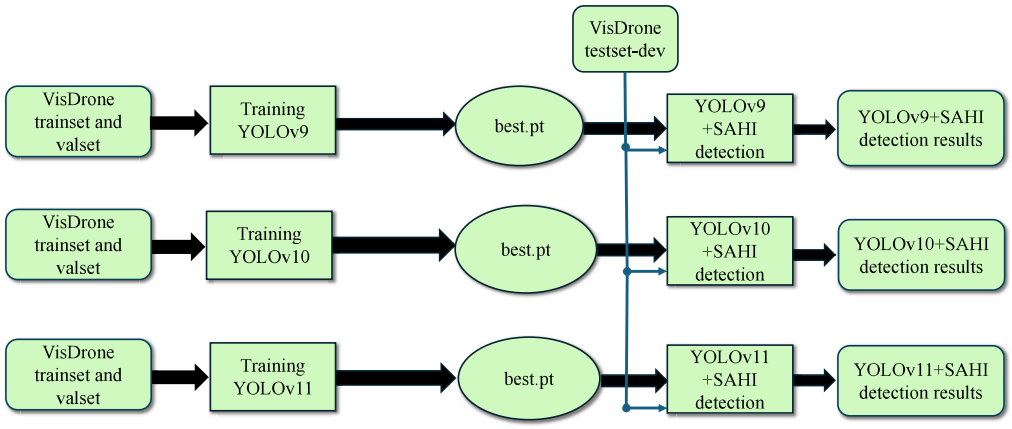

Figure 1. The proposed system diagram

start slicing and detecting without much code modification.

(2) Resource Efficiency: By breaking down large images into smaller parts, SAHI optimizes memory usage, allowing you to run high-quality detection on hardware with limited resources.

(3) High Accuracy: SAHI maintains detection accuracy by employing smart algorithms to merge overlapping detection boxes during stitching.

3.3 YOLOv9+SAHI

YOLOv9+SAHI enhances the model through a combined approach: YOLOv9 incorporates an improved architecture with multi-scale feature fusion, adaptive gradient normalization, and optimized convolutions, enhancing detection accuracy and stability. SAHI (Slicing Aided Hyper Inference) complements YOLOv9 by dividing large images into slices, improving small object detection, reducing false negative detections, and making inference more efficient. These methods ensure high accuracy (mAP), stable gradient flow, and superior performance on high-resolution images.

3.4 YOLOv10+SAHI

YOLOv10+SAHI enhances the model through an advanced YOLOv10 architecture, incorporating hybrid convolutional mechanisms, adaptive gradient normalization, dynamic feature fusion, and an optimized transformer block significantly improve accuracy, especially in complex scenes. SAHI (Slicing Aided Hyper Inference) complements YOLOv10 by dividing large images into slices, enhancing small object detection, and reducing computational costs. Together, these methods provide higher accuracy (mAP), more efficient memory usage, and better adaptation to objects of varying scales.

3.5 YOLOv11+SAHI

YOLOv11+SAHI enhances the model through a new YOLOv11 architecture, incorporating Transformer-enhanced convolutions, Hybrid Scalable Feature Pyramids, and adaptive gradient normalization, which improve accuracy, robustness in complex scenes, and small object detection efficiency. SAHI (Slicing Aided Hyper Inference) complements YOLOv11 by dividing images into slices, enhancing small object detection, reducing missed detections, and lowering computational costs. These methods deliver maximum accuracy (mAP), resilience in challenging conditions, and high performance on high-resolution images.

Experimental Results and Discussion

We conducted experiments to assess the performance of the proposed detection algorithm using the VisDrone-2019-DET dataset. The training and detecting processes were carried out on an Ubuntu 24.04 LTS operating system, utilizing MSI Geforce RTX 4060 Ti (16GB) GPUs, the Nvidia 550.120 driver, CUDA 12.4, and Python 3.12.4.

4.1 Dataset

Our research used the VisDrone2019 dataset[15], specifically “Task 1: Object Detection in Images.” This dataset includes 6,471 training images in the trainset folder, 548 validation images in the valset folder, and 1,610 testing images in the testdev folder.

4.2 Implementation details

Before starting training, we need to structure our dataset. The root folder is a VisDrone2YOLO-main, containing the train, val, and test subfolders. These subfolders should include images in .jpg format and YOLO labels in .txt format.

Next, we need to set the path and category details of the dataset. You can refer to the coco.yaml format in the datafolder. Pay close attention to the train, val, and test items—our image paths must be correctly specified next to them. Then, select the number of categories in the dataset and add the corresponding category names in the fields below. In our case, ‘nc’ is set to 10, and the class names are as follows: pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, and motor.

We used their ‘best.pt’ files after training the YOLOv9, YOLOv10, and YOLOv11 models files

for inference with SAHI by entering the following command in the terminal:

sahi predict --source VisDrone2YOLO-main/VisDrone2019-DET-test-dev/images --model_path best.pt --dataset_json_path labels_coco.json --model_type ultralytics --no_standard_prediction --slice_width 1024 --slice_height 1024 --overlap_height_ratio 0.2 --overlap_width_ratio 0.2 --model_confidence_threshold 0.5 --postprocess_match_metric IOU --postprocess_match_threshold 0.65

SAHI does not support the YOLO annotation format (txt), so we used the Yolo-to-COCO-format-converter [16] to convert text labels into a COCO-format JSON file. SAHI has two modes: sliced prediction and no sliced prediction. We chose no sliced prediction to achieve higher accuracy. We also kept the default settings, such as --slice_width 1024 and --slice_height 1024, as they provide better accuracy.

4.3 Results

The experimental results in Table 1 indicate the performance of different YOLO models (YOLOv9, YOLOv10, and YOLOv11) with and without Slicing Aided Hyper Inference (SAHI) for small object detection. The evaluation was based on Precision (P), Recall (R), mean Average Precision at IoU 0.5 (mAP50), and mean Average Precision across IoUs from 0.5 to 0.95 (mAP50-95).

The comparison between YOLOv9+SAHI and YOLOv9 indicates that YOLOv9+SAHI's Precision, mAP@.5, and mAP@.5:.95 results across all classes are nearly 1.5 times higher than those of YOLOv9. Meanwhile, the Recall result surpasses that of the YOLOv9 model by more than 2.5 times.

A similar situation is observed when comparing YOLOv10 and YOLOv10+SAHI. The Precision, mAP@.5, and mAP@.5:.95 results across all classes in our proposed method are nearly 1.5 times higher than those of YOLOv10. Meanwhile, the Recall result surpasses that of the YOLOv10 model by more than 2 times.

YOLOv11+SAHI also outperforms YOLOv11 in Precision, mAP@.5, and mAP@.5:.95 results by nearly 1.5 times. Its Recall surpasses that of YOLOv11 by more than 2 times. Additionally, it is worth noting that YOLOv11+SAHI achieves the best results, with Precision, Recall, mAP@.5, and mAP@.5:.95 not only outperforming the YOLOv9–YOLOv11 models but also ranking at the top among all YOLO+SAHI models.

As shown in Figure 8, YOLOv11 fails to detect a pedestrian on the left side near the house. However, as shown in Figure 9, YOLOv11+SAHI successfully detects this individual. The following detection image is Figure 10, which demonstrates that YOLOv11 doesn't detect three cars and bicycles far ahead. As shown in Figure 11, YOLOv11+SAHI detected many more objects than YOLOv11.

4.4 Discussion

According to the experimental results, YOLOv9-11+SAHI outperforms YOLOv9-11 in Precision, Recall, and mAP@.5 across all classes without incorporating additional models. At the same time, YOLOv11 achieves the best results among all six models. However, the mAP@.5:.95 for each class in our proposed method is significantly lower than the same metric in YOLOv9-11.

This is because SAHI breaks the image into manageable chunks, performing object detection on each chunk and then stitching the results together [14]. The result exceeded all our expectations; other models have not yet achieved such high detection accuracy. The mAP@.5:.95 in models with SAHI is lower than in models without SAHI because after merging predictions, its quality drops at high IoU values (e.g., 0.95), which can lead to a sharp decrease in mAP50-95 but leaves mAP50 high. This parameter is not critical; increasing the IoU in subsequent inferences can help mitigate this limitation. The comparison results of testing different models using the VisDrone dataset are demonstrated in Table 2.

The table presents different methods evaluated on the test-dev image folder, which includes 1,610 images, with mAP50 as the key metric. As you can see, TOOD+SAHI [10] achieves the lowest performance with an accuracy of 31.9%. TOOD (Task-Aligned One-Stage Object Detection) is an anchor-free detector that focuses on aligning feature representations with detection tasks, while SAHI (Slicing Aided Hyper Inference) is designed to enhance object detection in high-resolution images by slicing them into smaller patches. Density Crop [4] improves detection accuracy to 48.9% by cropping high-density areas, thereby enhancing model focus on regions containing objects. This approach is beneficial when objects are cluttered but may still struggle with scale variations and occlusions. CZ Det. [3] achieves a higher accuracy of 58.3%, indicating its robustness in detecting objects across varying scales and densities. The architecture includes multi-scale feature extraction techniques, contributing to better generalization over different object sizes and locations. Butterfly Detector performs slightly lower than CZ Det. with an accuracy of 54.6%. CZ Det, while still outperforming TOOD+SAHI and Density Crop, highlights certain limitations in handling specific challenges, such as complex backgrounds or overlapping objects. Our proposed method significantly surpasses all other methods, achieving accuracy of 91.6%. This high accuracy demonstrates

Table 1. Full Test Results Output Comparison Table

Total Labels=75102	P	R	mAP@.5	mAP@.5:.95
YOLOv9-E	0.791	0.345	0.56	0.402
YOLOv9+SAHI	0.978	**0.908**	0.888	0.645
YOLOv10-X	0.723	0.378	0.543	0.371
YOLOv10+SAHI	0.979	0.895	0.885	0.656
YOLOv11-X	0.735	0.373	0.546	0.376
YOLOv11+SAHI	**0.980**	0.891	**0.893**	**0.659**

Figure 8. Detection results by YOLOv11

Figure 10. Detection results by YOLOv11

Figure 9. Detection results by YOLOv11+SAHI

Figure 11. Detection results by YOLOv11+SAHI

that the model effectively solves small object detection problem.

During the experiment, the main limitation was the number of images in the VisDrone 2019 dataset. With a more significant number of images, the accuracy of predictions would have been much higher. Another major limitation was the lack of a mechanism for calculating Precision, Recall, mAP50, and mAP50-95 for each class in the SAHI configuration. To solve this problem, we had to write custom code to calculate these metrics.

Our proposed method significantly surpasses all other methods, achieving an accuracy of 91.6%. This high accuracy demonstrates that the model effectively solves small object detection problems.

During the experiment, the main limitation was the number of images in the VisDrone 2019 dataset. With a more significant number of images, the accuracy of predictions would have been much higher. Another major limitation was the lack of a mechanism for calculating Precision, Recall, mAP50, and mAP50-95 for each class in the SAHI configuration. To solve this problem, we had to write custom code to calculate these metrics.

Therefore, in future versions of SAHI, it would be desirable to see a built-in mechanism for converting labels from the YOLO format to the COCO format, as well as an automatic calculation of the necessary metrics for each class.

Table 2. The comparison results of various methods on the VisDrone dataset

Methods	Testing Images	mAP50
CZ Det. [3]	1610	58.3
BD [5]	1610	54.6
Density Crop [4]	1610	48.9
TOOD+SAHI [10]	1610	31.9
Ours	1610	**91.6**

To summarize, despite its limitations, SAHI demonstrates outstanding performance not only compared to baseline models but also in comparison with similar models from other authors. Thus, this study will make a significant contribution to the further exploration of object detection.

Conclusions

This paper proposes the SAHI-YOLOs object detection method for detecting small objects at large distances. Additionally, we provide supplementary code to evaluate SAHI metrics for each class. Experimental results demonstrate that SAHI achieves a 66% mean Average Precision (mAP) increase. The proposed method efficiently detects small objects while reducing computational costs. Furthermore, SAHI outperforms state-of-the-art models in detecting small objects from aerial imagery captured by visual drones. For future work, we aim to develop a novel mechanism for calculating mAP50-95 and adapt this technology for real-time small object detection in video files and streaming videos.

References

[1] Y. W. Chien, Y. I-Hau, and M. L. Hong-Yuan, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," National Taipei University of Technology, Taiwan, 2024.

[2] J. W. Zhang, "Using attention models to improve object detection accuracy in drone-view images", Department of Computer Science, University of Taipei, Taiwan, 2024.

[3] A. Meethal, E. Granger, and M. Pedersoli, "Cascaded Zoom-in Detector for High Resolution A erial Images", arXiv.org. [Online]. Available: https://

arxiv.org/abs/2303.08747
[4] A. Meethal, E. Granger, and M. Pedersoli, “Density Crop-guided Semi-supervised Object Detection in Aerial Images,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2308.05032
[5] G. Adaimi, S. Kreiss, and A. Alahi, “Perceiving Traffic from Aerial Images,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2009.07611
[6] Z. Tang, X. Liu, G. Shen, and B. Yang, “PENet: Object Detection using Points Estimation in Aerial Images,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2001.08247
[7] Z. Wei, C. Duan, X. Song, Y. Tian, and H. Wang, “AMRNet: Chips Augmentation in Aerial Images Object Detection,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2009.07168
[8] O. C. Koyun, R. K. Keser, İ. B. Akkaya, and B. U. Töreyin, “Focus-and-Detect: A small object detection framework for aerial images,” *Signal Processing: Image Communication*, vol. 104, p. 116675, May 2022, doi: 10.1016/j.image.2022.116675.
[9] X. Guo, “A novel Multi to Single Module for small object detection,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2303.14977
[10] F. C. Akyon, S. O. Altinuc, and A. Temizel, “Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2202.06934
[11] S. Tang, S. Zhang, and Y. Fang, “HIC-YOLOv5: Improved YOLOv5 For Small Object Detection,” arXiv.org. [Online]. Available: https://arxiv.org/abs/2309.16393
[11] G.-L. Huang, “YOLOv9 究竟做了什麼，一起來一探究竟 - Gi-Luen Huang,” *Medium*, May 28, 2024. Accessed: Feb. 10, 2025. [Online]. Available: https://come880412.medium.com/YOLOv9 究竟做了什麼-一起來一探究竟-9456740c9b0b
[12] A. Wang *et al.*, “YOLOv10: Real-Time End-to-End Object Detection,” arXiv.org. Accessed: Feb. 07,2025.[Online].Available:https://arxiv.org/abs/2405.14458
[13] R. Khanam and M. Hussain, “YOLOv11: An Overview of the Key Architectural Enhancements,” in *arXiv.org*, Oct. 2024. [Online]. Available: https://arxiv.org/abs/2410.17725
[14] Ultralytics, “SAHI Tiled Inference,” *Ultralytics YOLO Docs*, Nov. 12, 2023. Accessed: Feb. 17, 2025. [Online]. Available: https://docs.ultralytics.com/guides/sahi-tiled-inference/
[15] VisDrone, “GitHub -VisDrone/VisDrone-Dataset: The dataset for drone based detection and tracking is released, including both image/video and annotations.” GitHub. Accessed: Feb. 5, 2025. [Online]. Available: https://github.com/VisDrone/VisDrone-Dataset
[16] Taeyoung96, “GitHub - Taeyoung96/Yolo-to-COCO-format-converter: Yolo to COCO annotation format converter,” GitHub. Accessed: Feb. 18, 2025. [Online]. Available: https://github.com/Taeyoung96/Yolo-to-COCO-format-converter